Bachelor Thesis

Julius Schwarzweller

Fleming Kahn

Development of a Java based Cashier System using
Java POS and MySQL Database

Fakultät Technik und Informatik
Department Informations- und
Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and
Electrical Engineering

Julius Schwarzweller

Fleming Kahn

Development of a Java based Cashier System using Java POS and MySQL Database

Bachelor Thesis based on the examination and study regulations for
the Bachelor of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner : Prof. Dr. Hotop
Second examiner : Prof. Dr. Goerth

Day of delivery June 05th 2007

Julius Schwarzweller
Fleming Kahn

Title of the Bachelor Thesis

Development of a Java based Cashier System using Java POS and MySQL Database

Keywords

Java, MySQL, JavaPOS, database, cashier system, software engineering, POS, GUI

Abstract

Inside this report the development of a java software for a cashier system is described. This software is constructed for a flower shop. A graphical user interface supports the users' inputs. Automatic accounting as well as accounting data storage on a database system is implemented. A storage possibility of this data on an externally connected USB-stick is given. A special JavaPOS interface enables the software to control the cashier systems hardware devices (a receipt printer, a cash drawer and a line display).

Julius Schwarzweller
Fleming Kahn

Thema der Bachelorarbeit

Entwicklung und eines Java basierten Kassen Systems mit Hilfe von Java POS und einer MySQL Datenbank Software.

Stichworte

Java, MySQL, JavaPOS, Datenbank, Kassen System, Software Entwicklung, POS, GUI

Kurzzusammenfassung

In dieser Dokumentation wird die Entwicklung einer Java Software für ein Kassen System beschrieben.
Diese Software ist für eine Blumengeschäft konzipiert. Eine graphische Benutzeroberfläche unterstützt die Benutzer bei der Bedienung dieser Kasse. Diese Software bietet eine Funktionalität der automatischen Abrechnung sowie die Speicherung dieser Daten in einer Datenbank. Eine Speichermöglichkeit der Abrechnungsdaten auf einem extern angeschlossenen USB-sticks ist gegeben. Ein spezielles JavaPOS Interface ermöglicht eine Steuerung von Bondrucker, Kassenschublade sowie eines Kundendisplays.

Table of contents

List of Figures

1 Introduction

Electronic cashier systems have become very important tools within the last 10 years. They can be installed in small stores, restaurants, clubs / discotheques and many other places where a certain service is offered that has to be tabulated in a satisfactory manner.

The old system of using a cashier has only a few basic operations in common with modern cashier systems. A cashier system that was used years ago provided methods like the summing-up of product prices and the printing of receipts. These systems provided no graphic interface.
Accounting operations, printing receipts with logos, storage on a database were not provided by these systems.
A computer based cashier system, as it is used in our time, provides even more advantages such as easy navigation for the users. The most important feature that makes the use of a modern cashier system very efficient is the possibility that programmers can find a customized solution that completely fits the clients' requirements. The insertion of modern high level programming languages makes it possible.
Old cashier systems had only the basics because these systems had to be adopted for different areas of sale.

This developed software is to be used by florists. In the context of this documentation the florist (the shops owner) is appointed as client and customer is a person who buys something in the shop.
The used IBM cashier hardware together with a specific JavaPOS™ library system guarantees the operation of external devices such as cashier drawers, printers and line displays.

This report is divided into six major parts:

Chapter 1 contains the introduction. It presents the basic concept of the cashier system and an abstract of this report.

Chapter 2 presents an analysis of this project. Its purpose is to explain its application.

Chapter 3 deals with the requirements. The clients' requirements are obtained and listed here. Alternative solutions as well as the used hardware and software are discussed.

Chapter 4 describes the design phase of this software engineering process. The requirements listed form the basis for the design of the software. The requirements that can be implemented are transformed into classes, methods and variables.

Chapter 5 describes the process of creating the software. The design proposals are transformed into a high level programming language.

The 6th and last chapter offers a summation of the whole project. It presents the application of the solution. It discusses further developments as well as problems arising during the development phase.

This report also includes three supplementary chapters:
references, a glossary and the source code.

1.1 Projects Background

There are several reasons why the client might wish to install a modern cashier system.

Sales are tabulated in the old-fashioned way. The totalling-up of products is done with a pocket calculator and the articles sold are written into an account book. The accounting of the data has to be put into a computer program and regularly calculated. There is also no receipt for the customer so that after each transaction a receipt has to be written by hand.

Because the flower shop expanded several months ago, this way of accounting is not practical any more.

The insertion of an already developed modern cashier system is also no solution for the client. There are several POS (point of sale) systems available on market. But all of these systems offer too many possibilities for which the client has to pay but would not make usage of. The systems that are available on market provide features like warehouse keeping or automatic reordering of products.

For a small store, as in this case the flower shop, simple operations like summing products, automatic accounting as well as printing receipts with an own shop logo are sufficient.

Special functions like the possibility of storing accounting data in an external storage device make the assembling of such a software product mandatory. None of the software products available on market provide such a function in an appropriate way. That’s why the clients’ decision should be to commission the creation of a new software product. This task will be discussed in this report.

2 Analysis

2.1 Chapter Overview

This chapter deals with the important steps of analysis and project planning that have to be done in advance in order to develop a suitable product.

2.2 Software analysis

A very important source of information for planning a software project is the client itself. The client decides what the software product has to provide.
The software engineer's task is to understand how the particular application has to function.

First of all, the ideas and requirements for the software have to be assembled and discussed with the client. In this particular case the client is a florist, so that these special requirements have to be adopted.

The client has precise ideas what the cashier system has to provide. In several meetings all necessary requirements can be assembled.

The next part describes the clients' conceptions about the software for this project.
The software has to have a graphic user interface that is easily readable. Buttons for selecting the products have to be big enough so that the employees can quickly and easily find the product groups on the screen.
This user interface has to consist of different buttons for the products that are sold. The clients' products are divided into nine different product groups.
As these groups have different tax values, it has to be possible to change this product information without changing the source-code.
The graphical user interface has to show the actual date as well as the actual time on its screen.
One of the clients' requirements is that the number of products as well as the individual price can be inserted. Therefore a numerical insertion field within the GUI is desired. It has to be possible to delete already inserted products. A clear button

that resets all insertions has to be provided. The florist's shop's accepted means of payment are cash as well as EC-Card payment.
Because there is an external EC-card reader available, the EC-payment has to save only an entry on the accounting data that EC-payment was chosen.

A display has to show the bought items to the clients. A receipt has to be printed automatically containing the sold products.

Accounting is a central service this software has to provide. Therefore it is necessary that information on sold items be stored.
For the client, accounting has to be provided for an overview of income.
To discourage theft by employees, the money in the drawer at day's end must be compared with what was in the morning.
The employees who work in this store are paid from the money that is in the cashier's drawer. Each worker has his/her own personal ID, so that with this ID the amount that is paid can exactly be determined. This information also falls into the category of accounting.

The florist needs a safe and secure storage area for this information.
These data must not be changed or seen by the employees.

The accounting overview is done on another computer. Therefore there has to be a way to transfer these data to an external storage device. The accounting of sold articles is evaluated by the client at home.

The creation of the software using Java programming language is a clients' precondition that has to be fulfilled.

For a better understanding, the actual as well as the targeted situation are shown in the next section as decision charts.

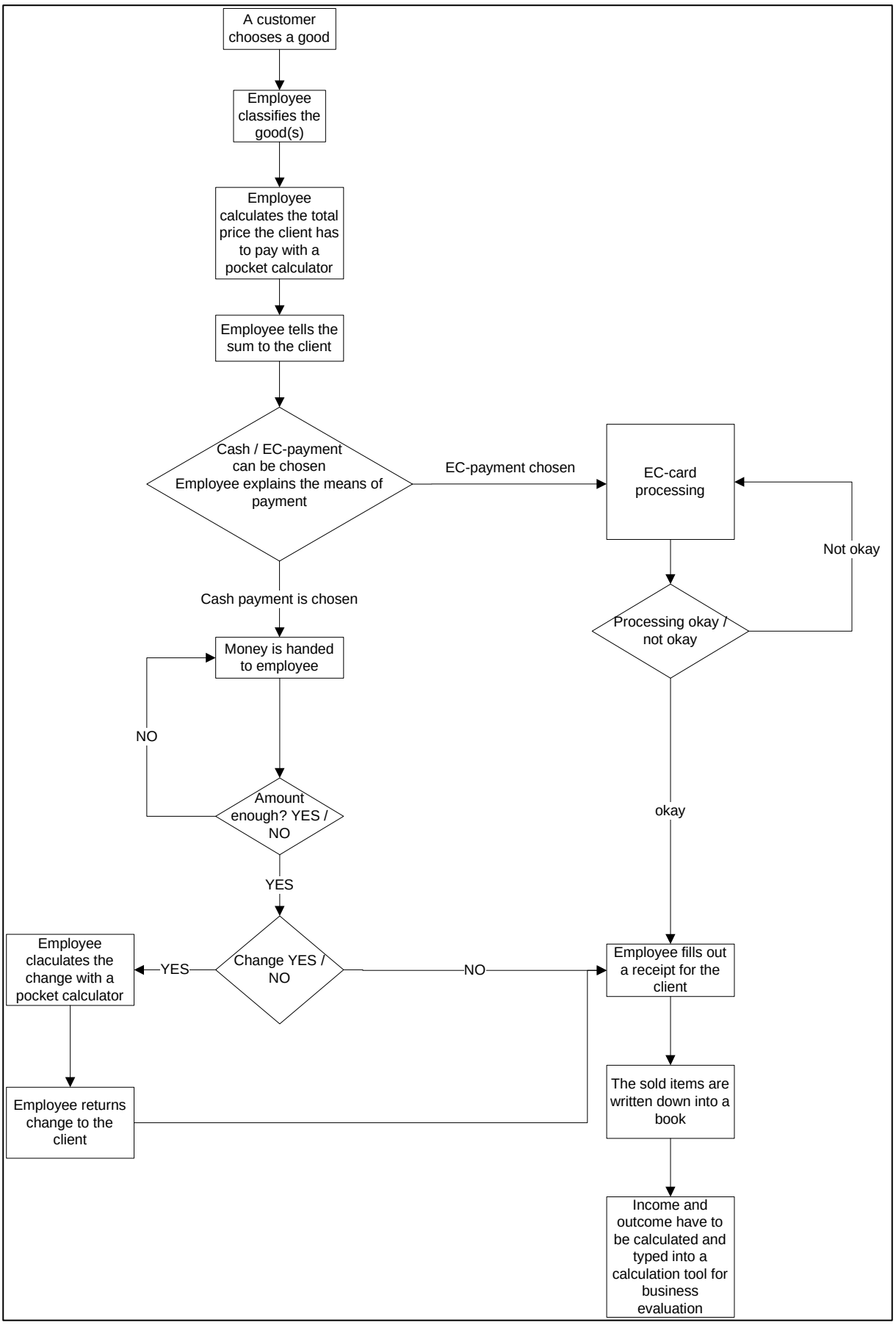

Figure 2-1: The process of buying a product as is actually the case

In this figure, the process of buying a product as it is actually done in the flower shop is shown.
The customer of the flower shop chooses a product. The employee classifies the product and calculates the total price the customer has to pay. The sum is told to the customer and the means of payment (by cash or by EC-card) are explained to the client.

If the customer chooses EC-card payment, an external EC-card reader processes the payment. If processing is successful, the employee fills out a receipt and hands it to the customer; otherwise the payment has to be repeated. The sold items are written down into a book. After an accounting term the income and outcome have to be calculated from these figures.

If the customer chooses payment by cash, the money is handed to an employee. The employee checks if the amount is enough and, if it is enough, if there is any change to be returned. All possible change is calculated with the help of a pocket calculator. The change is returned to the client. The employee fills out a receipt for the customer and writes down the sold articles into a book.
The income as well as the outcome is calculated with a calculator and typed into a computer calculation tool for business evaluations.

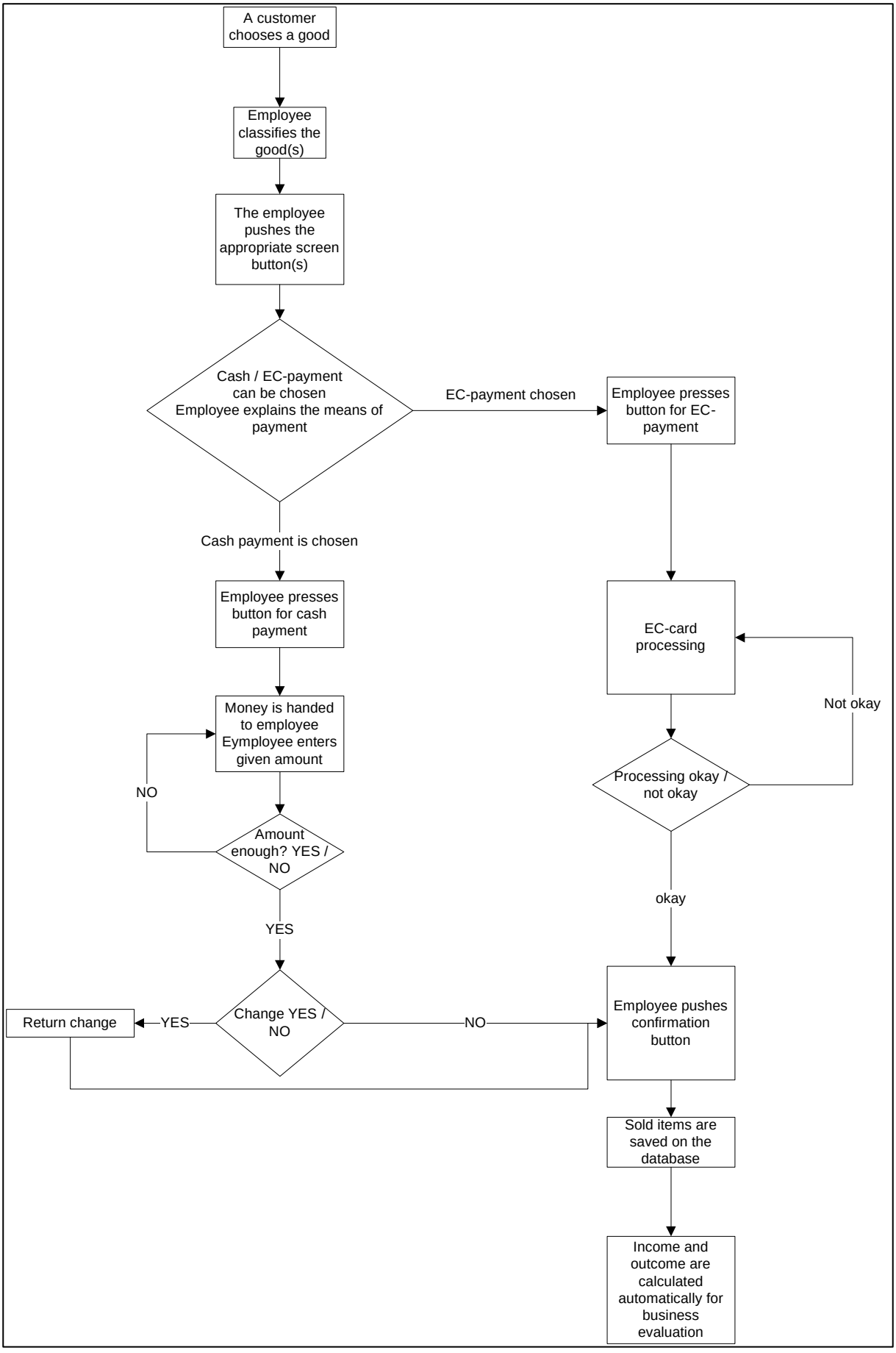

Figure 2-2: The process of buying a product with the electronic cashier system

In this figure a decision chart clarifies how buying a product looks like with the electronic cashier system.
The flower shop's customer first chooses an item he / she wants to buy. The employee has to classify the product, according to which of the nine possible product groups it belongs. The employee pushes the button for the matching product group and inserts the amount as well as the price.
After the employee has explained the valid means of payment, the customer has to decide whether to pay by cash or by EC-card.

If payment by cash is chosen, the employee presses the button for cash payment. The customer then hands the money to the employee who inserts this amount. The cashier system checks if the given amount of money is enough or not. The system calculates if change has to be returned or not. The shop's employee returns the change and presses a confirmation button. The sold items are stored in the database.

If payment by EC-card is choose, the employee presses the button for EC-payment. The EC-card is processed and checked for validity. If processing is not successful, this step is repeated. After the payment is made, the employee presses the button for confirmation. The sold items are stored in the database.

This solution has several advantages compared to the old one.
Accounting is done nearly automatically. No more calculations by hand need to be performed. No receipt has to be written anymore.
An insert of this solution saves a lot of time, the employee can use in order to server other customers.
The possibility to generate errors while calculating prices with a pocket calculator is not given anymore.

2.3 Hardware Analysis

2.3.1 Overview

The client requires a complete redesign of the old sales counter.
The old counter is nothing but a moneybox and a note book for writing the items sold and for keeping the accounting.
The client wishes to replace this with a computer controlled electronic Point of Sale system. A computer controlled electronic Point-of-Sale is a personal computer (PC) which consists of more than only display, main board, random accessible memory (RAM), hard disc drive (HDD) and human interface device (HID), but it additionally attaches special Point of Sale (POS) components. Points of sale (POS) components are devices such as receipt printer, scales, bar-code scanner and line display.

2.3.2 Characteristics of available Hardware

The client has already an own Point of Sale and it is expected that this system is used.

Figure 2-3: IBM SurePOS™ 500

The clients' hardware is the SurePOS™ 500 from the vendor IBM®, Type 4840-541. This is not the newest system. There are already newer, more advanced, faster and compacter systems on the market. The original system configuration is limited in terms of choice of the operating system, performance due to the limited amount of random accessible memory (RAM) or the slow hard disk drive (HDD) and operating noise.

The clients' point of sale (POS) system is from the year 2001. At this point of time the technology of personal computer systems were limited. So this system supports only the universal serial bus (USB) 1.1 standard, a comparable slow central processing unit (CPU) and limited capacity of random accessible memory. It was also the time where touch screens started to came into play with personal computing. With this model IBM supports touch screen drivers only for a few operating systems.

2.3.3 Improvements to be implemented in existing Hardware

The client also requested that some system limitations be improved. The operating noise of the system needed to be decreased in order to ensure a comfortable working environment at the sales counter. The hardware must be modified in order to reduce the operating noise.
But on the other hand a 'low cost solution' is demanded, so that the existing hardware must be optimized, but not replaced.

Certain hardware modifications are possible.

- The hard disk drive can be replaced with faster and quieter one.
- The random accessible memory can be upgraded to a maximum size of 512 MBytes.
- A new printer firmware version enables different character sets, larger print images which can be stored in the internal storage and faster data transfer time.
- The system and central processing unit fan can be replaced with a quieter one.
- The voltage of the fan can be decreased if the central processing unit is able to handle higher temperatures or a constant low central processing unit (CPU) load is used.

- The basic input / output system (BIOS) can be updated; this improves the hard disk drive capacity and also enables the use of different universal serial bus (USB) devices like USB-CD-Drive and provides the possibility of booting from a USB storage device.

2.3.4 Operating System Alternatives

Through the use of the printer and the touch screen the choice of the operating systems are limited. It is possible to use Linux Red Hat 7.1™, Windows 98™, Windows ME™, Windows 2000™ or Windows XP™ as an operating system.

2.3.5 Hardware / Software Interface

Another client requirement is to use Java™ as the programming language. This ensures that the program is later portable to other operating systems.
In fact, the client also requires a program that will be able to run on new versions of this hardware or on different systems from other vendors. Java™ is a purely object oriented language developed and maintained by SUN Microsystems®, but made available under terms of the GNU General Public License. Java™ can run on any hardware or operating system which supports a Java Virtual Machine™ (JVC).
In order to access the hardware components from the Java™ application, an extension of Java™ is required.
This can be accomplished by using JavaPOS™ as a hardware / software interface system. The usage of JavaPOS™ comes with certain advantages.

- The point of sale (POS) system is flexible in terms of different computer hardware and / or operating systems.
- It allows interchangeability of point of sale (POS) hardware components without changing the software program. For example the printer can be interchanged and only the hardware information file needs to be modified.
- The JavaPOS™ standard is maintained by an international multi-vendor committee which guarantees conformance to new hardware and hardware concepts.

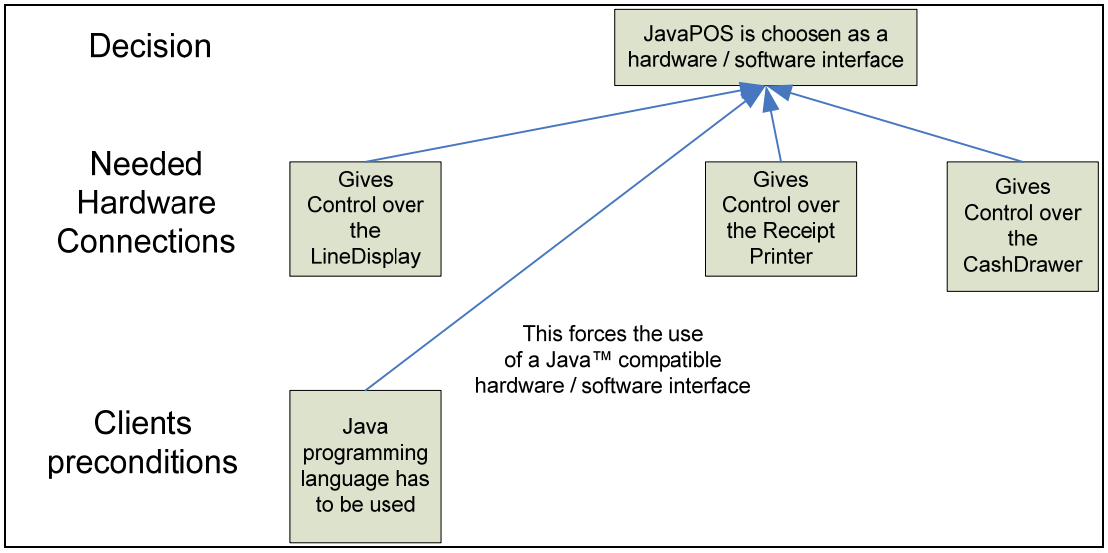

Figure 2-4: Decision logic JavaPOS™

This figure describes the decision which had to be considered.
The client requires that Java™ as a programming language has to be used.
Various types of external POS devices such as line display, cash drawer and receipt printer must be connected to the basic POS unit. Since Java™ itself does not have the capability to communicate with these devices directly, a Java™ compatible hardware / software interface is required. JavaPOS™ not only fills this requirement, but is based on an international non-proprietary standard, and will be a stable and dependable interface for years to come.

3 Requirements

3.1 Chapter overview

This chapter deals with the important steps of requirement analysis and project planning that have to be done prior to implementing the software.

3.2 General requirements

The client has overall requirements that hold for all further decisions. The first requirement is that a new software with the given requirements has to be created. No ready-made software is wanted because the client only needs basic functionality of such a cashier system. The software that can be bought on the market does provide too much functions, the client would not use but would have to pay.
So that an overall low cost solution is required. The florist already owns an IBM cashier hardware that has to be used in this project.

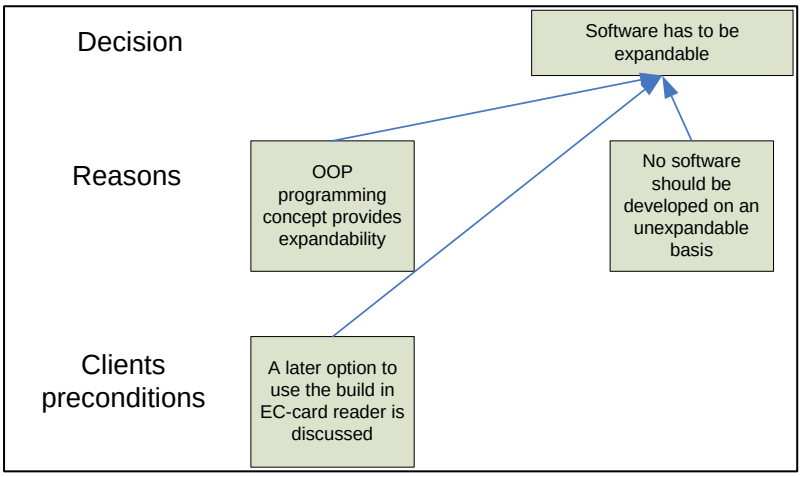

Figure 3-1: Decision logic, expandability of software

The software has to be developed on an expandable basis [3]. The first reason for this is that no software should be built on an undependable basis. The other reason for this decision is that a later usage of the hardware build in EC-card reader has to be possible. This decision parties the insert of a high level object oriented programming language.

3.3 GUI requirements

The client stated that the software has to have a graphical user interface. This is one of the central parts of this software product. A GUI (graphical user interface) enables the user to interact with a computer. This is done by providing the users with buttons, bars, info-displays and so on.

Such a GUI has to provide an easily manageable and concise way of operation. An intuitive method of operation has to be provided without special operations or functions that could disturb accounting procedures.

There are several steps necessary for such a GUI design to match the clients' requirements. The steps needed in designing a GUI are shown in the next figure.

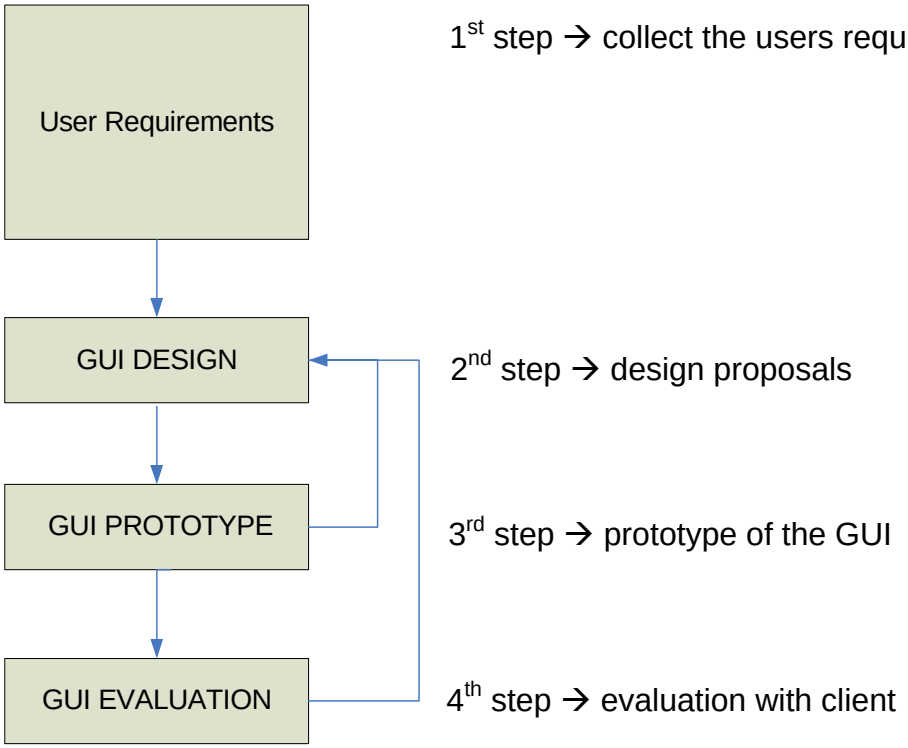

Figure 3-2: GUI-Design phase

The special requirements of a flower shop for such a GUI have to be pointed out to the prospective users of this software.

A decision logic is used to give a better overview, why the creation of a GUI is necessary.

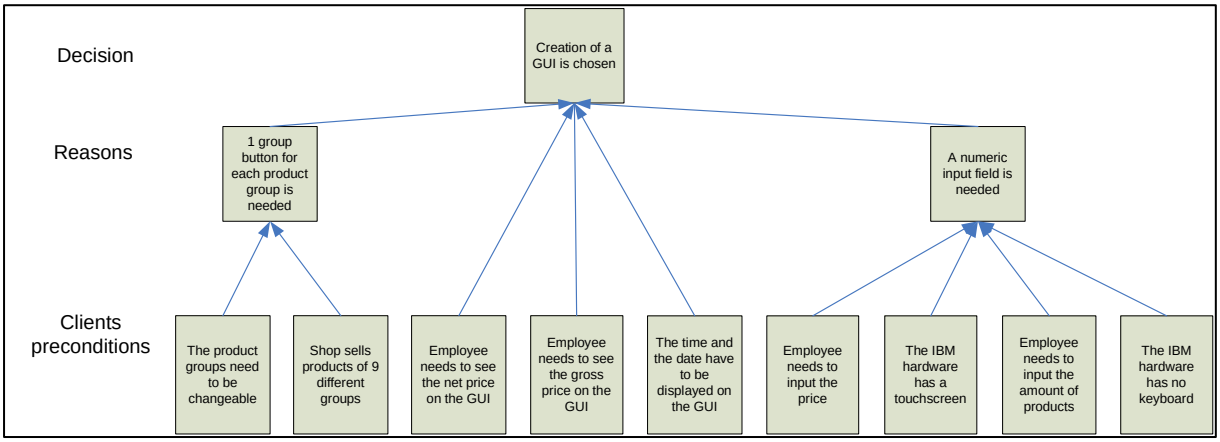

Figure 3-3: Decision logic, GUI

One requirement is that there be buttons for each product category. The shop sells nine different product groups that are differently taxed. The information (product name and tax value) of these product groups have to be changeable so that they are easily adjustable.

These buttons have to be as big as possible and there has to be enough space between the different buttons in order to prevent erroneous inputs.

A numeric input field is needed because of several requirements. It has to be possible that the employee inserts the prices as well as the amount of products with a numeric keyboard. The IBM hardware has a touch screen but no keyboard.

Two possibilities to pay the amount are accepted in this shop. Pay by cash or by EC-card are accepted means of payment. These payment methods need to have their own buttons on the GUI in order to choose the desired payment method.

These requirements make the insert of a numeric input field mandatory. It can easily implemented by providing a button for each number as well as a button for deleting the input and a cancel button to abort the transaction.

As earlier pointed out, accounting operations have to be possible when using the cashier software. There need to be different sections that contain buttons, input fields and everything else that is needed to do accounting.

The idea is to create a frame that holds all products; a frame that holds the chosen products as well as a frame that holds the numerical input field.

3.3.1 Prototype

After the clients' requirements are established, preliminary ideas how such a GUI would look are developed.

A first design proposal is drawn by hand in order to give the client the possibility of response and interaction.

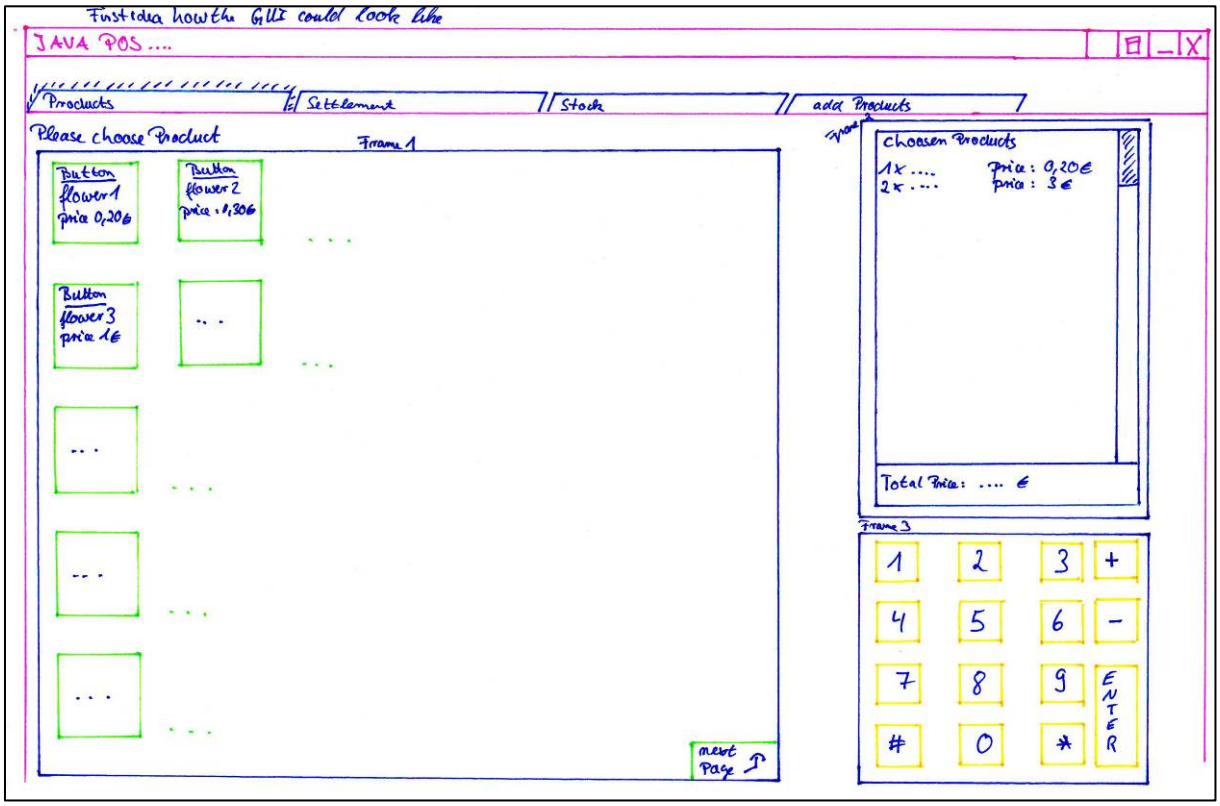

Figure 3-4: First paper work of GUI

This first paper work version is discussed with the client, whose reaction is positive. The division into three different frames is not accepted. Another frame that shows the amount as well as the price of a sold product has to be implemented. The design of the numerical input field is as expected. The idea to place the numerical input field on the right side of the GUI is desired by the client. Through the fact the most of the employees are right handed and the numeric input field is a main navigational element, this design proposal is better suited as other solutions, e.g. placing the input field on the left side.

The client desires bigger product group buttons. In this early prototype the methods of payment are not included so that this and the additional frame design for the amount and price of the products need to be enhanced in order to fit to the clients desires.

3.3.2 Toolkit

The decision that a GUI has to be created involves a choice of a toolkit with that the GUI is created. Toolkits are sets of basic building units for graphical user interfaces.
The client demands that the software is easy to maintain and modifications can be performed without special knowledge. This requires an easy and fast configurable toolkit that is known by the client as well as the programmers.

For Java™ there are different possibilities for designing such a GUI **[4]**, as it can be seen from the next figure.

	SWT	Swing
Advantage	Low resource consummation	Platform independent
Disadvantage	Runs only on SWT implemented platforms	Resource hungry

Figure 3-5: Advantages SWT compared to Swing

The SWT (Standard Widget Tools) libraries were developed in 2001 by IBM for the software development editor Eclipse. SWT is now maintained by the Eclipse Foundation and works together with the Eclipse IDE. The SWT libraries are written in pure Java which makes its application area quite large.
The most important advantage of using SWT compared to Swing is its performance. SWT uses, compared to Swing, the native graphic elements of the operating system that make processing faster and increase speed. Another point that supports the decision to use SWT is that it offers a wider range of functionality. An advantage of Swing is that it is platform-independent. This independence is not that important compared to lower resource consumption. A disadvantage of Swing is that it is very resource hungry. This does not accord with the chosen hardware so that this solution was discarded.

3.3.3 Final GUI

From the gathered requirements and the clients' information about the paperwork GUI version, a new GUI is created using SWT. The next figure shows a design for the GUI that is constructed using SWT.

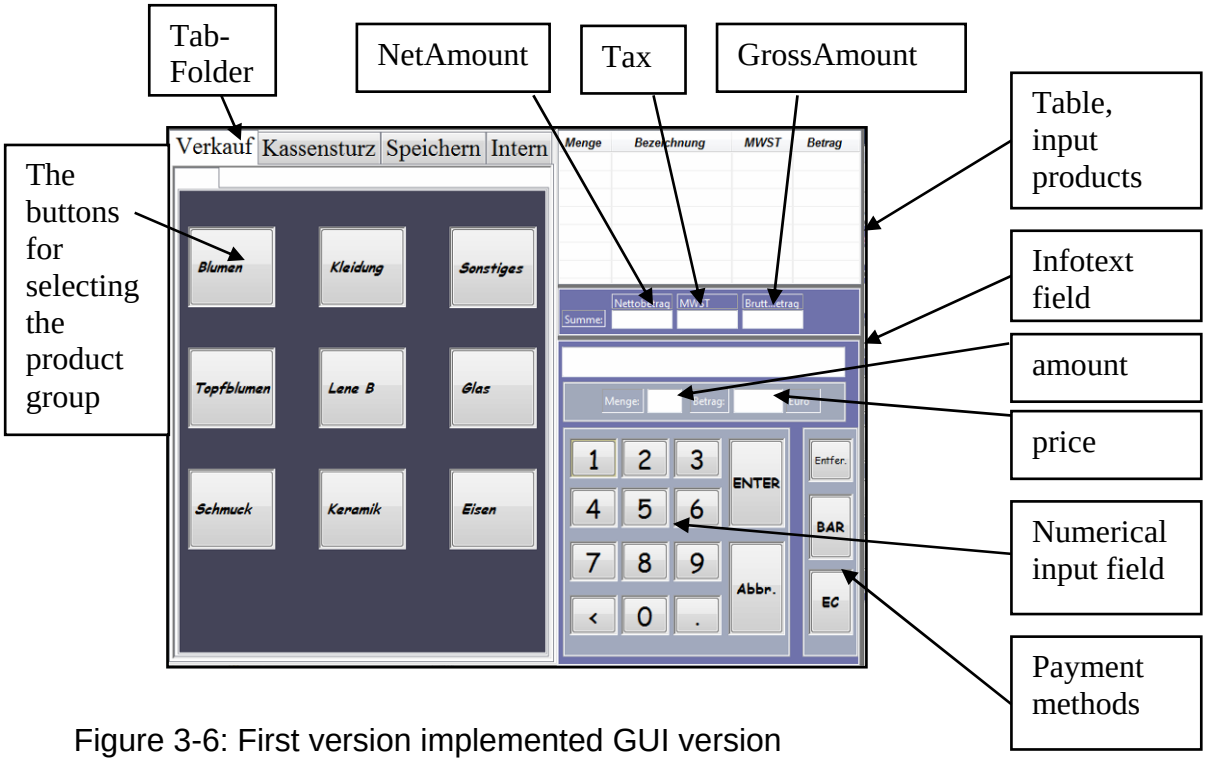

Figure 3-6: First version implemented GUI version

A proposal to have blank product buttons that are filled from a database is not acceptable because a flower shop has changing products and hundreds of different flowers. Saving the complete arrival in a database and showing these products as buttons would mean a very high administrative workload. Every single flower needs to be insert into this database. This fact influenced the clients' decision against this proposal.

A solution is to create a button for each product category.
The GUI consists of several tabs; the first tab is that where products can be sold; a second tab where the amount of money in the drawer can be calculated. A further tab "Speichern" is responsible for saving the data from the database on the external storage medium. Tab "Intern" makes the accounting of the worker's salary possible.
The number of products as well as the single price is insert by the numerical field.
The "Enter"-Button is the main navigational button of this GUI. Depending on which tab is visible, inputs are confirmed by this button.

The payment buttons are also implemented. The customers can choose either to pay by cash or EC-card.

This enhanced version of the GUI fits to clients' requirements. The time as well as the date is missing in this version but will be implemented in a final GUI version.

3.4 Database requirements

The clients' conception is to use a cashier system that does accounting automatically. A very important requirement for the user is to store accounting information in a very secure way, so that after some years access to this data is still guaranteed.
The requirements are to store each transaction that is made (whenever something is sold). There has to be the possibility storing an accounting summary at the end of each accounting term (one month). The last requirement is that the salaries taken out of the cash drawer for the employees need to be stored too.
A general requirement for all shops that sell products is that the data storage is a duty by law. This means that data about sold articles needs to be stored for a defined time period.

Persistence and data integrity of data are important arguments to decide to use a database system.

In general the advantage of using a database is the enduring management of data. Efficient maintenance of data is guaranteed; the access to specific records can be provided very fast. A database offers special operations for modifying / selecting data. Support for the transaction concept (operations that belong together are carried out as a whole) is also given. Data security through adoption of user roles and user rights is provided. A database is able to support multi-user operations.

There are several other possibilities of storing the accounting data on a database. An alternative would be to store this information on text-files. The advantage of this solution would be its easy implementation into the code. But the disadvantages

outweigh the advantages. The retention of data cannot be guaranteed as it can be when data are stored in a database system. One of the clients' requirements is that the employees shall not have access to these files, which could result at worst in the deletion of files. The time needed to access these files is more than that for a database. The file has to be read in as a whole.

These requirements and the disadvantages of file storage make the insertion of a database mandatory. In general a database is a collection of records or information which is stored in an appropriate way on a computer.
The computer programs pose queries whose answers are returned to the calling program.
This kind of program that poses queries to a database is known as a database management system (DBMS).
There are several ways of modelling the database structure. The model that is used in this type of application is the relational model. This model represents all information in the form of multiple related tables, each consisting of rows and columns. There are other models in usage like the hierarchical model and the network model; the application areas are more explicit representations of relationships.

3.4.1 Database solutions [5]

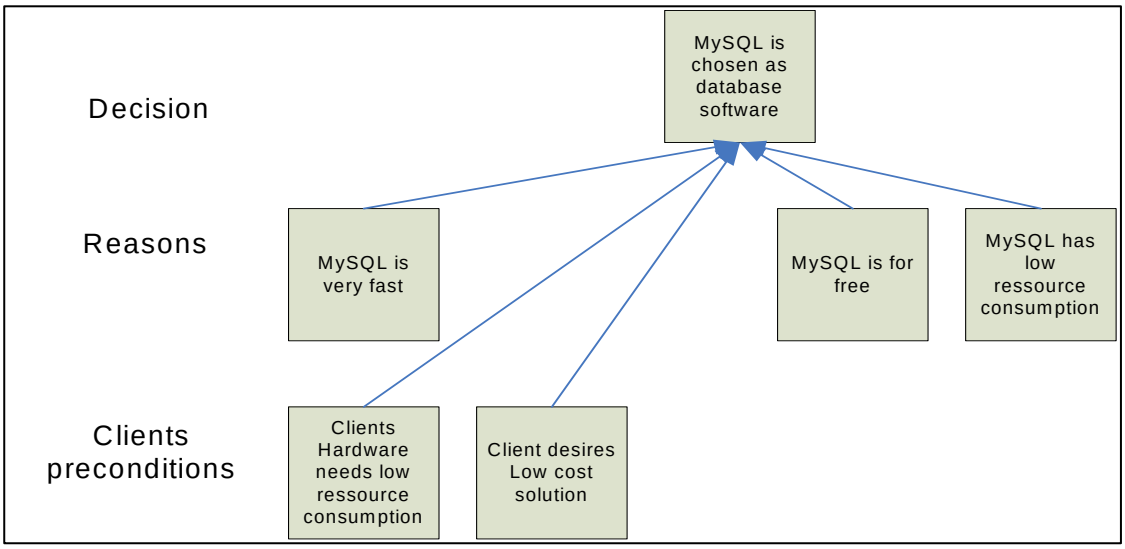

Figure 3-7: Decision logic, MySQL

The decision logic shows the reasons and preconditions, why MySQL is chosen as the database software.

Several database systems are available on the market. One must choose which database software best fits this project. The advantages and disadvantages of the most important ones will be discussed in the next figure.

Database System	Description
Oracle	▪ Developed & sold by Oracle ▪ "The Global Player" ▪ Expensive ▪ Cluster concepts ▪ Relational DBMS ▪ Stored Procedures ▪ More security, enhanced backup possibilities ▪ For big databases with huge archives
DB2	▪ Developed & sold by IBM ▪ Expensive ▪ Relational DBMS ▪ Database in DB2 is a logical grouping of table spaces
Microsoft SQL Server	▪ Developed & sold by Microsoft ▪ Expensive ▪ Late development compared to Oracle & DB2 ▪ Powerful architecture
MySQL	▪ GPL (General Public Licence), Program is free ▪ Functionality small compared to Oracle ▪ Field of operation is small databases

Figure 3-8: Compares database systems in use today

Database systems that are available on today's market differ in their functionality and price. The Oracle system is suitable for huge databases that require a professional level of integrity and retention; Oracle supports cluster concepts and has many stored procedures. The big disadvantage of Oracle is the price. Oracle is very expensive and support for such a database system can only be obtained by signing a maintenance contract.

IBM's DB2 database software is also very expensive compared to other database systems. The advantage of DB2 is that this software is very powerful and supports logical grouping of table spaces.

Microsoft SQL Server offers a very powerful architecture. It is a late development compared to Oracle & DB2. The biggest disadvantage is its high price.

MySQL is better suited for smaller databases. It is licensed with the GPL-licensing-system (General Public Licence). This means that use of this program is free. Low cost consumption was a very important clients' requirement in designing this project. This makes the insert of MySQL very interesting. The resource consumption of this software is very low; through the low processor performance of the cashier system, a clean, well programmed code of the database system helps to increase the processing speed when writing or reading from the database.

MySQL offers good flexibility because there are versions for Linux, Windows® and UNIX® available. This makes the cashier system more flexible because one could then think of installing a Linux operating system.
MySQL offers saving and recovering tools that help to prevent a complete loss of data. This fact is very important for the client. The data need to be stored, so that security is ensured.
These advantages of MySQL justify the choice of MySQL as the database for the cashier system [1].

The MySQL version that will be used for this project is 5.0.27.

3.4.2 Database storage possibilities

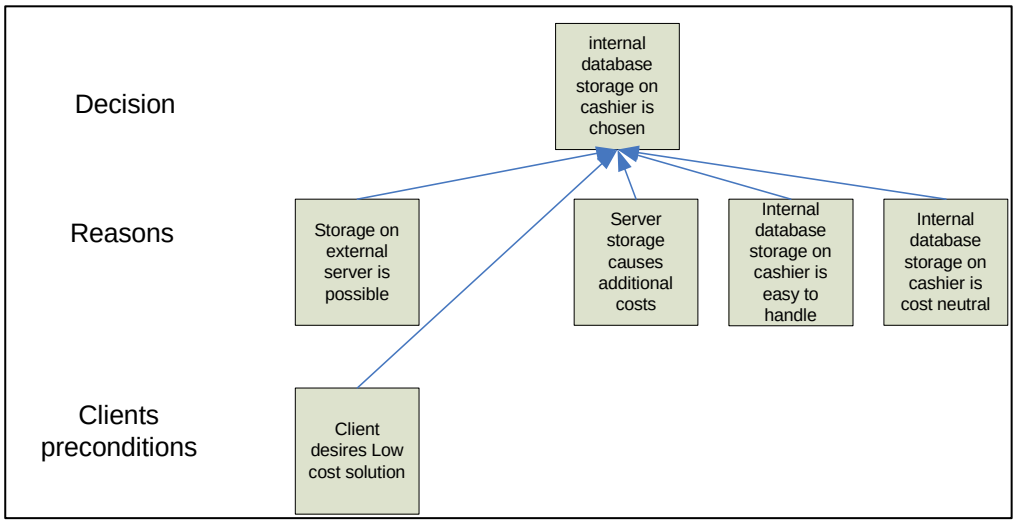

Figure 3-9: Decision logic internal / external database storage

Another decision that has to be made is where to store the database. A database can be installed internally on a computer that is database server and database client at the same time. In this application, internal storage would mean to install the database software on the cashier system.
Another method of database usage is to store the database software on another external server. The computer clients would then be connected to this server via a LAN or the internet.

	Pros	Contra
internal database storage	▪ No additional hardware needed ▪ Administration easier, database and user software located on one machine	▪ Database access burdens system performance ▪ In case of failure complete database is offline
External database storage	▪ Fail-proof ▪ Other applications can use database	▪ Expensive, additional hardware needed ▪ Wiring of cashier needed for LAN or internet access

Figure 3-10: Pros and contras of internal / external database storage

For internal database storage no additional hardware is needed. The database software can directly be installed on the cashier system. This also means that the administration is easier because the database and the user software are installed on one computer. There are disadvantages in that the database access burdens system performance. The possibility that more than one computer can access the database does not exist. And in the case of a computer's failure, that complete database is also offline.

External database storage can be accomplished by using a server that holds the database software. Such a system would be nearly fail-proof and there is the possibility to let other applications access the database. The main disadvantage of this solution is the additional cost. The cashier would need to be wired for LAN or Internet access.

The decision to install the database software externally on a server is rejected because the cost for the implementation would be too high.
The cashier software is set up so that it is possible to integrate a server that holds the database. For this, only slight changes in the source code need to be done and the cashier system will need a connection via a LAN-network, the internet or a VPN-tunnel to a server.

An internal storage is chosen because of the lower costs that this solution produces.

3.4.3 Database access

A so-called MySQL-Connector is used to enable the Java™ software to communicate with the database. This connector provides connectibility for client applications, in this version for Java™ programming languages. It converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database. This driver is written in pure Java™, which makes its insert into a Java™ project very easy.

As it is the MySQL database, MySQL-Connector/J is available under the GPL [6] (General Public Licence) license system. This means its use is free.

There are currently two different versions available:

An older version MySQL Connector/J 3.1 and the current version MySQL Connector/J 5.0. Because that version 5.0 offers 50 – 100 % more speed when accessing the database the decision, which version to use, is quite easy. This information is taken from the MySQL-Webpage, where the possibility of increasing processing speed when using the latest version (5.0) of MySQL-connector/J is explained [7].

3.5 Exporting of accounting data

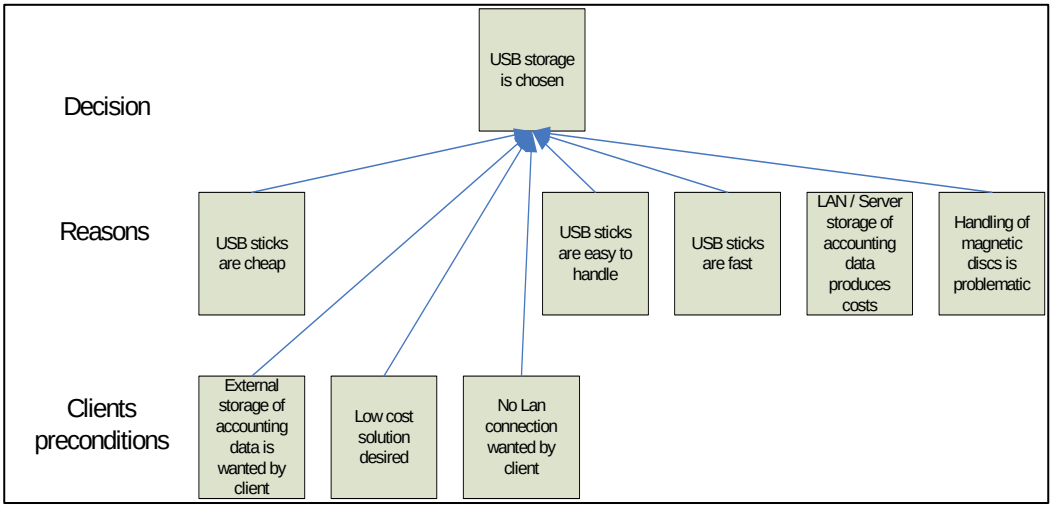

Figure 3-11: Decision logic, USB storage

External storage of the accounting data is a condition that needs to be fulfilled within this project, because the client needs to have a way to transfer the accounting data onto an external storage device, if he / she wishes to evaluate this information at home. Such evaluation is useful preparing and calculating business taxes as well as other expenditures.

The client does not have the time nor the technical knowledge to query the database for the needed accounting data. It is therefore necessary to easily and quickly access this data with one click on a button.

There are several possibilities for implementing this storage.

One possibility is to store the data externally onto an USB-Stick (Universal Serial Bus) or a USB-driven external hard drive. Such a USB device is connected to one of the two free USB ports of the cashier's hardware. USB offers high performance and is hot-plug -/ -unplug-able (device can be inserted and removed during operation of the computer).

Another advantage of storing the data on an USB-Stick is the cost-efficiency of such a storage device. USB-Sticks are smaller, more portable and cheaper than external USB-hard disc drives.

Another possibility is to store the data on another computer via a LAN or the Internet. This is not desirable because one of the clients' requirements is to have an overall low cost solution. An LAN access installation would produce high cost.

The third possibility would be to store this data on a magnetic floppy disc. These discs are portable but the disadvantages of such a solution outweigh this advantage. The cashier system does not have a magnetic disc driver built in. This disc drive needs to be bought. These discs are very sensitive, a failure of magnetic floppy discs can happen because of wrong usage.

Storage on an USB-Stick is the best choice with fewest disadvantages. The data are stored in the Excel-file format. The client evaluates the data using Microsoft Excel.

3.6 Programming Language

3.6.1 Overview

As already discussed, Java™ is chosen to be used as the programming language.
Java™ has the ability to run under every hardware or operating system which is supported by the Java Virtual Machine (JVM).
But the choice of the operating system is limited by the operating system support of the different hardware drivers.
This project can be used under the following operating systems: Linux Red Hat 7.1™, Windows 98™, Windows ME™ windows 2000™, windows XP™.

Java™ brings along certain advantages.
With this programming language encapsulation, inheritance, polymorphism can be used.
The main benefit comes with the use of classes, methods, and instances which allows a real object oriented software development of this cashier software.

3.6.2 Java™ Advantages

- **Encapsulation** is the concept of hiding the implementation details of a class and allowing access to the class through a public interface. An instance variable of the class needs to be declared as private or protected. The client code can only access public methods rather than accessing the data directly.
- **Inheritance** is a major component of object-oriented programming. Inheritance allows the user to define a very general class, and then later define more specialized classes by simply adding some new details to the original more general class definition. This saves work, because the more specialized class inherits all the properties of the general class and the programmer only needs to program the new features.
- **Polymorphism** means "any forms." In object-oriented programming, it refers to the capability of objects to react differently to the same method. Polymorphism can be implemented in the Java™ language in the form of multiple methods having the same name. Java™ code uses a late-binding technique to support polymorphism; the method to be invoked is decided at runtime.
- **Class**: The basic unit of object-orientation in Java™ is the class. The class is often described as a blueprint for an object. It allows a programmer to define all of the properties and methods that internally define an object.
- A **method** is a group of instructions that is given a name similar to a sub-routine in conventional programming. A public method can be called from outside the class, whereas protected and private methods can only be called from within the class. A method can return a result, or change variables within the class.
- An **Instance** is an object of a particular class. In programs written in the Java™ programming language, an instance of a class is created using the "**new**" operator followed by the class name. Each instance of a class functions independently and can have different variable states from other instances of the same class.

3.6.3 Java™ Versions

In order to use Java™ as a programming language it has to be ensured that Java™ can be used on the clients' hardware. To run a Java™ based application a Java Virtual Machine (JVM) is needed.
This Java Virtual Machine is available for different platforms.

In this project different Java ™ versions [10] can be used. For simplicity sake only the main three versions are described.
Available are: Java 2™ (1.2 to 1.4.2) [10], Java 5 Tiger Release™ (1.5) [10] or Java 6 Mustang Release™ (1.6).

A few remarks to each version in terms of this project:

- Java 2 ™ is already included in Windows XP™, so that there is no need to install additional software on the system.
- Java 5™ extends Java 2™ with more convenient functions like "linkedlist". It is easier to program such a cashier project with this version.
- Java 6™ is similar to Java 5™ in terms of the functional requirements of this project. The advantage is that it operates much faster.

3.6.4 Benefits of Java 6 Mustang Release™

Advantages of running applications on Java SE 6

- Applications run faster on the desktop and servers
- New 'Dynamic Attach' diagnostics simplify troubleshooting
- Improved 'native' look and feel across Solaris, Linux, and Windows
- First Java™ platform with full support for Windows Vista

Benefits in upgrading developer environments to Sun's Java SE 6

- JavaScript integrated and included with the platform
- Scripting languages framework extends support for Ruby, Python, and other languages
- Complete light-weight platform for web services, right out of the box
- Simplified GUI design and expanded native platform support
- Full JDBC4 implementation providing improved XML support for Databases

- Java DB included with the JDK, a free to use and deploy Java Database
- Full support by NetBeans IDE 5.5
- Sun Developer Services available to help build more robust applications

3.7 JavaPOS™ [2] Hardware / Software Interface

3.7.1 Overview

When developing a point of sales retail system, the question has to be addressed, how to interface with the hardware components. Theoretically it is possible to use the classes of the JavaComm Package to address the hardware devices through the RS-232 serial communication port directly. However extensive and specific knowledge of individual hardware and software protocols are necessary to realize this. Quality control becomes very complicated, because the total number of interface combinations to be tested and verified grows exponentially with the number of devices and commands. Furthermore this solution would be completely inflexible, requiring changes to the program code any time the hardware or operating system changes.

JavaPOS™ however supplies a layered interface architecture which isolates the application program from the hardware devices. This means that one standard command from the application program calls the JavaPOS™ application programming interface (API). JavaPOS™ then adapts this call to the specific protocol requirements of the hardware device, sends the message and monitors and controls the response.

3.7.2 History

The first version of a layered POS architecture was OPOS™, which was based on Win32 operating systems. This version became available in March 1996.

Soon after Java™ became available on the market, several developers of retail applications recognized that the Java™ language offers major advantages for the development of point of sales software. However OPOS™ could not be implemented in a Java™ environment. In April 1997 a meeting was held where a collection of retail

vendors (including IBM and NCR) and end users examined the ways in which these Java™ advantages could be fully exploited in the retail environment.
In March 1998 after ten months and 6 more meetings this committee put forth the original JavaPOS™ programming standard (v1.2). Later in 2001 JavaPOS™ and OPOS™ were combined into the Unified Point of Service™ (UPOS™) specification.

3.7.3 Advantages of Java™ and JavaPOS™ in the Retail Environment

Points of sale (POS) systems which are conformant to the JavaPOS™ standard provide several significant advantages for the retailer. For sites with large numbers of point of sale (POS) terminals, these advantages can result in a considerable savings in system administrative overhead costs.

Platform Independence:

- The JavaPOS™ standard utilizes the java virtual machine (JVM) as its retail platform, whether present in a browser, an operating system, or directly embedded within the microcode of a specialized computer chip.

Reduced Point Of Sale (POS) Terminal Costs :

- Applications written in the Java™ language executes by having a Java Virtual Machine (JVM) interpret their platform independent byte codes. These applications will therefore execute wherever such a Java Virtual Machine (JVM) is present.
- By lowering the minimum requirements for a Point of Sale terminal to a system capable of supporting a single Java Virtual Machine (JVM), the JavaPOS™ standard enables retail applications to be run on thin client platforms which are often less costly than more traditional Windows PC configurations.

Reduced administration costs of thin clients:

- The capability of the Java™ language platform to reside on thin clients offers additional cost savings to those sites that run JavaPOS™ compliant applications. If the client portion of a retail application is written as a Java™ applet or loadable application, all of the application code resides on the in-store server. This means that installing or upgrading the Point of Sale (POS) software on the server results

in the automatic loading of the new software into each local Point of Sale (POS) terminal, when the terminal is next booted.

- The system administrator need only install a retail application once, and it becomes installed everywhere in the store. Fix the application once, and it is fixed everywhere.

Increased Security and Stability:

- The absence of persistent storage on a thin client also eliminates the need to perform Point of Sale (POS) terminal data backup and recovery.
- However local disks could still be useful to cache pricing information and outgoing purchase transaction data, so as to allow the thin client Point of Sale (POS) terminal to continue to function effectively, even if the server connection was lost for an extended period of time.
- When the server connection is available again, then the cache is resynchronized with the server, and the terminal continues to operate in online modus. This combination of stand-alone and client-server functionality greatly increases system availability and stability.
- The fact that no critical data is stored long term on the client increases data security.

3.7.4 JavaPOS™ Architecture

JavaPOS™ is a system which enables a Java™ based application to communicate with the specific external hardware of a Point of Sale (POS) like the line display, the cash drawer or the receipt printer.

Because of its Java™ based implementation, it is standardized for usage on different operation systems like Windows™, Linux™, and other Java™ runtime time environment (JRE) equipped platforms.

JavaPOS™ provides a mechanism for Java™ applications to control Point of Sale (POS) hardware peripherals in a vendor neutral manner.

Hardware from different vendors can be controlled via the same Java™ application.

To achieve the device vendor neutrality, a three layer architecture is created with the lowest layer being provided by the hardware device vendors.

JavaPOS™ is more than just software. It is in fact a blueprint controlling how interfaces are to be designed, so that they work interchangeably between different hard and software vendors. That is why it is important that it is a committee controlled standard and not a proprietary system. With this standard, any hardware vendor can design a device for Point of Sale (POS) functions, and it will be available for any number of software applications.

3.7.5 Three layer model of JavaPOS™

JavaPOS™ based solutions have three architectural layers.

1. The upper layer is the application program itself, which is written in Java™.
2. The JavaPOS™ Device Control Layer. This layer presents a set of classes and methods for the application layer to use. The application creates an instance for every device that it communicates with. When an instance is created, the Control Component Layer communicates with the service component layer to create a connection to the device. The Component Layer knows all the attached hardware devices and makes these devices available to the higher Application Layer.
3. The JavaPOS™ Device Service Layer is the lowest level of software. This layer is provided by the device manufacturers. It implements all protocols necessary for the operation of the device, for example with a receipt printer the baud rate, handshake or other parameters needed.

With this architecture, devices with the same functionality are interchangeable without changing the Java™ application program.

Graphically, this architecture looks like this:

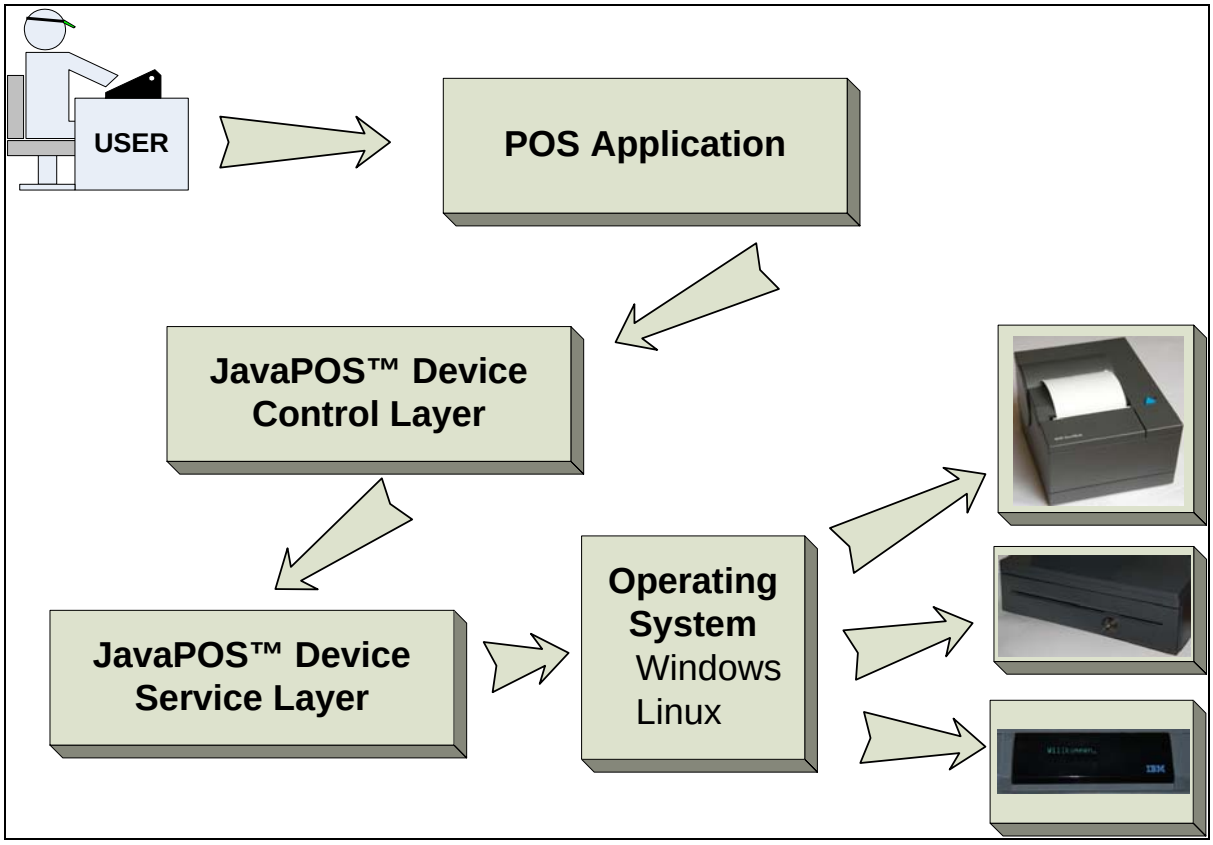

Figure 3-12: Layer description

The operating system (Windows™, Linux™, or other) exists below the JavaPOS™ Service Component Layer; it is followed by the Point of Sale (POS) hardware (connected via COM port or others).

The JavaPOS™ Service Component Layer must include some mechanism for interfacing with the operating system and hardware devices. The JavaPOS™ Service Component Layer uses the JavaComm Package to control the communication interfaces (Serial (COM), Parallel (LPT) and Universal Serial Bus (USB) ports). An alternative is use the Java™ Native Interface (JNI) technique. When Java™ Native Interface (JNI) is used at the JavaPOS™ Service Component Layer, the vendor of this layer will typically provide an operating system specific shared library (like windows.dll).

3.7.6 Interaction between the JavaPOS™ Layers

A deeper explanation of JavaPOS™ shows how it works in detail.

The JavaPOS™ specification categorizes physical devices according to their properties and communication requirements. The JavaPOS™ software includes a Java™ class for each category, supplying properties, events and methods necessary to control this type of device. This Device Control Class is in the Control Component Layer.

In order to communicate with the physical device, the device has to have a name common for all three layers. This name is used in the application programming when creating an instance of Device Control Class.

The Device Control Class then connects to the Service Component Layer by creating an instance of a class in the Service Component Layer.

This is done by using the Java Configuration Loader™ (JCL™) to get the details describing this device out of an XML file. This information is found in the XML file under the name which is passed from the application. There the Device Control finds the correct *category*, *service class* and *version* which it needs in order to access the Service Component Layer correctly.

The service control layer once again uses the application name and the Java Configuration Loader™ (JCL™) to access the XML file, and extracts the *device specific information* that it needs to communicate with the hardware, such as *baud rate, hand-shake* or *COM port number*.

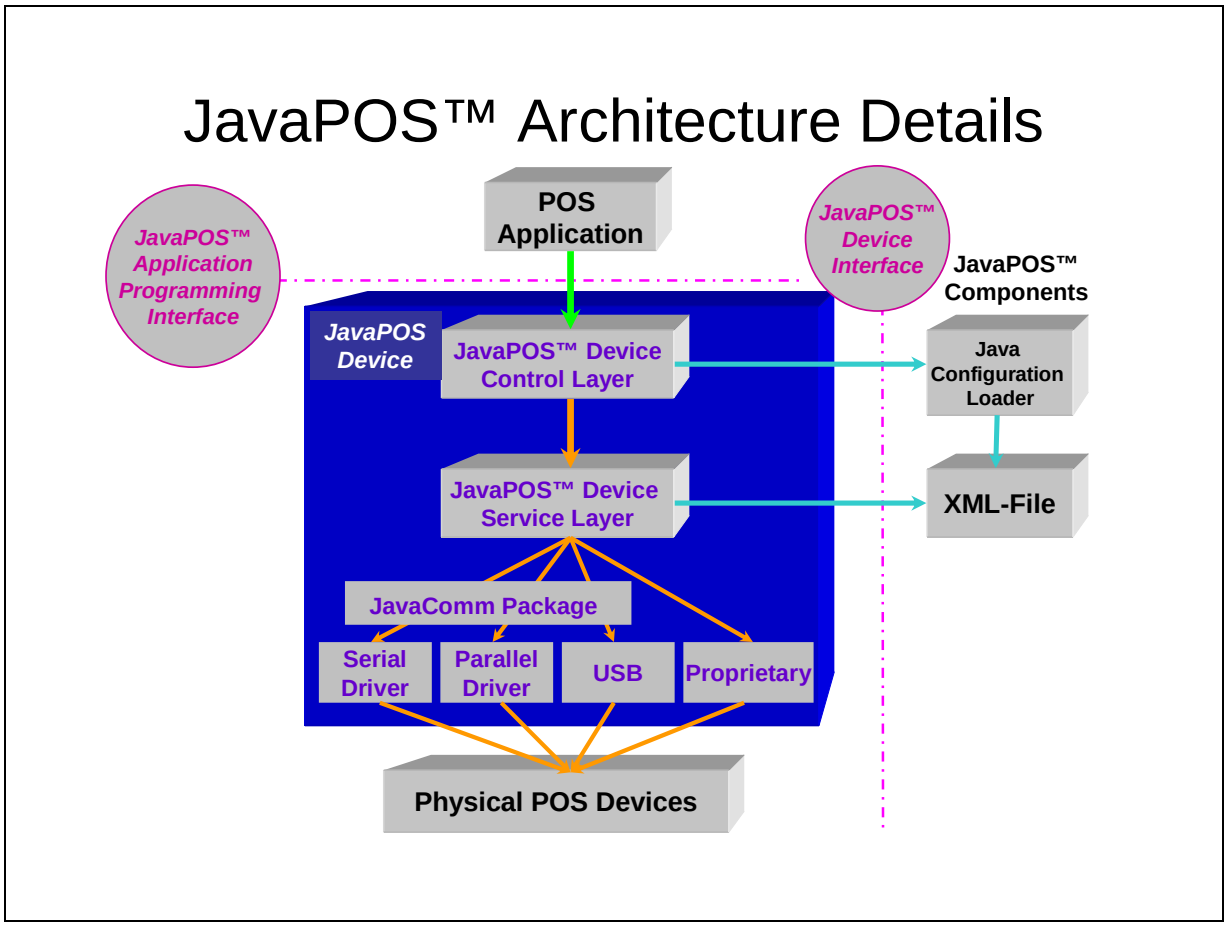

Figure 3-13: JavaPOS™ detailed architecture

3.7.7 Java Configuration Loader™ (JCL™)

The jpos.config / loader (JCL™) is a very simple loading / configuration Application Programming Interface (API) which searches in an XML file for a device name, and returns a flexible set of parameters, describing this device.

This technique allows differing Point of Sale (POS) terminal device configurations to be supported by a single software. The **device name** is the link binding the application program call to the properties of the device. If a device is exchanged or linked to another port, only the xml file has to be changed. In a sense the Java Configuration Loader™ (JCL™) functions here as a sort of Java version of the NT Registry.

Because the Java Configuration Loader™ (JCL™) uses a "plug-in" architecture, this also allows third-parties to define their own config / loader if they wish.

3.7.8 Device Control Class

This layer is responsible for supporting the retail application's use of the Device Interface, and it decouples the application from the lower device layers.

Each Device Control defines a unique retail device category in terms of a group of properties, events and methods.

A Device Control can be considered as a fairly thin wrapper for its corresponding service. It has the following responsibilities within the JavaPOS™ architecture:

1. Properties – Make Device Service properties visible to the application, and notify the service if the application changes them.
2. Events – Propagate Device Service generated events up to the application level.
3. Methods – Forward all application requests for a device directly to the Device Server, and return status or repost any device exception directly back to the application.

The Device Control Class stands in an 1 : N relationship to the Device Service Classes. This means that the JavaPOS™ specification defines a particular Device Control Class for a category of devices. Now the device vendors must develop and deliver Service Control Classes which correspond to this category. Therefore there could be many Service Control Classes for a single category, but only one Device Control Class.

The Device Control Class functions by using the name of the device given by the application to access the XML file through the Java Configuration Loader™ (JCL™) call. It then determines exactly which Device Service Class to connect to, and checks the version. When it accesses the Device Service Class, it checks the version compatibility, and if it is not compatible, reports this to the application program through an event.

3.7.9 Device Service Class

The Device Service Class opens the physical connection to the Point of Sale (POS) device.

It also uses the device name passed from the application through the Device Control Class to access the XML file though the Java Configuration Loader™ (JCL™) call. This returns the flexible set of parameters required to communicate with the device.

Typical parameters used by the Device Service Class are:

1. DeviceBus – RS232, RS485, USB Port etc.
2. FlowControl – Handshaking (Xon/Xoff)
3. PortName – COM1 etc.
4. BaudRate – 1200, 2400, 4800, 9600 etc.

Standard JavaPOS™ Device Service Classes use the JavaCOMM Package and these parameters allow a flexible connection to the device. It is also possible for a vendor to use the Java Native Interface (JNI) to create a direct connection with limited flexibility. This is the “proprietary” solution seen in the graphic.

After opening the physical device, the Device Service Class is responsible for:

1. Receiving commands and messages from Application Program through the Device Control Class.
2. Adding required escape characters to the data string and sending the completed string to the device.
3. Monitoring the device for responses or hardware exceptions.
4. Monitoring the device for events and data
5. Passing these responses, data and exceptions back to the Application Program.

3.7.10 JavaPOS™ Device Initialization and Finalization

In order that a Java™ based application is able to use a POS device, certain actions need to be done:

- The Java™ application must create an instance of the Device Class which is based in the JavaPOS™ Device Control Layer. For example, if the application wants to communicate with the cash drawer, it must first create an instance of the generic JavaPOS™ Device Class “CashDrawer”.

- The next step is that new instantiated device must be prepared for the events that the application needs to receive. To initiate activity with the Physical Device, the application needs to call the device open method. The open method passes a string parameter “logicalDeviceName”. This parameter specifies a logical device to associate with the Device. The “logicalDeviceName” is stored in the XML hardware properties file and holds all necessary hardware and properties which are needed to operate this device. The open method performs the following steps:
    - It creates and initializes an instance of the proper Service Class for the specified name.
    - It initializes many of the properties, including the descriptions and version numbers of the Device, which are stored in the XML file.

More than one instance of a JavaPOS™ device may have a Physical Device open at the same time.

- Therefore, after the device is opened, the application which needs the device must call the claim method to gain exclusive access to it. Claiming the device ensures that other JavaPOS™ device instances do not interfere with the use of the device. An application can release the device to share it with another JavaPOS™ device instance. For example, at the end of a transaction.
- Next step, before the device can be used, the application needs to set the “DeviceEnabled” property to “true”. This value brings the Physical Device to an operational state, while false disables it. For example, if a Cash Drawer Device is disabled, the Physical Device will be put into its non-operational state (when possible). Whether physically operational or not, any input is discarded until the device is enabled.
- After the application finishes using the Physical Device, it should call the close method. If the “DeviceEnabled” property is true, close disables the device. If the Claimed property is true, close releases the claim on the device. Before exiting, an application should close all open Devices to free device resources in a timely manner.

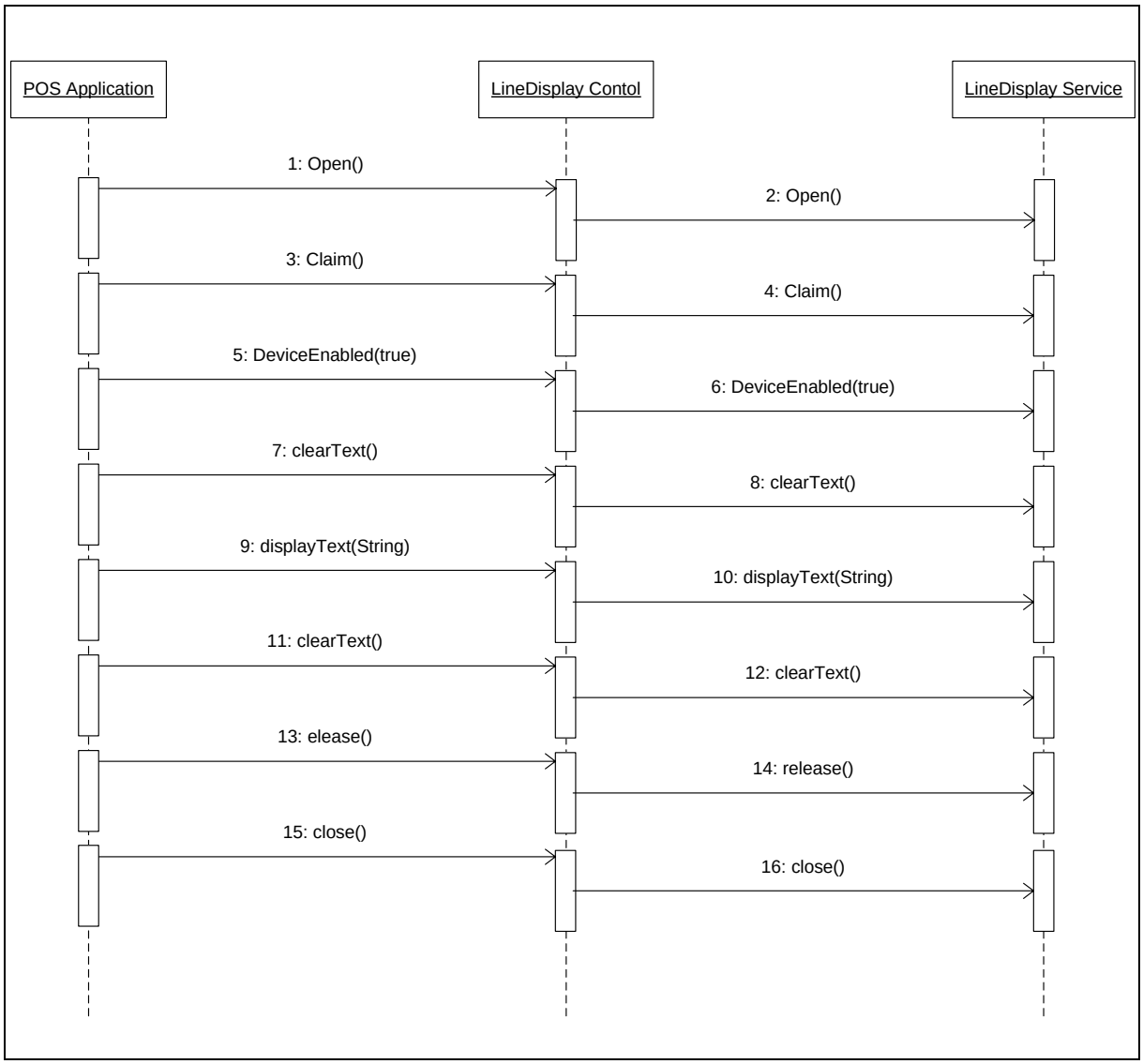

Figure 3-14: Sequence diagram of a line display

In the figure a sequence for a line display is shown:

1. The POS application calls the open method of the line display control, the control then directs this call to the physical device. The physical device, the line display starts operating.
2. The application claims the device; the call is again directed to the physical device.
3. Then the device needs to be enabled.
4. Now the device is ready to receive operating commands, first the display is cleared.
5. Then the application can send a String of characters and these characters will be displayed on the line display.

6. When the application finishes the use of the line display it should clear the display again, so next time the display would not start with the last characters used.
7. After the clearing of the display the line display can be released, so another application can start using it.
8. If the line display is not used any more it should be closed.

3.8 Operating System and System improvements

3.8.1 Overview

As mentioned in previous chapters, only a few operating systems can be used to run this clients' hardware. The advanced and more stable Windows XP™ was chosen to be used. Windows XP™ provides the most stable und convenient environment from the windows family.

The clients' hardware, IBM SurePOS 500™ was originally designed to be run under Windows 98™ or Linux Red Hat 7.1™.

To continue the usage of Windows 98™ is not recommended, because Windows 98™ is not supported by Microsoft any more. Linux Red Hat 7.1™ is also not up to date.

3.8.2 System Driver for Windows XP™

The choice of the operating system is mainly limited by the POS system drivers. Drivers are for the following components needed:

- VGA Graphics driver
- 10/100 Mbit Ethernet LAN driver
- The Touch Screen driver is needed, because the main control is this human input device (HID) and this is done only with the use of the touch screen. This driver is just available for a few operating systems: Windows 98™, Windows ME™, Windows 2000™, Windows XP™ and Linux Red Hat 7.4™. This limits the choice of the operating system.
- Chip set driver which include the serial, parallel and USB port drivers and the peripheral component interconnect (PCI), and integrated drive electronics (IDE) drivers

With the use of the different operating systems, not all drivers need to be installed.
For Windows 98™, Windows ME™ and Linux Red Hat 7.1™ all above mentioned drivers need to be installed.
Window 2000™ and Windows XP™ have almost all needed drivers included in the installation process. Only the touch screen driver needs to be added after the installation.
All other drivers are already implemented in the XP installation.

3.8.3 Choice of the Operating System

Below a table is presented which shows pros and cons for each of the possible operating system which could be used.
The decision which operating system is used is based on this information.
From the clients' point of view, the most important requirements on the operating system are:

1. Stability and data security
2. Superior File System (NTFS)
3. Future Support of Operating System
4. Boot time
5. Price and resource requirements

in that order.

Since Windows XP is the best option in all requirements except for the least important one, the choice obviously falls for Windows XP.

The System Requirements / pros / contra of the Operating System

	System Requirements	Pros	Contra
Linux Red Hat 7.1™	▪ 486DX (min. 66 MHz) CPU ▪ HID (Keyboard)	▪ Very Stable ▪ GNU License	▪ Long boot time ▪ Difficult to use ▪ SWT usable with restriction
Windows 98™	▪ 486DX (min. 66 MHz) CPU ▪ 16 MB RAM ▪ 196 MB free HDD space ▪ VGA Display Resolution ▪ HID (Mouse and Keyboard)	▪ Fast Boot Time ▪ Low system resources	▪ Not supported anymore ▪ Unstable ▪ Only FAT 32 support
Windows ME™	▪ Intel Pentium150 MHz or compatible CPUs ▪ 32 MB RAM ▪ 320 MB free HDD space ▪ VGA Display Resolution ▪ HID (Mouse and Keyboard)	▪ Fast Boot Time ▪ Low system resources	▪ Not supported anymore ▪ Unstable ▪ Only FAT 32 support
Windows 2000™	▪ Pentium or compatible CPU with 133 MHz ▪ 32 MB RAM ▪ 650 MB free HDD space ▪ VGA Display Resolution ▪ HID (Mouse and Keyboard)	▪ Stable ▪ NTFS-support	▪ Expensive, additional RAM is needed
Windows XP™	▪ Pentium 233-MHz-Prozessor or compatible CPU ▪ 16 MB RAM ▪ 1,5 GB free HDD space ▪ SVGA Display Resolution ▪ HID (Mouse Keyboard)	▪ Very Secure ▪ Fast boot Time compared to Windows 2000™ ▪ Stable ▪ „Cleartype Fonts“ better readablity of TFT-Displays ▪ NTFS-support ▪ Cheaper than Windows 2000	▪ Expensive, additional RAM is needed ▪ Faster HDD is needed

Figure 3-15: The system requirements

The difference between each operating system:

- Linux Red Hat 7.1™ needs little hardware resources. A central processing unit (CPU) with 66 MHz is sufficient. Linux is a very stable operating system especially when compared to Windows 98™ which was originally installed from IBM as standard. It also runs under the GNU (general public license) License, which is in relation to Microsoft product very cost inexpensive. The downsides to Linux are:
  1. It has a long system boot time in comparison to Window 98.
  2. For those which are familiar to the windows products and are only end user, it is difficult to operate Linux.
  3. In this project eclipse Standard Widget Toolbox (SWT) is used. Standard Widget Toolbox (SWT)is only usable with certain restrictions under the Linux environment.
- Windows 98™ also needs few hardware resources; a 66 MHz CPU with at least 16 MB random accessible memory (RAM), 196 MB free space on the hard disk drive (HDD) is adequate. The benefit of Windows 98™is that it has the fastest boot time from all these operating systems which were tested on the clients' hardware. The disadvantages are:
  1. The operating system behaves very unstable when the project is running.
  2. It also uses the File Allocation Table (FAT) 32 as file system, which is not very secure.
- Windows ME™ needs more hardware resources as windows 98™; a 150 MHz central processing unit (CPU), 16 MB random accessible memory (RAM) and 320 MB free space on the hard disc drive. The only reason why windows ME® can be used is that the system drivers from Windows 98™ are also usable for Windows ME™. From the official IBM side, Windows ME™ is not supported. While testing this system no real differences to Windows 98™ where identified which are relevant to this project. ME also only supports File Allocation Table (FAT) 32 as the file system.
- Windows 2000™ needs at least a 133 MHz central processing unit (CPU), 32 MB random accessible memory (RAM) and 650 MB free hard disk space (HDD). This system behaved very stable while testing the project environment on it. The configuration of the system is quite simple compared to the others. It also uses New Technology File System (NTFS) which is much more stable and secure

against data loss than File Allocation Table (FAT) 32. But the license of Windows 2000™ is the most expensive of all these operating system.

- Window XP™ is the most advanced operation system which can be used on the IBM SurePOS 500™. In order to use operating system the hardware needs to be modified, in terms of the size of the random accessible memory (RAM) and the speed of the hard disk drive (HDD). With some operating system fine tuning, this system has the shortest boot up time. The license is less expensive and will be supported for a long time. Windows XP™ supports the New Technology File System (NTFS) file system. The project environment runs very stable and smoothly under this system. Also the standard widget toolbox (SWT) representation looks more modern in comparison to the other operating systems.

3.8.4 Hardware modification to run Windows XP™

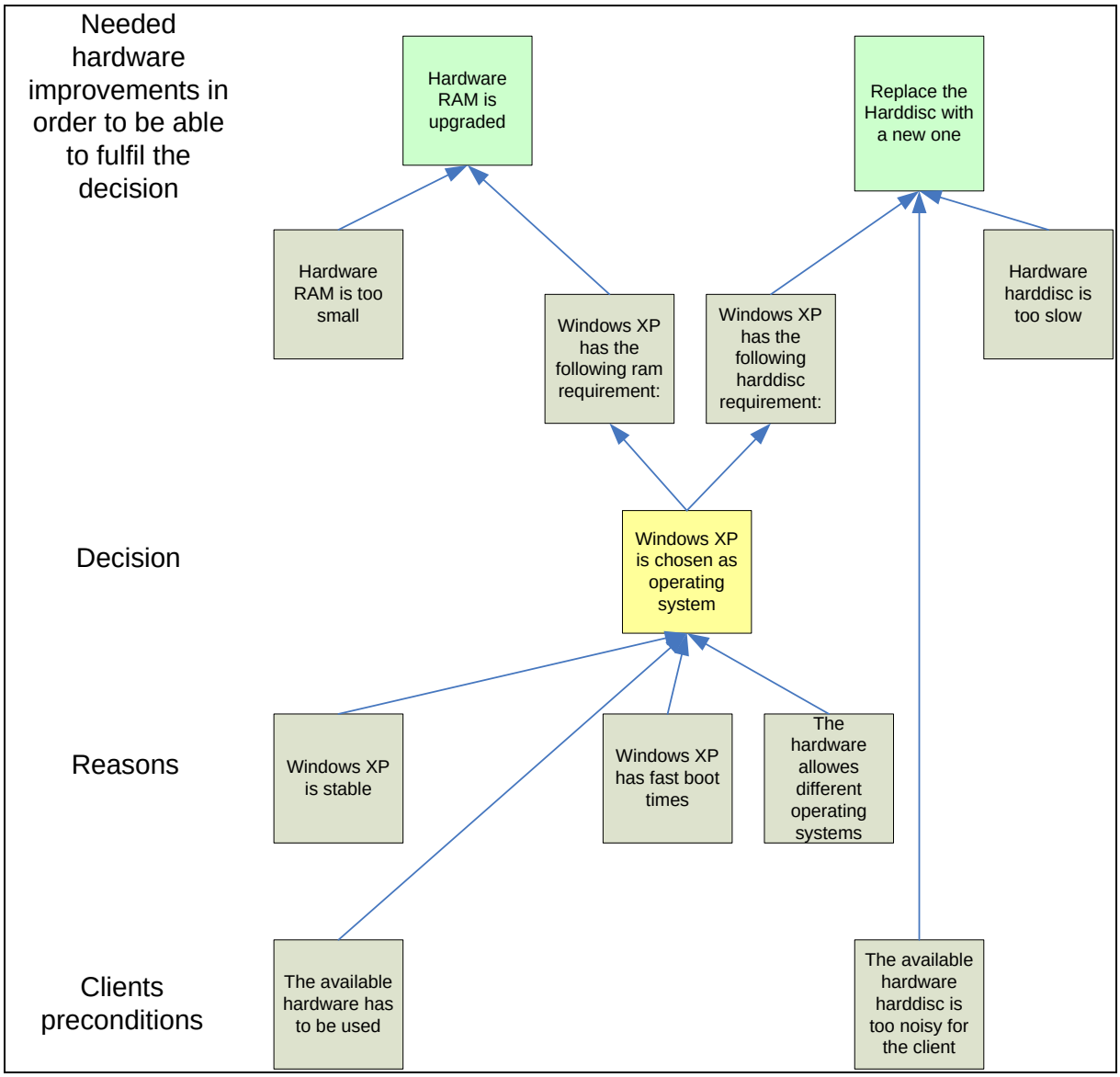

Figure 3-16: Decision logic, operating system and hardware

In order to install a smoothly running Windows XP™ on the IBM SurePOS 500™, certain modifications are necessary.

This system is designed to be run under operating systems which need less hardware resources than Windows XP™ requires. Specific components need to be updated and replaced with faster or bigger ones to accommodate Windows XP.

Special hardware and software fine tuning needs also to be done to increase the performance and the comfortable usage of the IBM SurePos 500™ under the operating system Windows XP™.

Hardware modifications in detail:

- The existing 64 MB RAM is not sufficient for Windows XP™.
- The old HDD is too slow.
- The BIOS needs to be renewed via a firmware update.

To upgrade the random accessible memory (RAM) of the system the complete POS body has to be disassembled; the cashier system is not designed for the frequent change of internal hardware components.
For the RAM upgrade, a special RAM type Single Data Rate with a transfer speed of 100 MHz (SDR PC100) standard is needed. To decrease the boot time and to improve the development speed of the whole software project, the decision to use 512 MB random accessible memory (RAM) is taken.
The speed of the hard disk drive needs to be increased; this is solved by replacing the old disk with a newer one. So that the clients' hardware recognizes the newer faster hard disk, the BIOS need to be updated.
The Basic Input / Output System (BIOS) update needs to be performed, so that the new hard disk drive (HDD) is recognized and that it is possible to boot from different media instead of only from floppy and a special CD- ROM drives. With this Basic Input / Output System (BIOS) update it is now possible to boot from USB stick, USB CD ROM, USB HDD and new HDDs are supported like 2,5" notebook disks.
With this new updated configuration, the POS system functions much faster as measured.

After all these steps have been performed the IBM SurePOS 500™ is now able to run advanced systems like Windows XP™ and still has some additional reserves. The development can be done with a Java™ Eclipse 3.2 environment directly on the POS system with good response times

3.8.5 Modification to Reduce the Operating Noise

Hardware modifications:

- 3,5” hard disk drive is to noisy because of it’s vibrations.
- The noisy CPU fan needs to be modified, to reduce the operating noise.

In the same work step of upgrading the random accessible memory (RAM), the central processing unit (CPU) fan is analyzed. After analyzing the central processing unit (CPU) fan, more information is gathered.

1. The central processing unit (CPU) fan has a temperature sensor to reduce the speed of the fan.
2. This sensor is not used by the main board.
3. The POS does not need the full CPU fan power to cool down the system.
4. The voltage of the central processing unit (CPU) fan can be dropped from 12 volts to 5 volts.

With this new information it is possible to reduce the CPU fan noise.

The 3,5” hard disk drive (HDD) is replaced by a new 2,5” notebook hard disk drive (HDD). The advantage of this solution is that it produces less vibration and operation noise due to the smaller size.
To make this smaller hard disk drive (HDD) fit into the old chassis, it is necessary to use a special 3,5” to 2,5” IDE adapter, and two rails to improve the width of the smaller 2,5” hard disk drive (HDD).

3.9 Hardware Description

3.9.1 Overview

The clients’ hardware is the SurePOS™ 500 from the vendor IBM®, Type 4840-541. The IBM SurePOS™ 500 series has an innovative touch screen interface that is designed to enable rapid, convenient transactions, while being robust enough to withstand retail environments, in this case a flower shop.

3.9.2 Detailed SurePOS 500™ components

This figure shows the complete system with all its components.

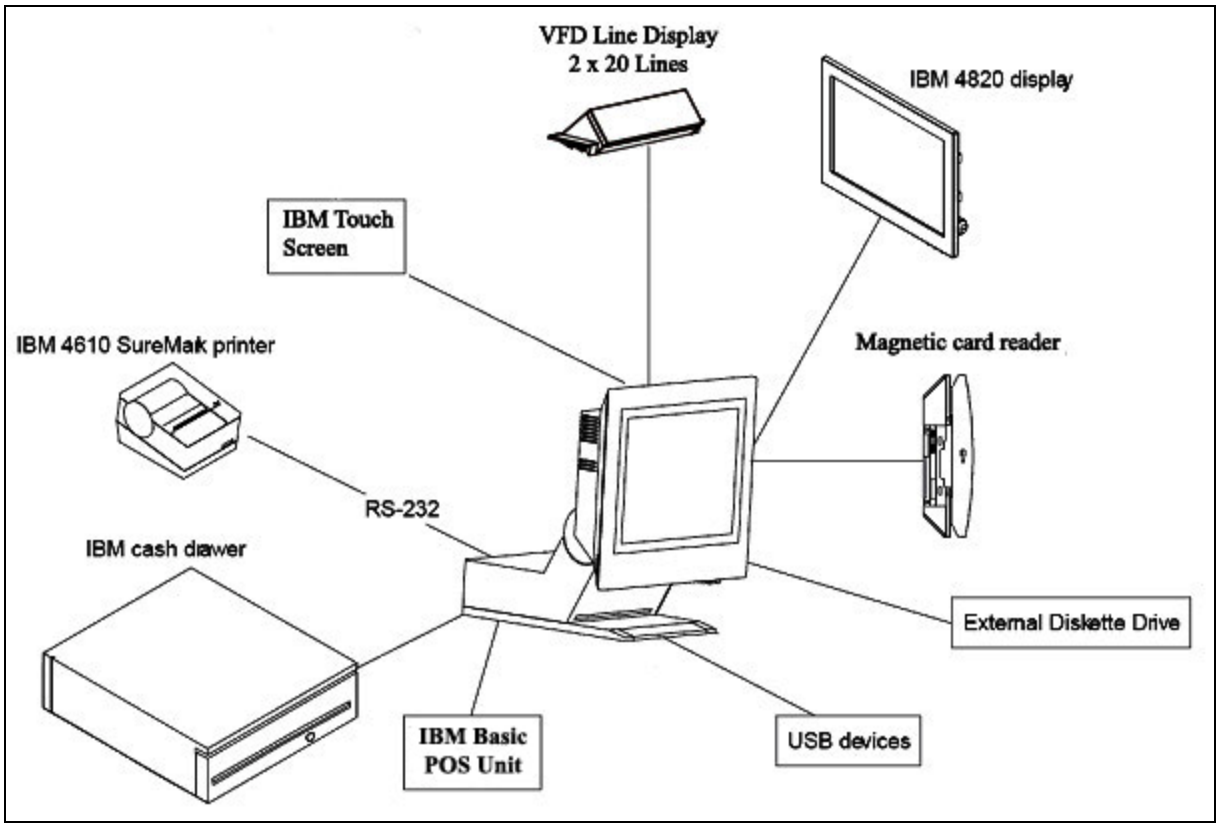

Figure 3-17: Complete POS system with all components

3.9.3 IBM Basis POS Unit

This unit is a 342 mm x 385 mm x 323 mm small computer system which also holds a display and a corresponding touch screen. It contains all necessary personal computer devices like main board, central processing unit, random accessible memory, system fan, hard disk drive and two universal serial bus, two serial communication port, an Ethernet port and a parallel port. The basic configuration for this System includes a micro **A**dvanced Technology Extended (ATX) main board, a Central Processing Unit (CPU) from the vendor Advanced Micro Devices Inc. **(**AMD®) with a clock speed of 400 MHz named AMD-K6™ 3D, which is cooled by a special flatly designed heat pipe central processing unit (CPU) cooler, 64 MBytes random accessible memory (RAM), 20 gigabytes (GB) Maxtor® Fireball LCT15 UltraATA-100 3,5” hard disk drive (HDD).

3.9.4 IBM Cash Drawer

Figure 3-18: The cash drawer

The IBM cash drawer is a 440 mm x 480 mm x 140 mm cash box with a drawer inside of it. This cash drawer is created to allow the computer unit to be set on top of it. This is a robust metal drawer were the money is stored, which can be opened via a software command or manually with a key.

3.9.5 IBM 4610 SureMark Printer

Figure 3-19: The POS printer

A thermo printer is a must, so that the cashier does not have to worry about ink cartridges, toner or ribbons, but only has to change the paper when a roll is used up.

The SureMark 4610™ printer supports the following functions:

1. Barcodes, company logo, monochrome schematics and TrueType-fonts provides possibilities for individual design of receipts for a selective marketing.
2. Sensor for end of receipt paper.
3. Automatic Receipt cutter.
4. Tear-off-edge.

The SureMark 4610™ printer uses eighty millimeters wide thermo receipt paper and it is a fast and reliable thermo printer. It offers the possibility to store a company logo, bar codes, graphics, or advertising texts in the internal flash storage. This ensures the printing of the receipt without any additional time delay. The communication connection with this device is established with a RS-232 serial communication port. The selection of optimal operating system is limited by the availability of driver software for this printer.

3.9.6 Magnetic Card™ Reader

The magnetic card reader (MSR) is attached to the side of the touch screen. However it will not be used in this project.

3.9.7 IBM 4820 Display

The display of this system is a 12” thin film transistor-liquid crystal display (TFT-LCD) touch screen with a super video graphics array (SVGA) on a resolution of 800 x 600 pixels.

3.9.8 IBM Touch screen

Is a touch sensitive screen on top the normal display. With this it is used as a Human Interface Device (HID). This has the advantage than no extra keyboard and mouse are needed.

3.9.9 VFD Dual Line Display 2 x 20 Lines

Figure 3-20: Dual Line Display

The dual line display sits on top of the back of the computer unit. This line display is able to display two rows of twenty characters from a predefined character set. It is a vacuum fluorescent display (VFD) and the letters have a bright yellow color. Unlike liquid crystal displays (LCD), a VFD emits a very bright light with clear contrast and can easily support display elements of various colors.

3.9.10 Universal serial Bus (USB) Devices

With this communication port external devices can be attached. For example it was used in this project to attach an external CD-ROM drive.

3.9.11 External Diskette Drive

With this device it was ensured that a removable storage can be used. At the time that this Point of Sale (POS) system was developed, USB storage sticks were not available. It is also used to update the Basic Input/Output System (BIOS).

3.9.12 Power requirements and consumption

The power supply supports 100 to 240 Volt and 50 or 60 Hertz.

The power consumption measured at 100 volt is as follows:

- System turned off: 3.5 watts
- Standby modus: 23 watts
- System turned on but idle (no processes active in CPU): 45 watts
- System turned on maximum load: 115 watts

3.10 Requirement summary

A software project requires that the client specify his requirements and that the software engineer fulfil them.

The cashier software has to comply with the following functional requirements

Keyword	Function description
General	- A new software with the given requirements has to be created - Software needs to be expandable - The available IBM cashier hardware has to be reused
GUI	- easily readable, big buttons - input products without changing code - show date and time - nine product groups - numeric input field, clear button, delete button - payment by cash or EC-Card - net / gross amount displayed on screen - SWT is chosen as toolkit in order to create the GUI
Export of data	- External storage of accounting data on USB-stick
Accounting, database storage	- Each transaction stored (Whenever something is sold) - Accounting summary at the end of a day - salary accounting of employees - Storage on a database - MySQL as database software - Central storage of the database software
Support of external devices	- Customer receipt printing with logo - Open cash drawer - Customer information on the line display
Operating system	- Stable and robust operating system - Java™ and JavaPOS™
Hardware	- IBM SurePos 500™

Figure 3-21: Requirement Summary

4 Design

4.1 Overview

In the design phase of a project the assembled requirements have to be transferred into a programming language.

4.2 Class design

The following section describes how the software has to work. The class candidates are marked in italics. These class candidates are implemented as classes in the later realization phase.

The task is to develop a software tool, which is suited for a cashier system. There has to be a *graphical user interface*. This cashier system has to have different buttons for the *products* sold. It has to be possible to insert these products without changing the source-code. The idea is to generate a text file that has all product names as well as their tax-values stored. By this process, the client can easily adjust the product's names and tax-values.
Another service the program is required to offer is to show the actual date, as well as the actual *time*. The proposal is to implement this as a thread.
After an item is sold, the information has to be stored in a *database*. The *sold articles* are stored in a table.

After each day the client needs to have the possibility to get an overview of the income. The money that was in the drawer in the morning needs be compared with the amount of money that is still there at the end of the day in order to discourage theft by workers.
This information is stored in a table called *Accounting Cashier*.

The employees are paid from the money that is in the cashier's drawer. Each worker has its own personal ID, so that with this ID the amount that was paid can exactly be tabulated. Therefore a third table called *Worker* is needed.

There has to be the possibility to store all information that is saved in the database on an external portable device in order to evaluate this data externally.

As earlier pointed out, the cashier uses IBM's JavaPOS™ system. This system is needed in order to communicate with the IBM specific peripherals. There must be the possibility to open the *cash drawer*, to print *receipts*, as well as to display the bought items on a *line display*.

The software creation has to be based on today's valid specifications for object oriented software engineering. This includes a possible expandability of the software product.
A very important fact is that the GUI has to be separated from external objects as well as from many operations. It is important to acquire an overview of such a project. The idea is to separate the GUI from other objects or methods. A separate class contains all methods that are needed by the GUI's objects. Even many action Listeners can make such a GUI's source code extremely unreadable; they are also isolated as a new class.
A *controller* is used that creates all objects and plays the role of a kind of communication centre.

The detailed description of the created classes

- Controller

The Controller class is that in which the generation of objects of all other classes as well as the program flow is controlled. This class contains an operation to initialize the Display. A linked List is created that contains all product objects that are created. Methods for getting the needed objects are listed here.

- GUI

The GUI class generates the needed SWT-elements, e.g. buttons, tables and labels. All action Listeners as well as GUI's relevant methods are removed from this class and moved to separate classes. This is done in order to retain visibility of this huge class.

- GUIMethodClass

Contains all methods that are needed for the GUI operation. This class holds methods to count inputs, save intermediate results and format Strings e.g. for printing.

- Time

A Thread is used in order to update the time that has to be displayed on the cashier's screen. This thread is implemented into its own class.

- ActionListenerClass

All action Listeners are removed from the GUI and moved to this class. The current GUI's object has to be passed to this class in order to have access to all created SWT-elements. For each button a special action Listener is needed.

- Product

A product consists of a product's name as well as its tax value. These values are fixed and only changeable by editing the text file that holds all product groups.

- ProductItem

A Product Item consists of a product (name and tax value) combined with the price and the number (how many items of this product are sold); the price as well as the number are input by hand.
A Product Item object then contains all necessary information about a product (name, tax, single price as well as its quantity)

- CashDrawer

Methods are provided to communicate with the cashier system's cash drawer. These methods make use of JavaPOS™. A method is needed that opens the drawer.

- LineDisplay

The line display is controlled by its class. Three methods are needed; one to initialize the display, one to clear its content and one to shows the desired content to the shops customers.

- Printer

This class allocates methods to print out the customer's receipt. A method initializes the printer; another method is required to initialized this printer. The last method's task is to specify the needed JavaPOS™ methods to print the content onto paper.

- ReadFromTextFile

As earlier pointed out, a product consists of a name and tax values (7% or 19% actual tax values in Germany). These values need to be easily adjustable. Therefore this class contains a method that reads this information from a file and creates product objects out of it. The input products are saved onto a Linked List; from this list the button names are set. This class also contains methods to get isolated information about a product, e.g. getName().

- Database

A database class contains the initialization of the database connection as well as other reusable methods. In order to make the reusable methods accessible by other classes, inheritance is used. The Database class inherits to the child classes Accounting, SoldArticle and Worker.

- Accouting

This class contains all specific methods and attributes to query and write into the database; in this case the accounting table. There is also a method to transfer the database content to the USB-stick.

- Worker

The Worker class includes methods to query and write to the database table "tableWorker".

- SoldArticle

The SoldArticle class includes methods to query and write to the database table "tableSoldArticle".

4.3 Class diagram

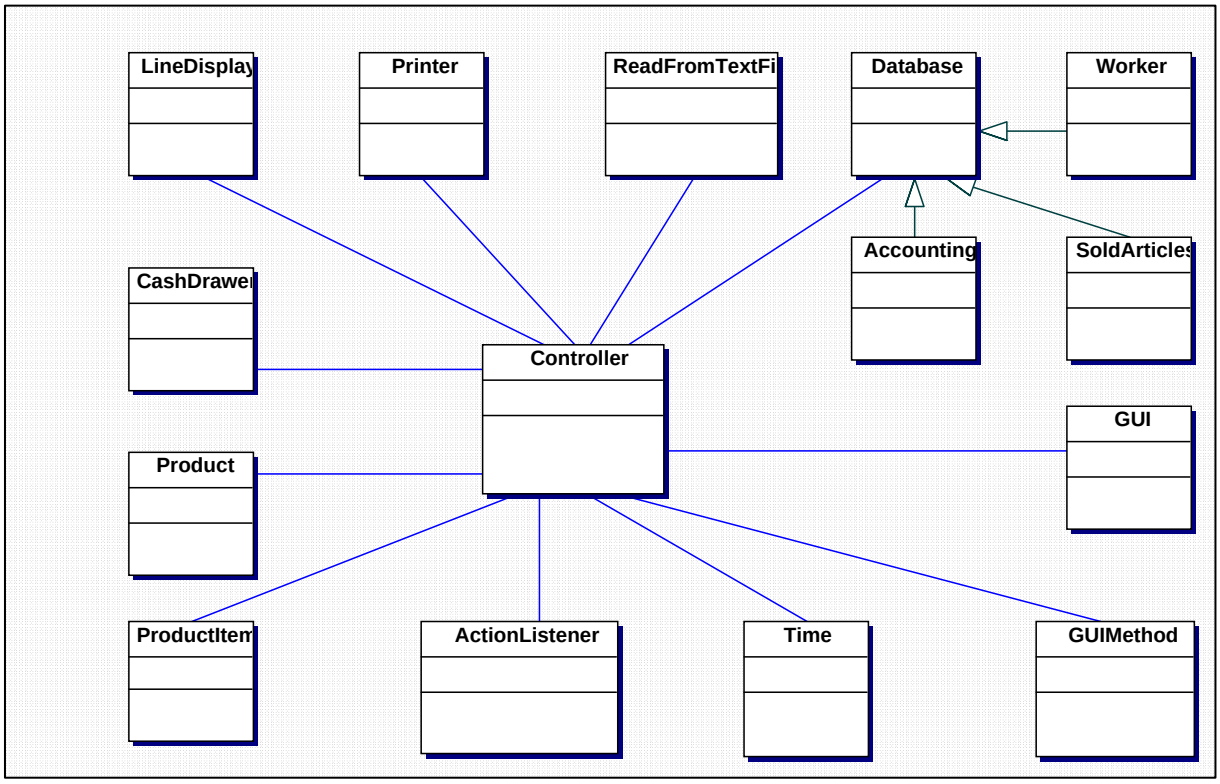

Figure 4-1: Class Diagram

Figure 4.3.1 shows the actual class diagram. The Controller class holds the main method of this program. The main method is the first method that is executed when the program is started.

4.4 Database class model

Accounting data are stored within a database, where three main transactions are accomplished, the storage of the sold articles, an accounting summary as well as the salaries that were paid to the employees. Three separate database tables are created. The use of three tables is very important for later database queries from the program. The information that belongs together need to be stored within one table.

The next figure gives an overview of the tables created and used.

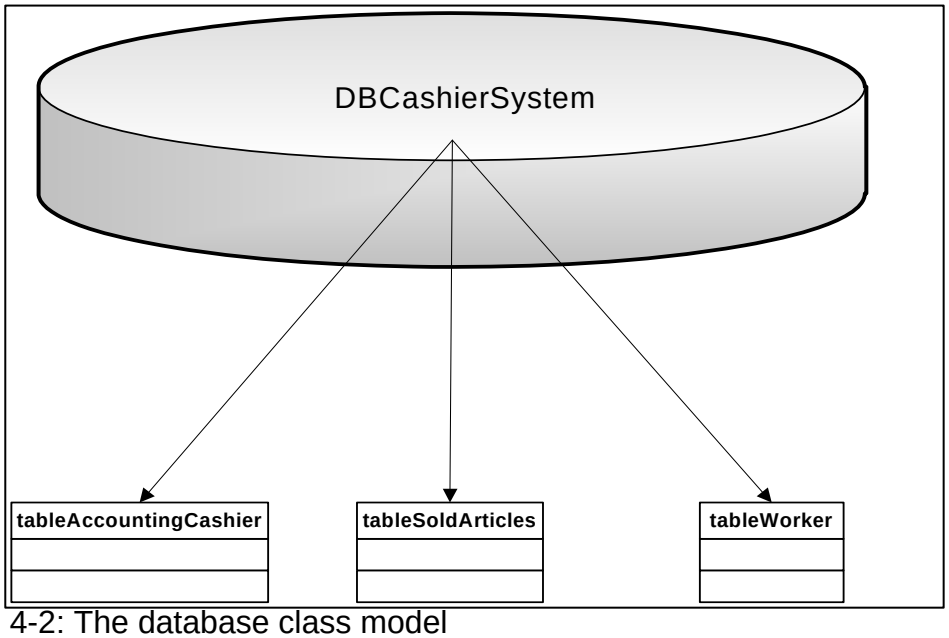

Figure 4-2: The database class model

Database tables

The sold articles, *tableSoldArticles*

This table stores all information that is gathered for each sold item.

- soldArticleID
  Stores an ID for each transaction; this is done to enable the creation of a suitable Primary Key
- date
  In order to specify the date when a transaction was executed
- amount
  The number of products that are sold within one transaction
- name
  The identification of the product
- price
  The unit price of the product
- Tax
  The taxation of the product
- Net amount
  The net amount of the product * amount
- Gross amount
  The gross amount of the product * amount
- Method of payment
  It can either be paid by cash or EC-card

The accounting of the cashier, *tableAccountingCashier*

This table stores all information that deals with the accounting of the shop. The income, outcome, as well as the money in the drawer have to be stored and evaluated. For this purpose the design of the tables looks like the following

- accountingID
  Stores an ID for each accounting transaction; this is done to be able to create a suitable Primary Key
- date
  In order to specify the date when a transaction was executed
- income
  The amount of money that was input into the drawer
- outcome
  The amount of money that was taken out of the drawer (for paying salaries etc.)
- moneyInDrawer
  The amount of money in the drawer that was counted by the cashier at the end of each day
- difference
  The difference between money that was in the drawer and the money that was taken out of it. This has to be done in order to make sure, that no miscalculations occur or that employees steal money.

The last table is needed to store data is the table for the worker's salaries.
Each worker has his/her specific id. The shop owner pays the worker's salary directly from the cashiers' drawer. Therefore this table needs to store the id, the date and the time as well as the amount that was paid.

tableWorkers

- date
  In order to specify the date when a transaction was made
- Id
  Each worker has his/her specific ID from which he/she is uniquely identifiable
  This Id is a candidate for a sound Primary Key
- amount
  The amount of salary that was paid out

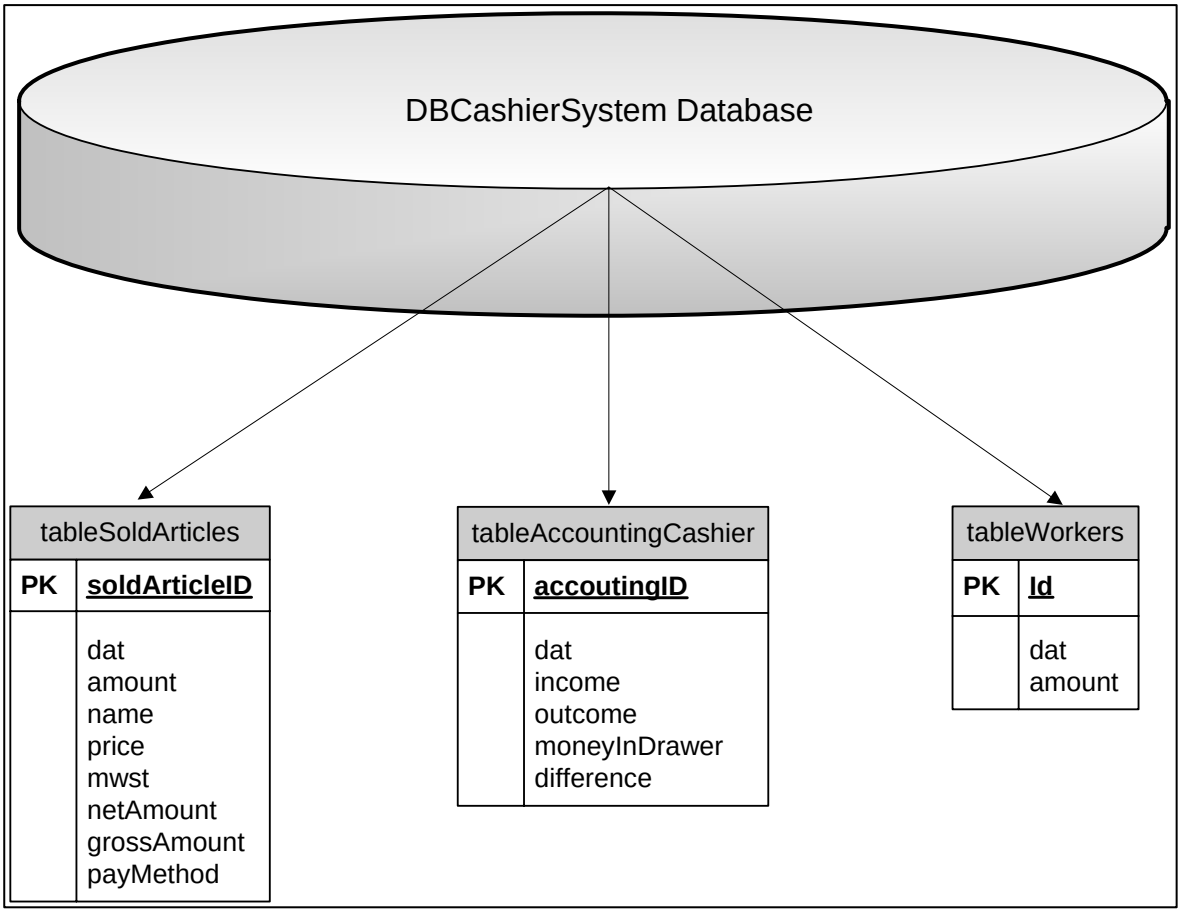

Figure 4-3: The database model

4.4.1 Connection to database

Loading the JDBC driver is the first step in connecting to a database.

Since the connection to the database is also a method that can be reused, it is part of the inherited Database class.

When the driver class is loaded, an instance of itself is created and registered with the DriverManager that will be used while initializing a connection.

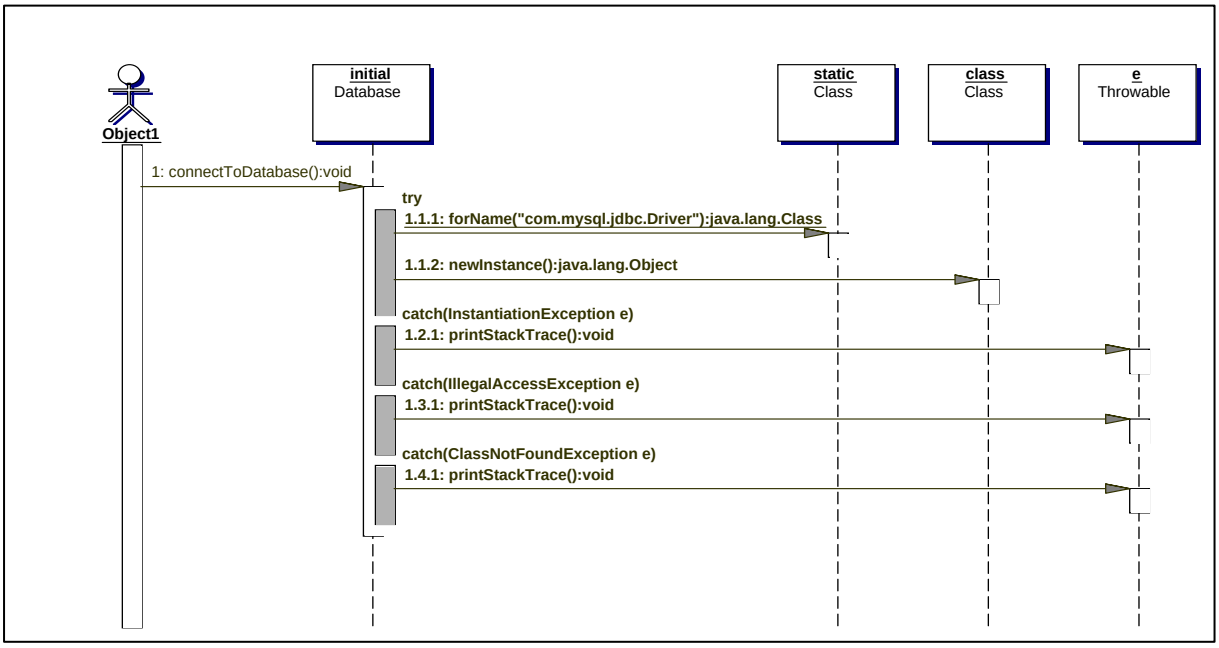

Figure 4-4: Sequence JDBC driver initialization

4.4.2 Initialization of the database connection

The method initConnection() of the class Database has the task of initializing a connection and returning an sql-statement object (A SQL-Statement is an instruction that is executed; the return value is a result Set. Such a SQL-Statement poses queries directly to the database. The returned valued from this query are stored in a result Set).

The method initConnection() creates a JDBC connection with help of the Driver Manager.`"jdbc:mysql://localhost/DBCashierSystem"` means that the “JDBC” driver for MySQL loads a database that is locally stored.

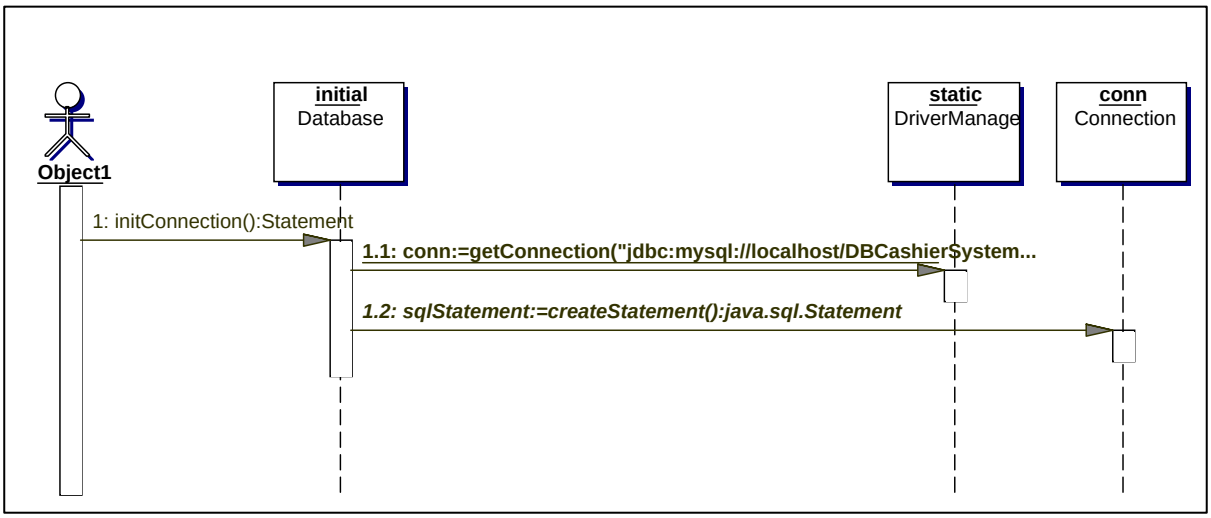

Figure 4-5: Sequence initialize database connection

The databases name is DBCashierSystem.

4.4.3 Writing to the database

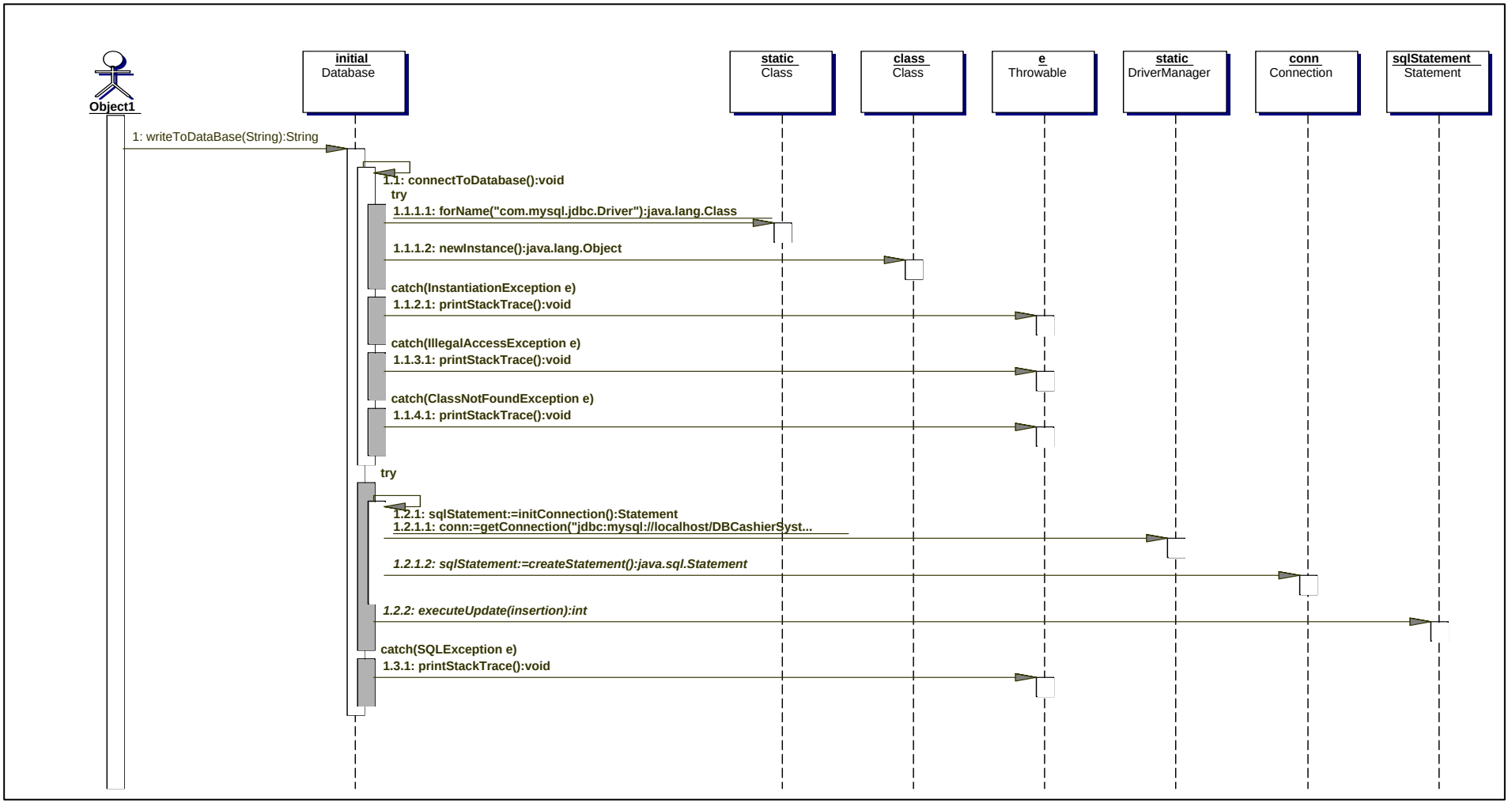

Figure 4-6: Sequence diagram, writing to database

4.5 Interface design

In this project three classes are available to interface the hardware.

4.5.1 CashDrawer

When the instance of the Use Cash Drawer class is created, the constructer uses the JavaPOS™ Device Control classes to open the cash drawer device, and to claim and enable it.
After that, the method OpenDrawer is used to open the cash drawer when required.

Since the POS system is a dedicated system, there is no need to close the connection to the cash drawer, since no conflicts with other threads can occur.

4.5.2 Dual Line Display

When the instance of the Use Line Display class is created, the constructer uses the JavaPOS™ Device Control classes to open the line display device, and to claim and enable it.
One method sends a string of 40 characters to the line display. This is then displayed in two rows of 20 characters per row.
A second method clears the display.

4.5.3 Receipt Printer

When the instance of the Use Printer class is created, the constructer uses the JavaPOS™ Device Control classes to open the printer device, and to claim and enable it. Next the constructor stores a BitMap in the the Printer flash storage which contains the company logo. This takes time once, but is very fast when printing the receipts.

The method PrintText receives a formatted string from the application.
First this method prints the logo from the flash storage. Then it sends the string to be printed. This string already includes line feeds and other control sequences to format the print. Finally some empty lines are printed and the paper is cut.

5 Realization

5.1 Chapter overview

This chapter deals with the realization of the software tool. Realization in this manner means the transfer of all designed classes, methods and variables into source code. Testing and quality insurance are also very important parts of this development step.

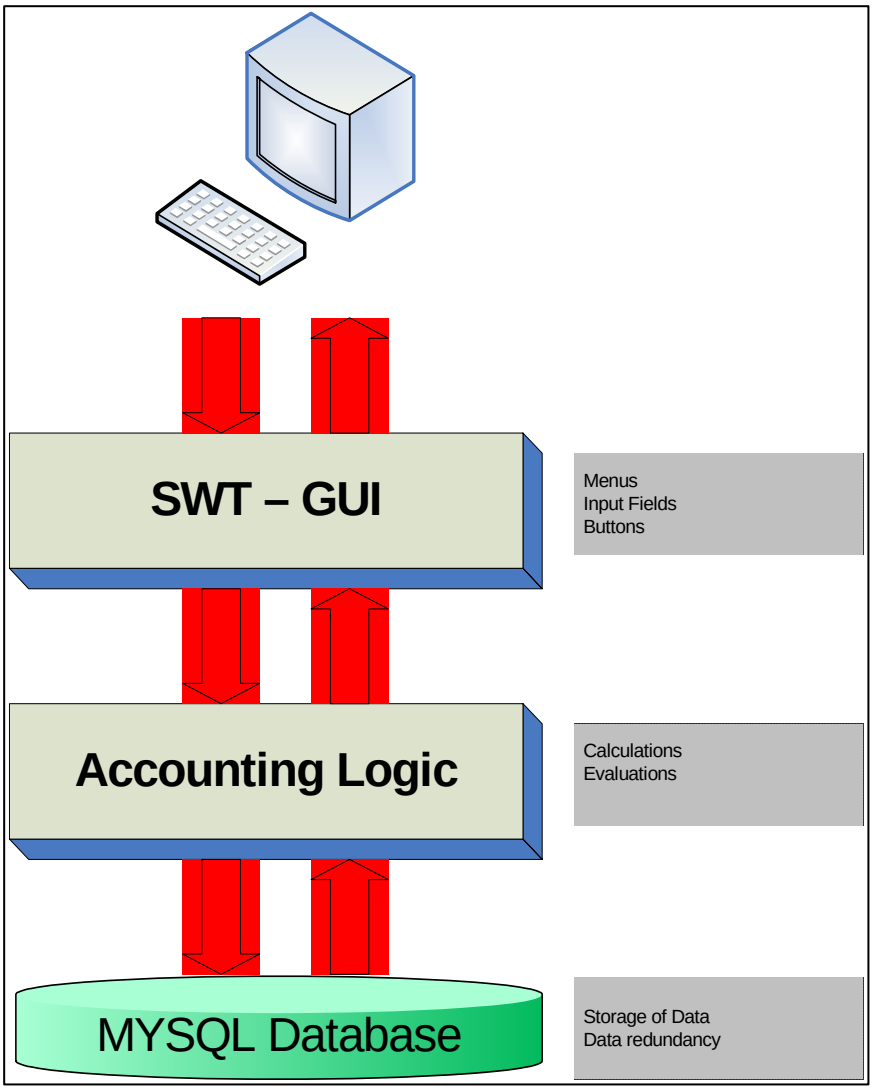

Figure 5-1: Architecture overview

This figure gives an overview of the architecture and software parts used.

5.2 Database realization

As previously described, the database system that used is MySQL. The advantages and disadvantages have already been discussed.

The following example will show the creation of one of the tables.

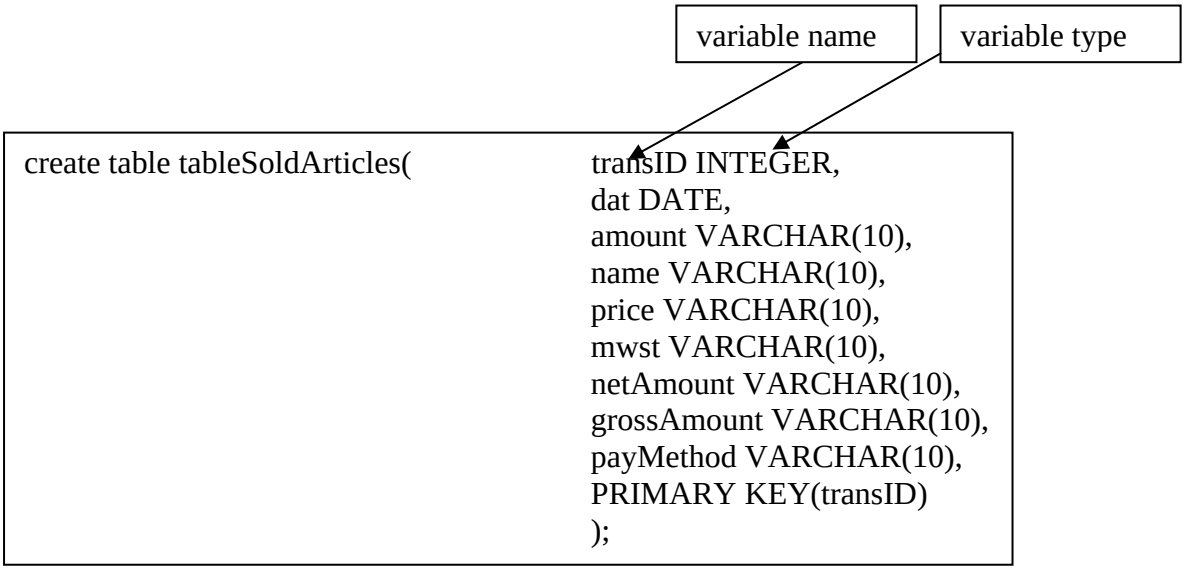

Figure 5-2: The creation of a database table

The create table statement create the table with the given variable names and types.

5.3 Software realization

Software realization in the context of this project includes the realization of software parts that are needed to keep the program flow running. This paragraph describes classes and methods that do not directly refer to the environment of the cashier system. The environment is described as the operation system, the database integration as well as the graphical user interface.

This paragraph will give readers an overview of how the design requirements are implemented into Java™ code. The most important program activities will be explained in detail.

5.3.1 Program execution

The first step in explaining the software realization is to clarify what happens when the program is executed.

The controller-class generates objects of all classes. A linked List[8] is created that later stores all items that are input into the shop customer's basket. The method startup() initializes the display and starts the shell. An object of class timeThread is created that starts a Thread that is responsible for updating the displayed time.
The creation of the object ProductsReadFromFile causes the initialization of a linked List in this class. A method is called (readProductsFromFile()), that reads the product groups from a text-file. For each product an object is generated and stored in the linked List.
The text-file is stored on the harddisc and its content has the following form:

```
Blumen    ,7%
```

"," as well as the "%" sign serve as separations for reading the content of this file.
The class ActionListenerClass initializes all action Listeners. The Listeners that refer to product buttons call a method that is contained within the Controller class to get the linked List that was created in ProductsReadFromFile class.
After the software is started, it is ready to be used.

5.3.2 Sales process

The main use of this system is the selling of products. The classes and methods involved will be explained in this paragraph.
When a product group button is pushed, the matching action Listener in the class ActionListenerClass is executed. This action Listener calls a controller's method to return the product object. With this object it is possible to receive the product's name and its tax value. The name of the product is shown in the GUI's information field.
The current object of the class GUI, is passed to the classes ActionListenerClass and ActionClass.
The textField "textMenge" is set to editable.

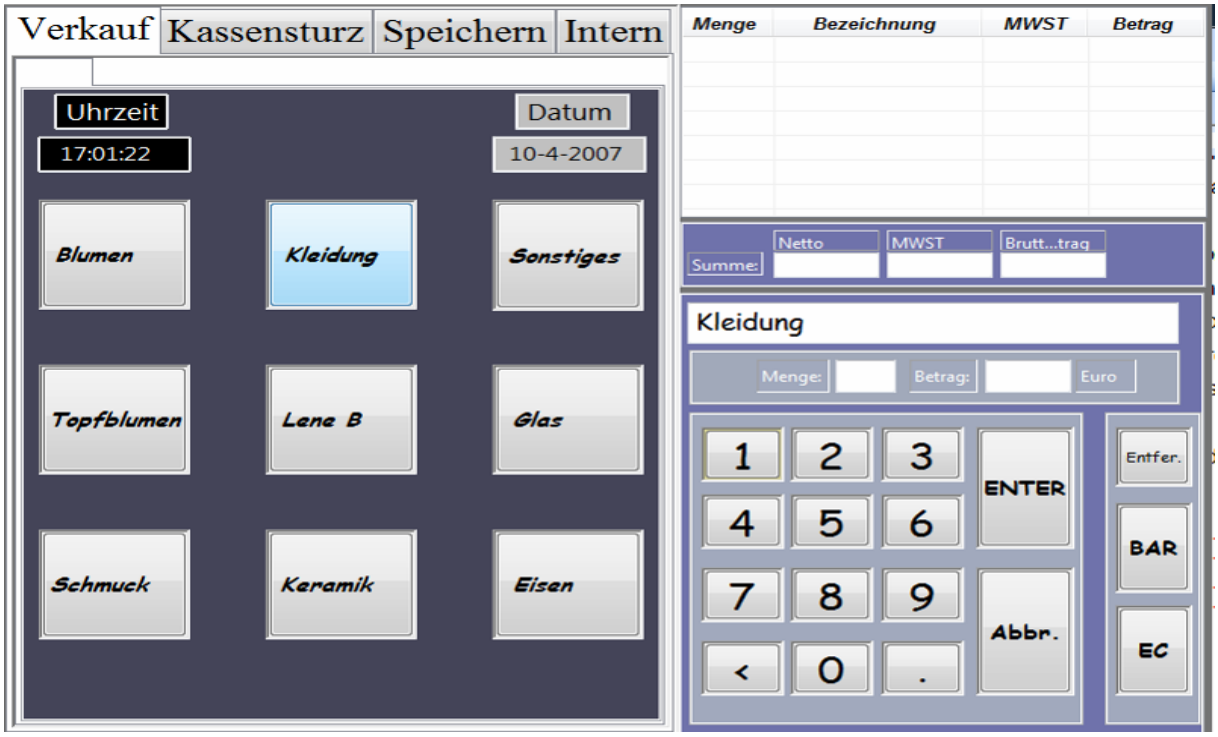

Figure 5-3: Choosing a product

This is done in order to specify the textField where its input is set; the “Enter”-button has to know which program operation is active.

The number of products can now be input. This is done by pressing the number of the numerical field. Each number has it’s own actionListener that calls a method of class ActionClass, setInput(with the appropriate numerical value).

This method evaluates and formats the input; e.g. the number cannot be “0”. It is checked to which textField the input is set; this is done by checking which textField is set “editable”. There is also the possibility to delete the last input digit by pressing “<”. If this button is pressed, the method setInput() creates a substring from the 0^{th} digit until the (last digit – 1).

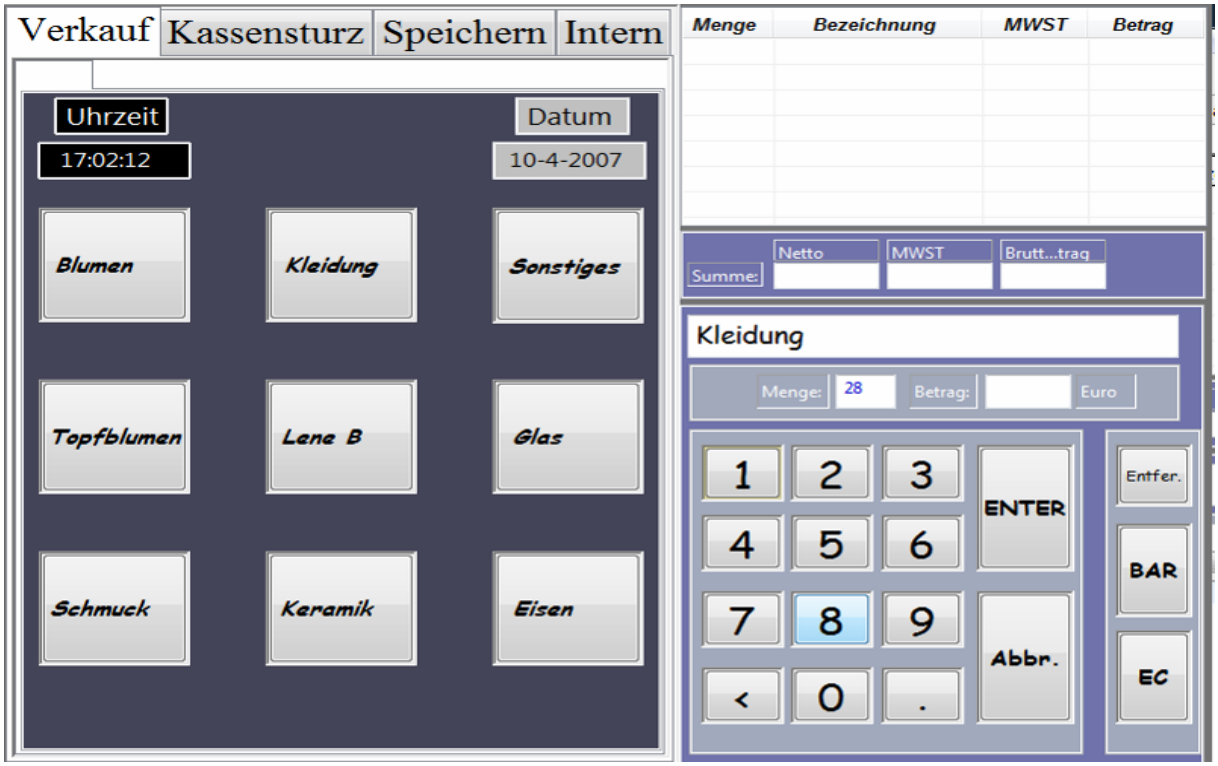

Figure 5-4: Input the amount of selected products

When the input is finished, the user presses the “Enter”-Button. This button has the task of confirming the user’s inputs. The actionListener[8] for this button is also contained in the ActionListener class.

The method of processing is that this actionListener first checks which textField is visible. The input for the number of products textField “textMenge” is already finished, so that this field can be set to “non visible”; the price textField, “textBetrag” is set to “editable”.

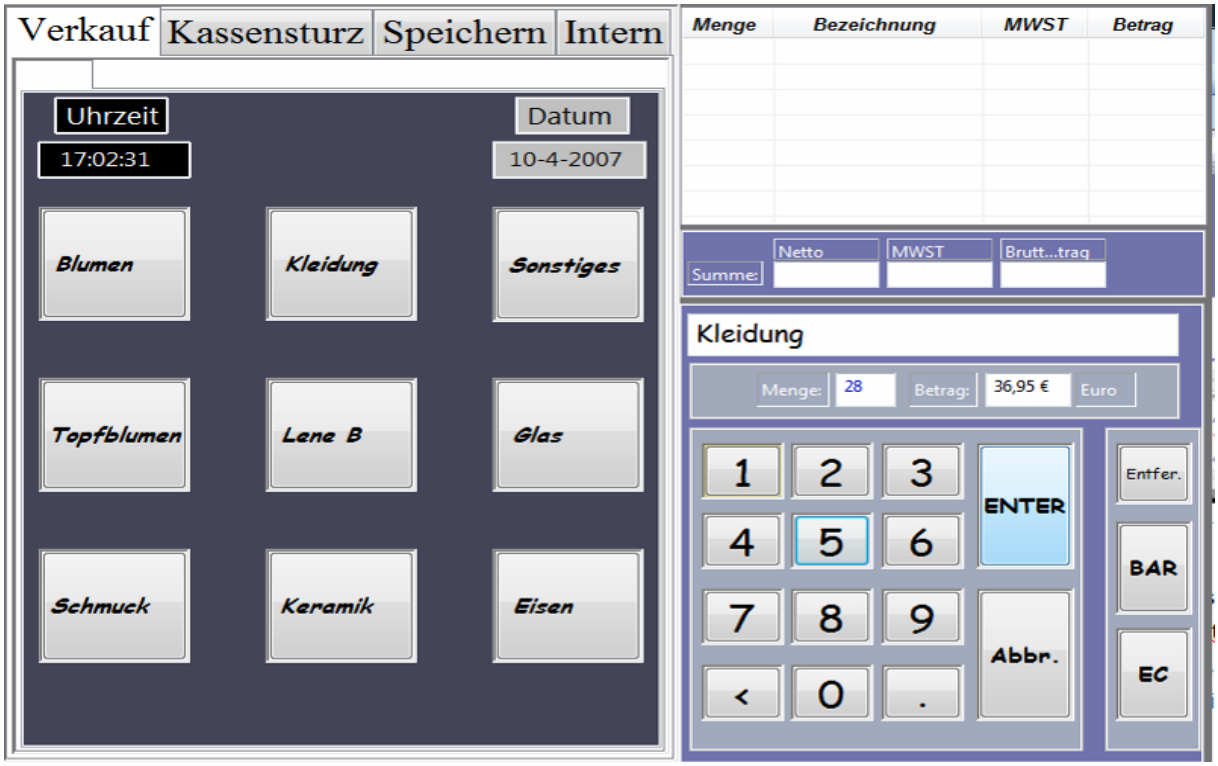

Figure 5-5: Input the single price of a product

The price of a single product is then input by the employee. If the “Enter”-Button is pressed once again, the input amount as well as the price are added to the controller’s basket linked List, together with the chosen product object.

The content is to be displayed on the line display. Therefore the display has to be cleared. After it is cleared, the content is written by the method displayText(). The number of products, the name as well as the complete price (number * price) are displayed.

The same information is added to the itemTable. The method convertListToString() of the class ActionClass converts this information into a corresponding String array that the table is capable of reading.

Figure 5-6: Line display, what the client can see

After a product is added to this table, the net amount, the gross amount as well as the tax amount have to be updated.

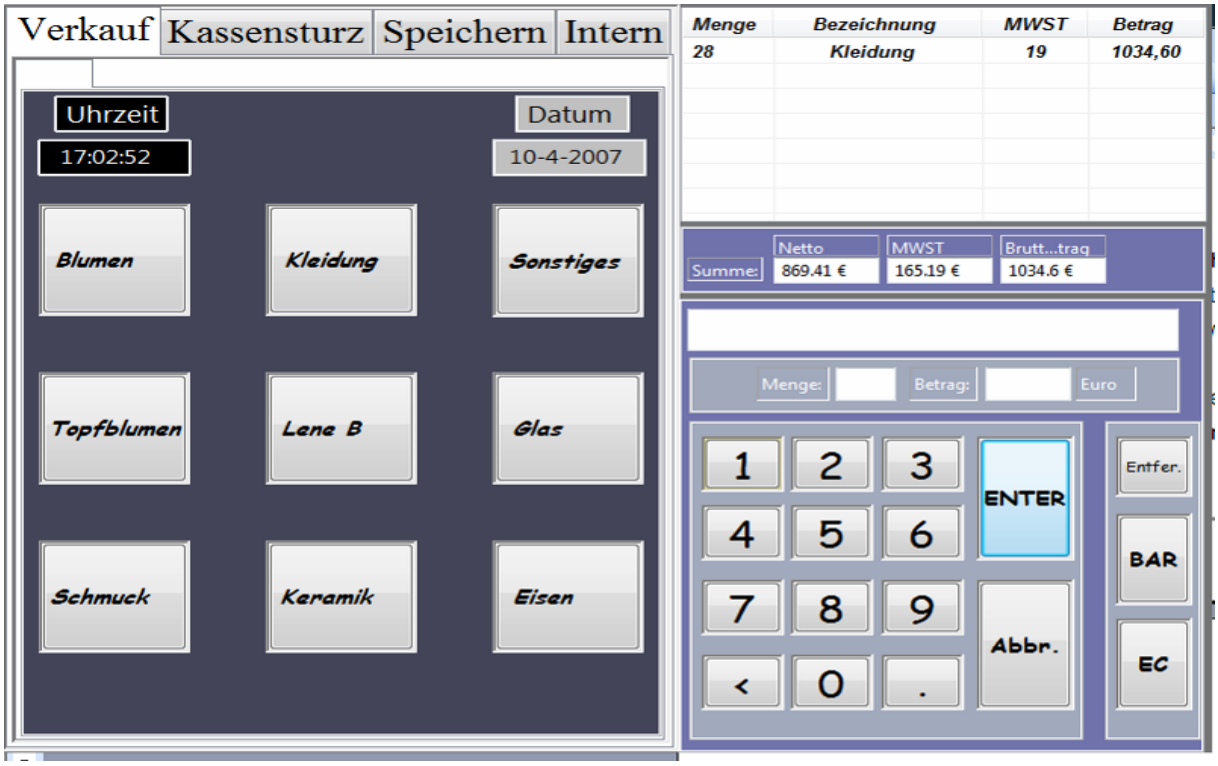

Figure 5-7: Displaying of net amount / gross amount

The gross amount is shown on the textField “grossAmount”. A method of the class ActionClass, countTotalPrice() returns the price of all products that are contained within the basket linkedList.

The same holds true for the netAmount with the difference that total tax, evaluated by the method countTotalTax(), is subtracted from the grossAmount.

After all products are added to the basket linked List, the payment method is chosen. EC-Card and cash payment are action Listeners of class ActionListenerClass.
The action Listener for payment via cash checks to determine if the basket linked List is not empty. The textField “textBetrag” is set “editable”, so that the given amount can be input into this field. The price is displayed on the lineDisplay.
After the amount of the money tendered is input, the “Enter”-button is pressed. This causes a call of the function isEnough() of class ActionClass. This method checks to determine if the entered amount of money is equal or bigger than the amount the customer has to pay. If this check is correct, the content is written on the database.
The sold articles are inserted into the database table under tableSoldArticles.
The method insertArticles() that is contained in class DBSoldArticles is called as often as products are contained in the basket linked List. The method convertToStringForDatabase() gathers all information and stores them into a String that is returned to the calling method. The payment method is set to “BAR”.

An activation of the method printText() of class UsePrinter prints a receipts for the customer. This is done by first calling the method createBill() of class ActionClass. This method creates a String in an appropriate format.

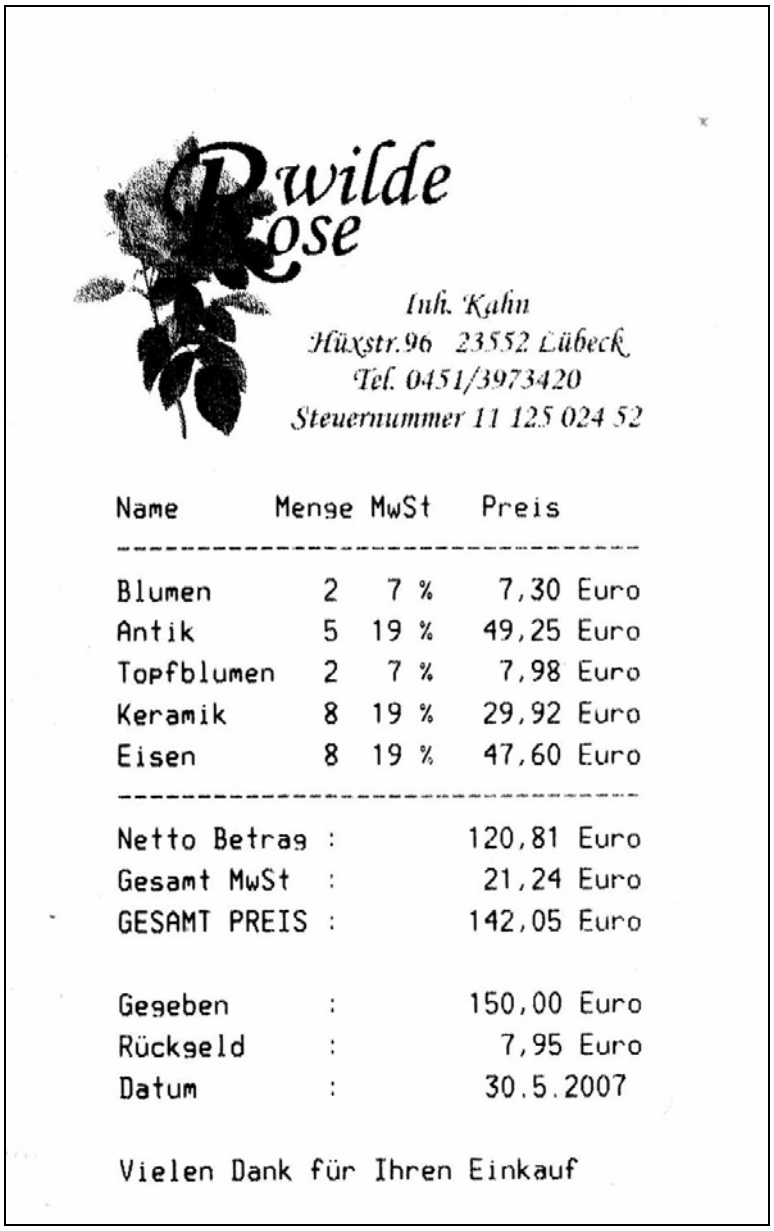

Figure 5-8: The printed receipt

The other payment method that can be chosen is payment by EC-card. EC-payment in the context of this project means that the client uses the already available EC-card reader and inputs the information by hand. EC-payment for the cashier system is necessary because there has to be an entry in the database that these items were paid by EC-card.

EC-payment works nearly the same as payment via cash with only one restriction that there no need to check the entered amount of money. If the articles are added to the basket linkedList, pressing the EC-button effects a direct storage into the database as well as printing of the receipt.

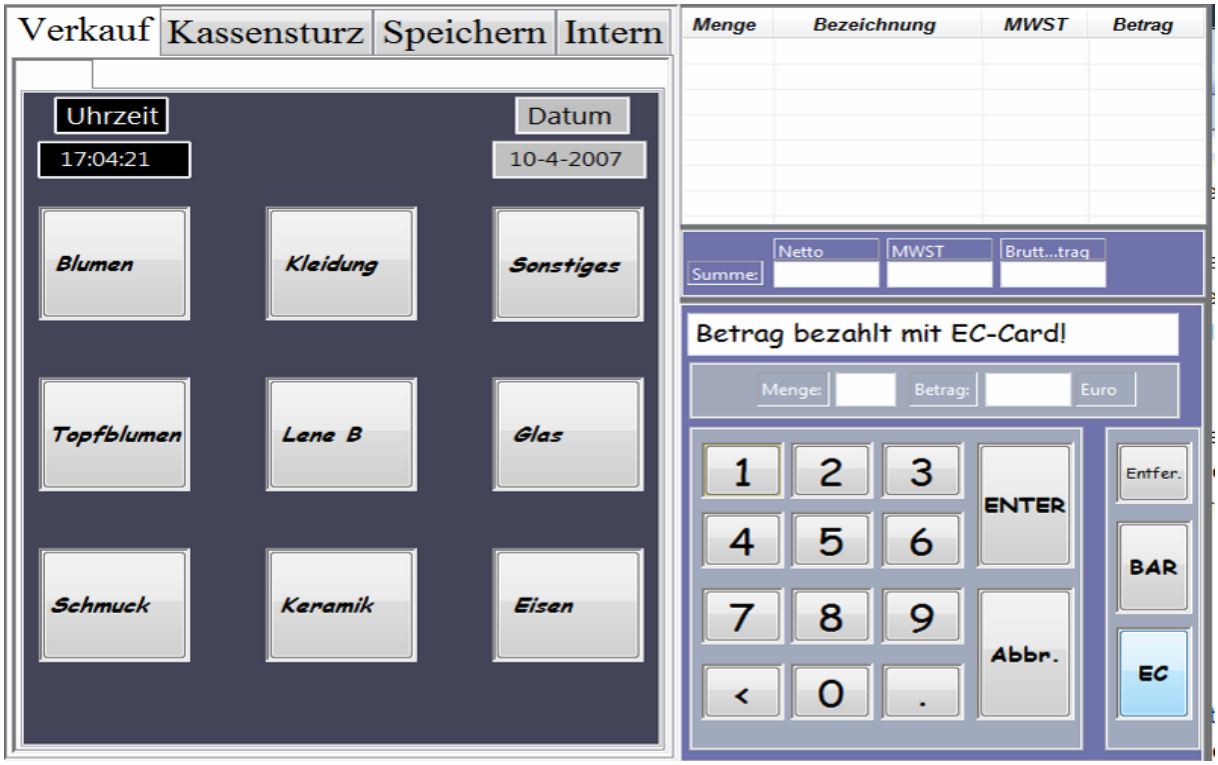

Figure 5-9: EC-payment is chosen, payment successful

Both action Listeners, for cash as well as for EC-payment provide a clearing of all parameters in common. After the articles are paid and written into the database, linked List as well as text fields and variables, are reset.

5.3.3 Accounting Cashier

Everyday a cash flow check has to be executed. The money that was in the drawer at the beginning of the day has to be compared with the actual money there at the end of the day. The difference between these two amounts is the cash input of sold items minus the money that was taken out of the cashier for salaries.

The actual money that is in the drawer at the end of the day is input by hand. The cash is counted by counting the number of each denomination of coins or bills, and entering this quantity in the touch screen. The system then calculates the total cash. The GUI's tab "Kassensturz" contains buttons all of which have action Listeners to input the money in the drawer the employee has counted.

A method setCountedMoney() of class ActionClass adds these inputs and returns a String in German money format.

Figure 5-10: The tab “Kassensturz” is selected

When the employee has finished counting, the button “Fertig” has to be pressed.

Figure 5-11: The amount of money in the drawer is input and the cashier saved the information successfully in the database

This action Listener first checks to determine if the cash check for this specific day has already been made. A method in class DBAccountingCashier, alreadyAccountingDone(), queries the table tableAccountingCashier for an entry for the specific day.

The action Listener checks to see if the input amount of money in the drawer is not zero.
The accounting is done by inserting the information into databases table tableAccountingCashier. This is done by calling the method insertArticles() of class DBAccountingCashier. This method inserts a transactionID, the date, the income (what was sold), the expenses (amount taken out of drawer for salaries), the counted money in the drawer, as well as the difference between the money in the drawer and what was taken out of it.

In order to know how much money was in the drawer in the morning, one needs to know the amount of money in the drawer from last day. The method

readMoneyInDrawerFromLastDayFromFile() reads this value from a text file and returns its amount.

The salaries that were taken out of the cashier need to be added to the difference. The method getGehaltTakenOutOfCashier() queries the databases table tableWorker.

The amounts for a specific day are added and then returned to the calling method.

The next step is to store this accounting information in a text File. The method queryDataFromDatabase() of class DBAccountingCashier queries the database for all items and then writes the content into a text file.
The counted money in the drawer needs to be written into a text file for the next day's cash check. This is done by calling the method writeMoneyInDrawerToFile() of class DBAccountingCashier.

The cashier system can now be switched off by pushing the action Listener buttonForShuttingDownSystem.
This action Listener of class ActionListenerClass first checks to see if the cash check has already been made. Therefore the already explained method alreadyAccountingDone() of class DBAccountignCashier is activated and a boolean "false" is returned if the accounting has already been performed.

In order to shut down the cashier system, a runtime object is generated and the system command
"shutdown –s –t 00 –f" [9] is executed.

a) s specifies that the computer is to be shut down
b) t specifies the time the system has to wait to be shut down
c) f closes every active program

This shutdown-program is part of all Windows NT-derivatives.
The command System.exit(0) closes the cashier program.

5.3.4 Exporting accounting data

The cashier software offers a means to save the sold articles as well as salaries taken out of the cash drawer directly onto a USB-stick. This is done in order to enable the client to evaluate this data externally on another computer. All accounting information that is stored in the database will be queried and saved onto an USB-stick. This enhances mobility of the data and ensures efficient administration.

The employee has to input the month and year; the accounting data from this month are then stored on the stick.

Figure 5-12: The tab “Speichern” for external storage on USB-stick is chosen

The tabFolder “Speichern” provides a button to save the data onto an USB-stick. This button has an action Listener that performs several operations.

A method queryDataFromDataBase() is called within the classes DBSoldArticles as well as DBWorkers. These methods query the tables tableSoldArticles and tableWorker. These two tables contain the data that needs to be externally stored.

The month and the year can be input by using the numerical field. This is confirmed by pushing the “Enter-Button”. A method is called with the inserted month and year.

If no date is specified, the actual months and years data are taken.
The content that is stored in these tables for the specified date is then stored in an Excel File onto the connected USB-stick.

Figure 5-13: The accounting month is input and the accounting data is successfully saved on the USB-Stick

5.3.5 Salary withdrawal

The client is able to take money out of the cashier's drawer and pay his employees directly with cash money.

Figure 5-14: The tab "Intern" for payment of employees is chosen

To do this the tabFolder "Intern" needs to be selected.
With the help of the numerical input field the employees ID is input. After this, the amount that is taken out of the cashier drawer is input using the numeric input field.

The program first checks to determine if all information is input correctly, the ID as well as the salary to be taken out of the drawer. If all information is available, the data are inserted into the databases table tableWorker. This is done by calling the method insertArticles(), class DBWorkers, with the parameters ID as well as the amount taken out of drawer. The cash drawer is opened and the money can be taken out of the cashier.

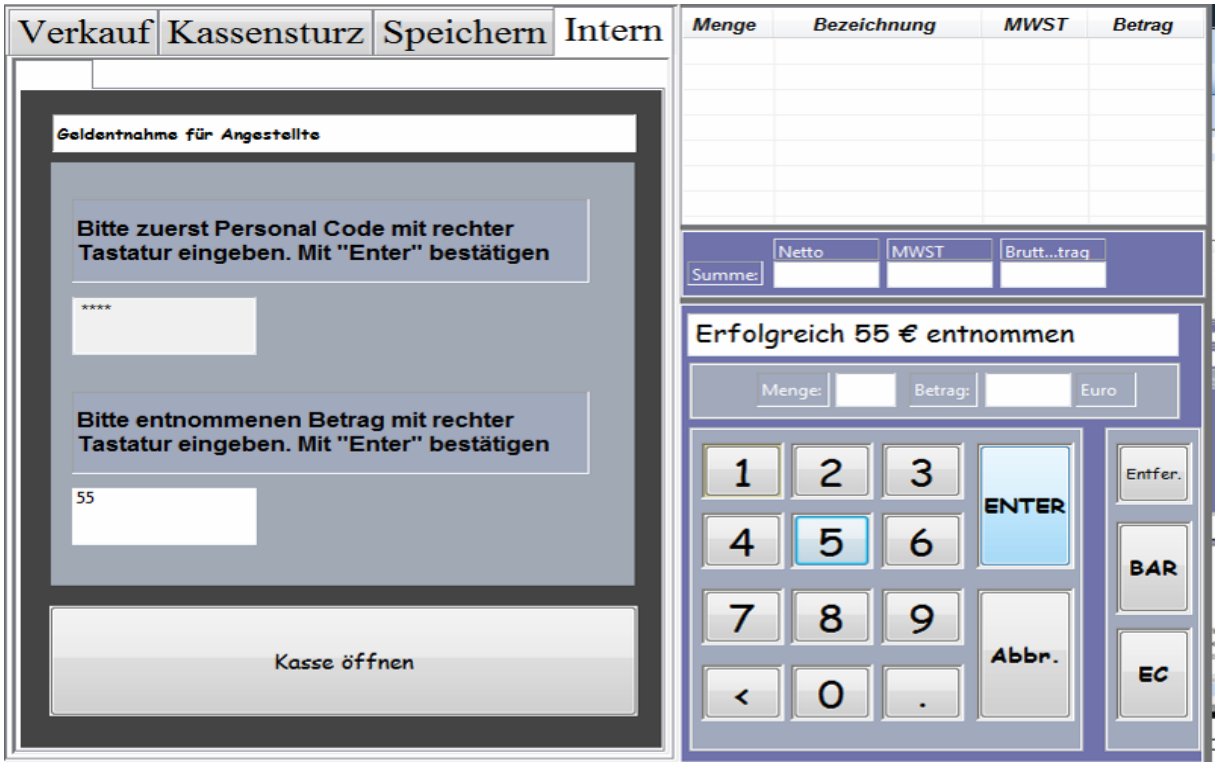

Figure 5-15: The personal code as well as the amount taken out of drawer is input

5.4 GUI realization

For the GUI realization, a specific class is created. This class contains all objects for buttons, shells and tables. These objects are needed to create the graphical user interface.

The implementation of this GUI is done with the supporting software tool “Visual Editor”. This editor allows for visually aided creation of graphical user interfaces. The SWT objects are added via drag and drop to the editor that automatically generates the source code for this user interface.

There is a general requirement for object oriented programming languages that the GUI and the application software have to be completely separated from each other. This means that no operations or methods are placed within the GUI’s class that are needed for the operations of the whole software.

An specific class “ActionClass” contains all operations to set and evaluate inputs. All action Listeners are transferred onto an specific class called “ActionListener”.

5.4.1 Layout

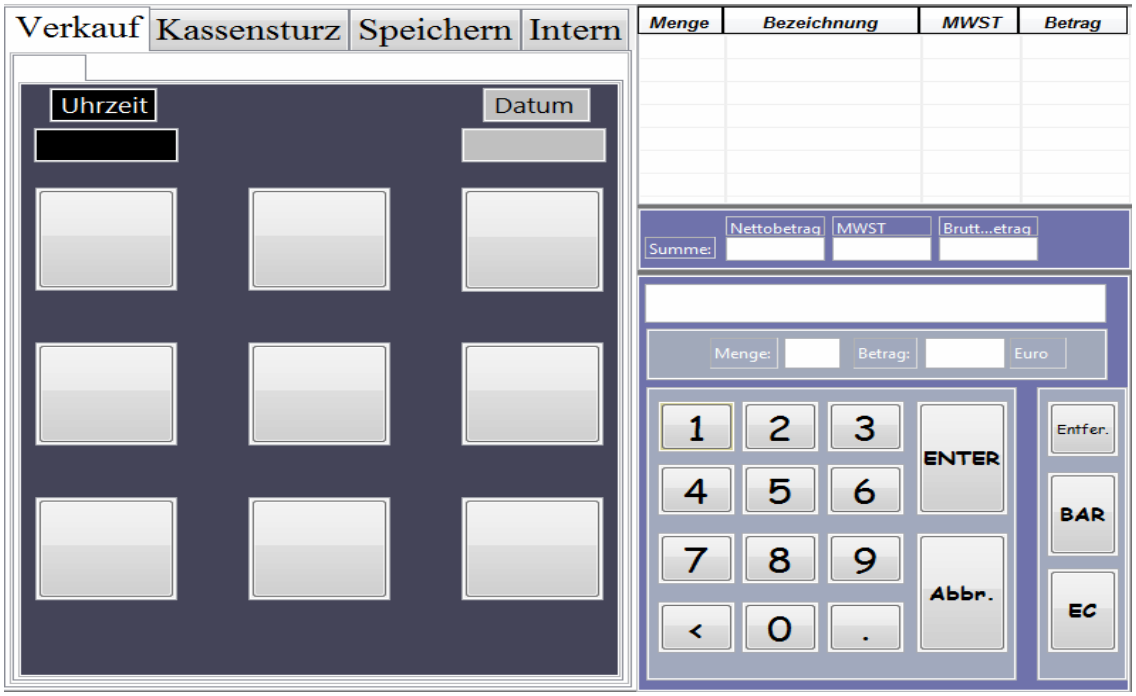

Figure 5-16: The GUIs Layout, final version

5.4.2 Layers

The main layer of this GUI forms a shell. A shell is a window, controlled by the operation system window manager. Every SWT application requires at least one Display and one or more Shell instances.

The Shell forms the group plate of the user interface, where other layers can be placed.

The shell consists of four different layers.

- tabFolder → A tabFolder is a SWT widget that allows users to select a page from a set of pages.
  - Verkauf → products as buttons
  - Kassensturz → accounting
  - Speichern → save data on USB-stick
  - Intern → worker's salaries taken out of cashier

Figure 5-17: The TabFolder

- Table → shows a table where the sold articles are displayed

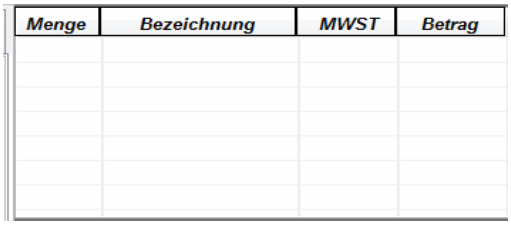

Figure 5-18: The Table

- Lump sum → the net amount, the tax amount as well as the gross amount

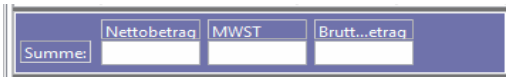

Figure 5-19: The Lump Sum

- Numerical field → Buttons to input the amount of money, the amount of products, the date for saving the data on USB stick as well as worker ID's, Payment method and remove button.

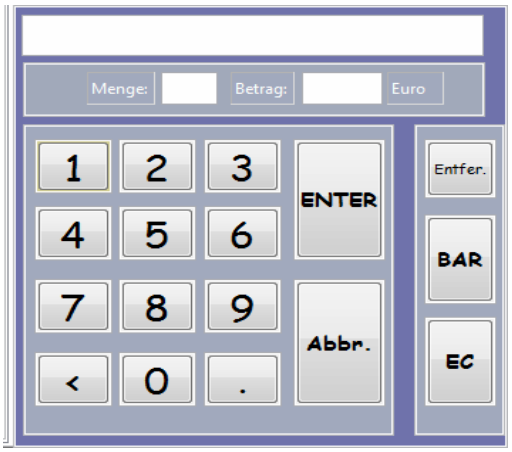

Figure 5-20: The numerical field

5.4.3 SWT-objects

For the creation of this GUI, different objects are used all of which belong to the SWT libraries. In addition to the already explained objects, shell and table, the following SWT objects are applied.

TableItem

The table rows are created with a tableItem object.

The implementation

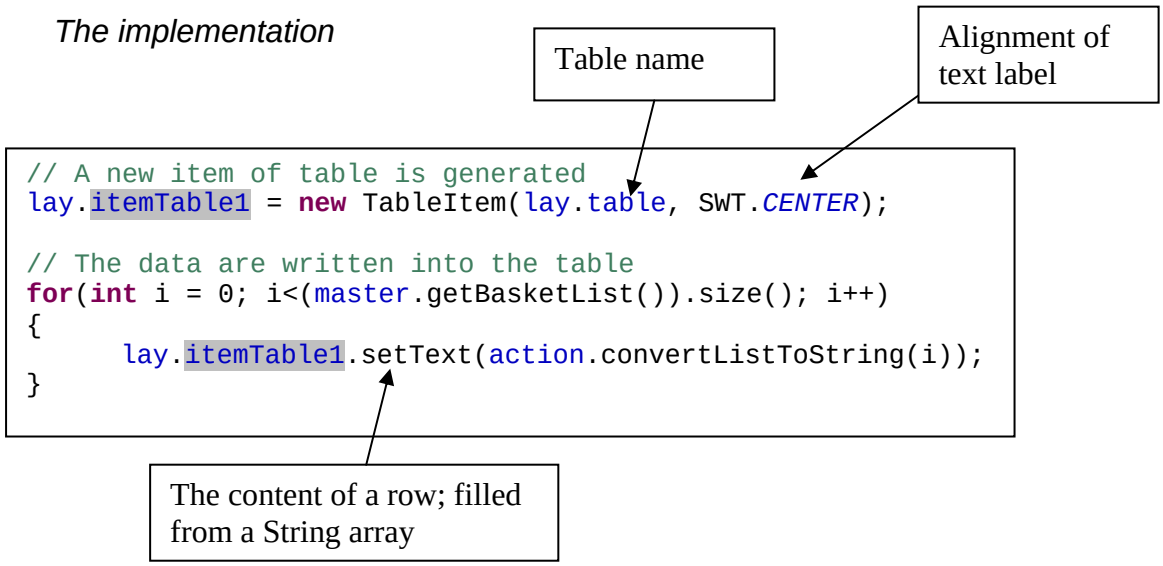

Figure 5-21: Implementation of the TableItem

5.4.4 Action Listener

The class ActionListener contains all action Listeners that are needed to operate the program. An action Listener is an event actuator that performs any given action when the specified event occurs.

The object, as an example a button that sets 50€, adds a selection Listener to the selector's notification list. A selection Listener is an adapter class that provides default implantations for the methods that are described by the Selection Listener interface.
The "widget Selected" method is sent when the selection occurs. When it is activated, the assignments that follow are executed. In this example it is a call to display 50€ on the screen.

5.5 Environment Realization

5.5.1 System Realization

With the above mentioned hardware modifications, the IBM SurePOS 500™ runs very well with the operating system Windows XP™. On this installation all new Microsoft updates must be added via the Windows XP™ update or via an off line update. To increase the performance of this system further unnecessary services and functions have been deactivated, like the Windows firewall, the Windows security center, the IMAPI-CD-BURN-SERVICE, the balloon tip and unneeded auto start applications.

5.5.2 Client Point of Sale (POS)

In order to run the POS software in the flower shop the following software is required:

- The Java Runtime Environment 6™ (JRE 6).
- The latest version of MySQL 5.0™.
- Touch screen software

All project components such as jar files, dll files, xml file and properties files are stored in the directory. An entry in the classpath points to this directory.

5.5.3 Developer Point of Sale (POS)

A second POS Hardware was configured as a development platform. For the development environment the following additional software was installed:

- Eclipse 3.2
- Java Development Kit
- JavaPOS™ Software
- Java Pos Editor
- JavaPOS™ Control software

5.5.4 Challenges

Some difficulties arose while getting the development system running. In the beginning the not even the examples in the IBM JavaPOS™ would run on the system. First the Java Development Kit™ (JDK™) and JavaPOS™ software were installed. The installation of JavaPOS™ automatically set the classpath correctly so that the needed IBM Java™ class files, the hardware information XML file, the IBM hardware properties files were known to the system.
But still the Point of Sale hardware was not able to communicate with the devices.
The Java Runtime Environment™ (JRE™) did not contain the needed information to connect to the serial communication port.
In order to fix that problem the Java™ COMM Package needs to be placed in the Java™ environment (bin file).
With this the IBM JavaPOS™ example could be executed.

After the project had been developed under the Eclipse environment and the client POS system had been prepared, some difficulties with the standard Widget Toolkit (SWT) occurred. The Standard Widget Toolkit (SWT) was not known to the operating system. This is not solved by only add the the SWT to the classpath, but also a special SWT Windows© library file needs to be added to the projects folder or the classpath in order to make it work.

5.5.5 XML Configuration

An XML file is delivered with the POS Hardware which includes all possible IBM and generic devices that can be attached to this basic POS Unit. The challenge for the developer is to select the devices which are to be attached and to remove all other configurations. Furthermore the device parameters have to be reviewed and modified to fit this individual installation.

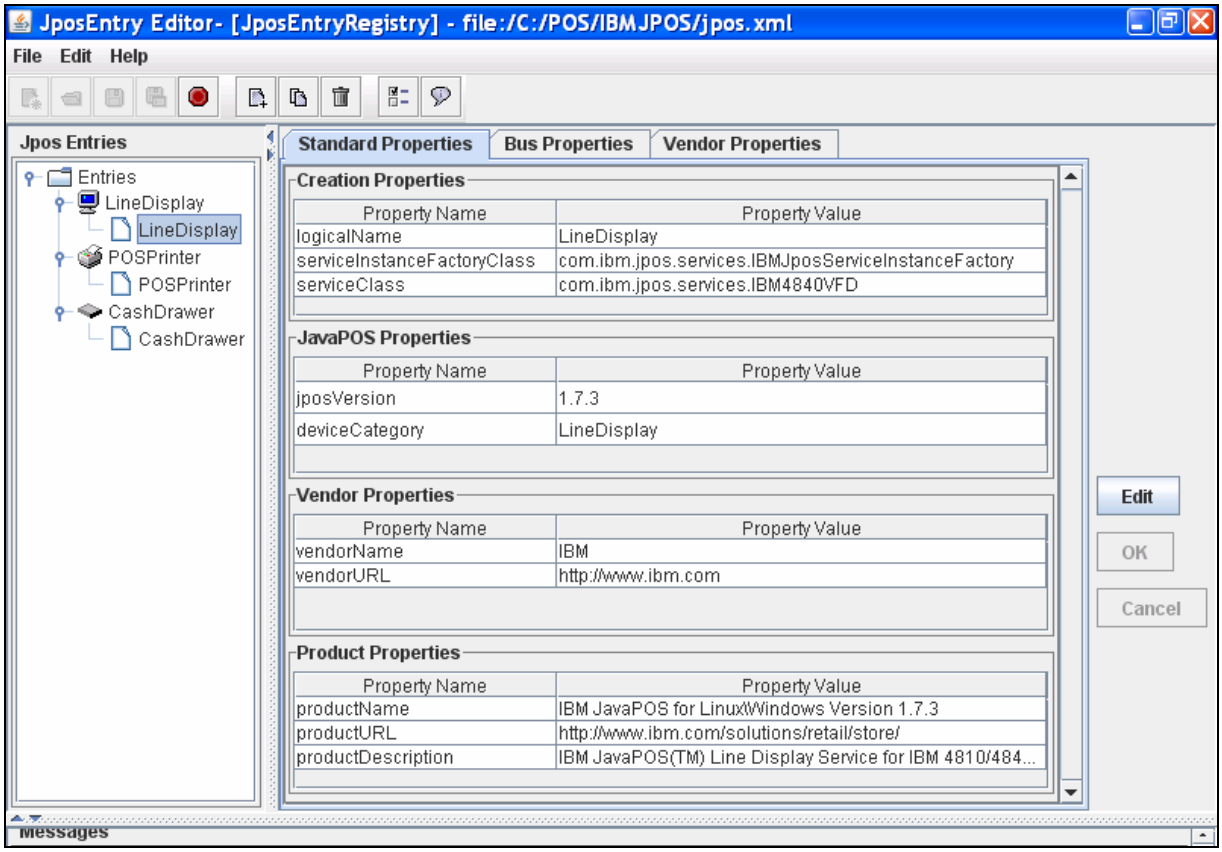

Figure 5-22: JposEntry Editor

There are several ways to configure the XML file:

- Using the JposEntry Editor is a comfortable way of configuring the device hardware specifications and the logical device name. Under this editor devices can be added or deleted.
- Using the JposEntry Editor in conjunction with the JavaPOS™ Control Software. This software scans the XML file and checks every device if it is connected to the Point of Sale. After the scan only a few devices per device group are left in the list. Now the developer views the remaining configurations and deletes all devices except for the ones that are to be used.

- By “hand” using a normal text editor. Here the developer needs to know which devices are connected to the Point of Sale, so that unused devices may be deleted. This is a cumbersome method but very fast.

5.5.6 Project implementation

With this batch file it is possible to run the project via a batch command.

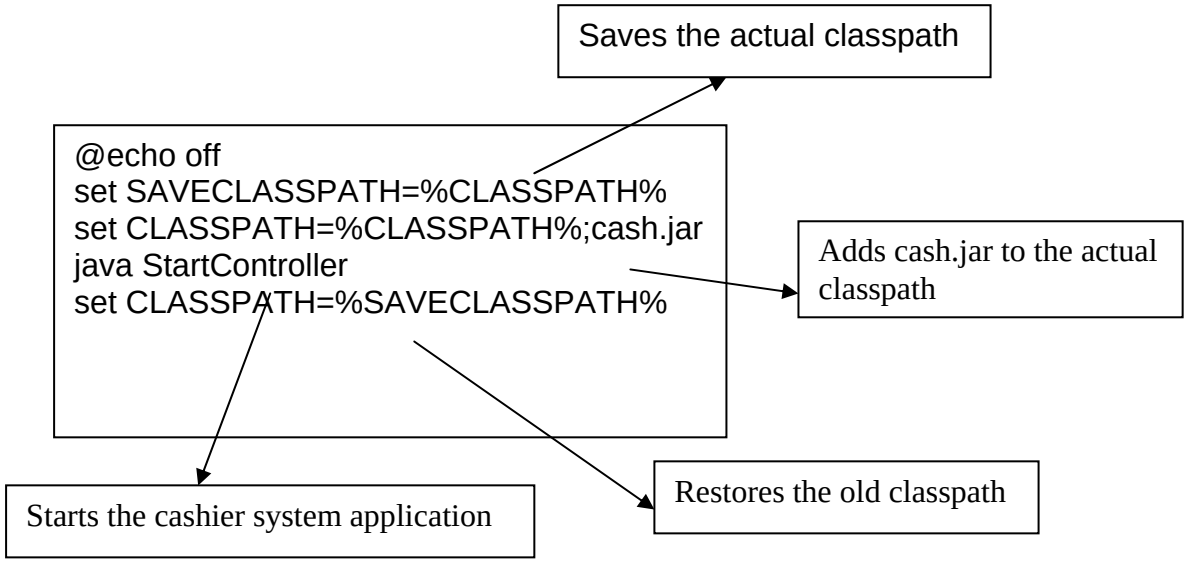

Figure 5-23: Batch instruction

- The *@echo off* means that while this batch runs no messages are given.
- In the second row the actual classpath is saved.
- In the third row the project is added to the classpath.
- Then in the fourth row the cashier program is started.
- In the last row the old classpath is restored.

5.6 Room Plan

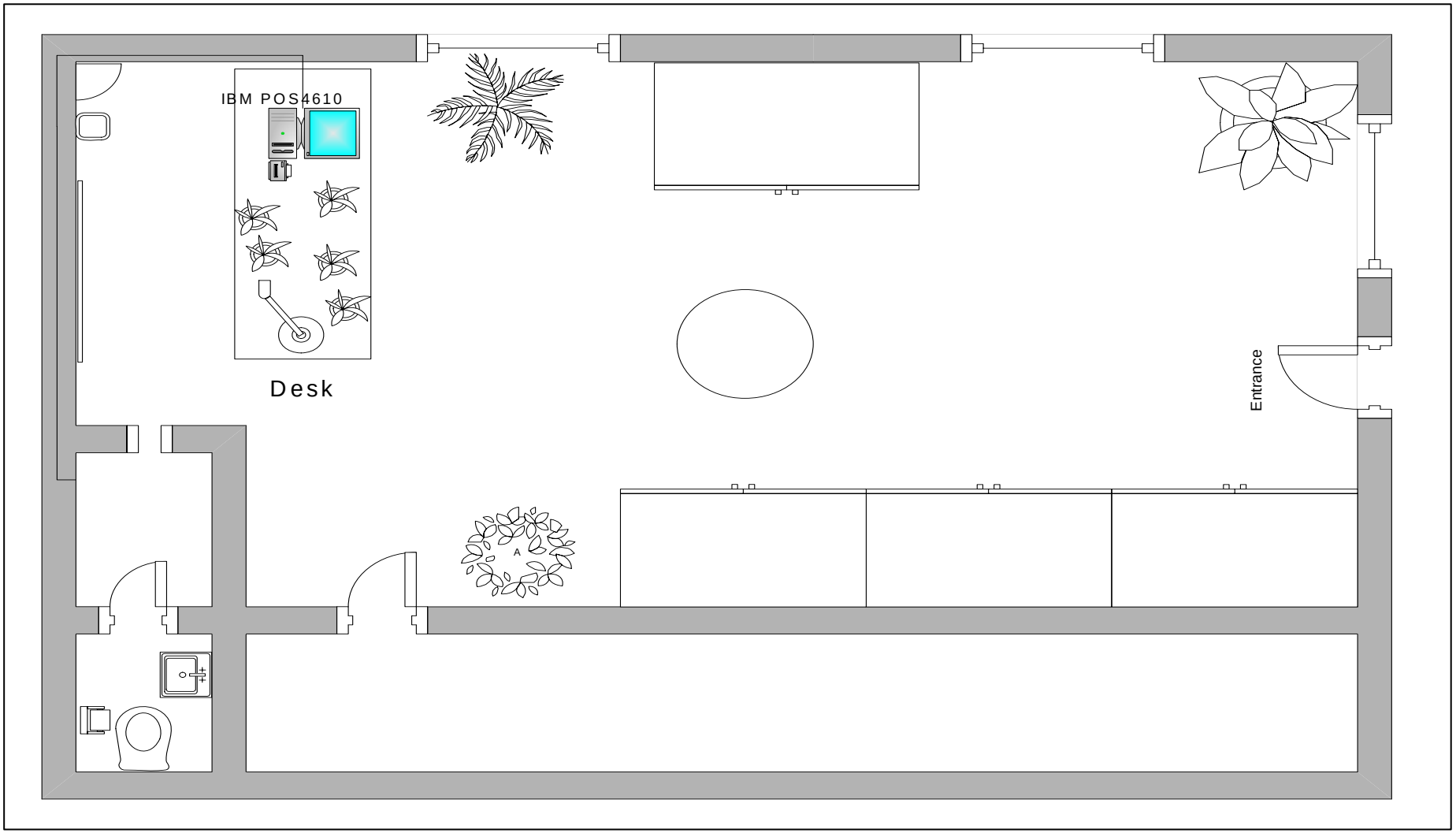

Figure 5-24: Room Plan

In this figure a room plan of the flower shop can be seen. The shops location is Lübeck in a shopping street.
The cashier system is positioned on the desk where the articles are sold. For further developments, an adjacent room is positioned where a server could be placed. The desk comes with several power plugs that are needed in order to operate the cashier. A LAN cable is not available. For further developments, an installation of such a LAN cable is possible.

5.7 Software validation

Software validation plays an important role in software engineering.
Testing and validation of software are divided into two parts:

1. Testing by the software developer
2. Testing by the client

5.7.1 Testing by the software developer

Testing by the software developer is done throughout the whole development phase. It is checked to determine if the generated software behaves as previously defined.

The testing applied to this software project can be divided into three parts:

First of all a normal test case is performed. This tests normal inputs. According to the cashier's software this means choosing a product, adding the amount, the price and then paying. This testing can also be applied to cash checks as well as to saving data on the USB-Stick

Secondly an Output Forcing test ensures that all software outputs are tested. This is done to determine if the cash check generates reasonable outputs. Therefore the database is fed with predefined values (product information like price, name and amount). The cash-check values are compared with the manually calculated values.

The last test that is performed is a robustness test. The software also has to behave correctly, when an invalid input is created. It checks what happens when a price is entered that is not plausible, e.g. a price with 2 commas. The program must not crash in any of these circumstances.
Robustness in this case also means that the software has to work correctly even after several thousand operations. This is the main operation field in the flower shop.

5.7.2 Testing by the client

When developing individual software, the client is ultimately responsible for what he gets. He must be given the opportunity to exercise this responsibility. Testing with the client should begin at an early stage. After the analysis of the client requirements, the developer produces a blueprint of the software he proposes to produce. This blueprint is signed off with the client, and is in fact the first level of testing, to see if the developer and client agree on what is to be developed.

After the development has progressed far enough that the input and output are visible, the client should once again become involved, to confirm that this fulfills his needs and meets his expectations. It is typical that gaps between the developer's vision of the project and the clients vision of the product are identified at this point. At this point in the development it is much easier to make adjustments that affect database architecture, business processes or other fundamental concepts than later after all the code has been completed and tested.

The client should also be involved in testing, after each module of the development is programmed. One big advantage of having the client assist in testing is that he will always do things differently than what the programmer expected. This uncovers the blind spots in the programmers testing techniques.

When the programming development is finished, the client begins the so called integration test. Here all business processes are exercised in the proper sequence. This is a final test of the functionality and also serves as training for the client personnel. This process may be iterative until a degree of software perfection has been realized.

After the integration test has been successfully completed, the software is ready to be taken into production. During the early production phase, change requests often crop up. Therefore the developer should be prepared to support the client in the first months of use with fine tuning of the software.

6 Conclusion

6.1 Chapters overview

This chapter draws a conclusion of this software project. The agreed program features and requirements between client and developer are checked for proper implementation. Future developments will also be discussed in this section.

6.2 Implementation evaluation

The client and the developer check for the proper implementation of the given specification.

A new software tool with the given requirements has been created. This is fulfilled by the developers. The task was to create a software that is extendable. This is achieved by using the object oriented programming language Java™. The clients IBM cashier hardware has been reused. This is done with some small system changes like upgrading the hard disc as well as the system memory in order to speed up the operations.

The project had to be set up on a low cost basis. This includes the usage of available components as well as free software. A usage of the free database software MySQL is one example for the cost efficient decisions and implementations.

A graphical user interface has been created. The product buttons were created as big as possible. The requirements that the product names as well as the tax values are changeable without changing the system code is solved by providing a text-file that holds all possible product groups. This file is read into the system at start-up time. Time as well as date is displayed on the GUI permanently as desired.

The two payment methods, cash and EC-card, are possible as required by the client.

The net and the gross amount of the products basket is displayed on the screen as desired, so that the clients' employees can see this information easily.

The agreement to use the SWT toolkit is followed.
An external storage function has been implemented that stores the accounting information onto an USB-stick. It is possible to download any selected accounting month.

Accounting is a central requirement towards this software tool. After each transaction the sold articles have to be stored. An accounting summary at the end of a day has to be created. Information about employee's salaries are also stored in a proper way. The clients' decision to use a database system, in this project MySQL, is satisfied. The accounting functionalities work as expected.

6.3 Future developments

The POS System is in production in a small Flower shop since the Feb 26. 2007.
From this shop improvements are recommended on regularly basis.

6.3.1 ”Willkommen” message in line display

One of the further developments is that a customer is welcomed over the line display: ”Willkommen”. One way to achieve this is to access the line display after a customer has paid and the receipt is printed and send the “Willkommen” out to the display. The disadvantage of this approach is that the “returned money” is not displayed anymore. Another approach could be to use a thread which terminates after a certain time and displays the welcome text.

6.3.2 Additional detail in day-end report (“Kassensturz”)

The daily report has to be extended to differentiate between the different sales tax percentages. This is a requirement of the German tax authorities. Furthermore total sales has to be divided between EC-Card payments, Cash payments and Invoices. An entry field for cash withdrawals (payment to bank) has to be added.

Tag	Kalender Woche	Gesamt Umsatz	Ware: 19 % A	Ware: Blumen 7 % B	Ware: Topfblumen 7 % C	Kassenstand Am Morgen

EC Zahlung	BAR Zahlung	Rechnung oder Überweisung	Kassensturz	Differenz	Entnommenes Geld

Figure 6-1: Example Day-End-Report

This requires some significant changes to the project, because both the database and various methods have to be changed

- The database needs to be modified
- The method which reads the products must be changed, because more information needs to be queried from the database. The method must determine whether the product belongs to the group A, B or C.

6.3.3 Deactivation of Quantity Field (“Menge”)

Practice has shown that the quantity sold is usually 1. The mandatory entry of the quantity slows the data entry process. The quantity field should be closed for entry and set to default “1”. The quantity can be useful, if several units are bought at once. Then the total amount is quantity * price. In order to activate this feature, the sales person should press the quantity field, and it will then open for entry. The quantity can be entered and used in conjunction with the price to calculate the total amount. This requires no changes to the database and associated classes and methods.

6.3.4 Suppress receipt print-out

It turns out that a lot of costumers do not need or want a receipt. In order to protect the environment it would be a good idea to find a way to only print a receipt if it is wanted. The approach could be that a popup shell asks for this decision.

6.3.5 Goods returned button

The cashier system does not provide a goods returned button (“Gutschrift”). In the requirement analysis of the project this process was not identified. But it turns out that this button is really needed. Customers do occasionally return both flowers and other goods and get their money back. The cashier needs a button which will reverse the sign of the sales amount and quantity. This returns line should fit into a transaction, so that it can be combined with other purchases or used alone. In the first case it would reduce the total amount due, in the second case it would calculate an amount which is to be paid out to the customer. The database structure stays the same, because amount is simply stored as a negative value in the database.

6.4 Developers conclusion

The total development of this software took about 6 months. This is a very short time frame for developing such software. The Cashier systems works as expected and as reliable as other software systems from other vendors that are sold for several thousands of euros.
As soon as a test period of 6 months is successful, the initiators of this tool will think about a possible commercialization. The software can be adapted to other requirements without big problems.

The work in a team of two people was very efficient during all phases of the project. A division of the tasks guaranteed a fast and parallel development as well as good results when writing the code for this software.
Discussions about important decisions guaranteed an optimal solution for the client as well as the developers.

7 References

The references that were used during the development process of this software tool are listed in this paragraph.

[1] http://www.mysql.de/why-mysql/toptenreasons.html, top 10 reasons that argue for the usage of MySQL (visited 01.03.2007)

[2] http://www.javapos.com/downloads/JavaPOSfaq.pdf, what is JavaPOS™, JavaPOS™ FAQ (visited 18.03.2007)

[3] Objektorientierte Softwareentwicklung, Analyse und Design mit der Unified Modeling Language, Bernd Oesterreich, Oldenburg Verlag, Release 4, ISBN: 3-486-24787-5

[4] Java™-Entwicklung mit Eclipse 3.1. Anwendungen, Plugins und Rich Clients, Berthold Daum, D-Punkt Verlag, Release 3, ISBN: 3-89864-338-7

[5] Script Computer Science II V 2.3, Database Management, Prof. Bernd Kahlbrandt, Chapter 14 Products, page 144 ff

[6] http://de.wikipedia.org/wiki/GNU_General_Public_License, General Public License System (visited 18.03.2007)

[7] http://www.mysql.com/products/connector/j/, the speed advantage of MySQL-Connector / J in latest version (visited 21.03.2007)

[8] Java™ 2 Primer Plus, Steven Haines, Stephen Potts, SAMS publishing, ISBN: 0-672-32415-6

[9] Shutdown Windows XP with Java™ commands, http://entwickler-forum.de/archive/index.php/t-2867.html, forum post, (visited 20.03.2007)

[10] Java™ in a Nutshell, David Flanagan, publisher O'Reilly, Release 4, ISBN: 0-59600-283-1

8 Appendix

8.1 Glossary

*.jar file:	stores the compiled Java™ code and the associated metadata
Action Listener:	event Listener that responds to the user's indication that some implementation occur
Adapter class:	class used to reduce the code for event Listeners
AMD:	“Advanced Micro Devices”, manufacturer of integrated circuits
API:	Application programming interface, source code interface in order to provide requests for services
ATX:	“Advanced Technology Extended”, form factor for motherboard and computer chassis
AWT:	“Abstract Window Toolkit”, Java™’S platform independent windowing widget toolkit
BIOS:	“Basic Input-Output System”, basic software that loads Computer after power-on
Buttons	control element in a graphical interface
cash check	establishing the inventory
CD-ROM	“Compact Dics – Read Only Memory” read only media, stores digital data
CLabel	SWT-Element, used to display text
Cleartype Fonts	Software technique developed by Microsoft to enhance readability of fonts on LCD-displays
Cluster concepts	Data structure with different data types
COM port	Serial Computer interface
Database	System for electronic data management
DB2	relational database, IBM
DBMS	“DataBase Management System”

dual line display	Display with two displayable columns
Eclipse IDE	“Integrated Development Environment”, in this case Eclipse
EXCEL-Table	spreadsheet calculation program by Microsoft
FAT 32	“File Allocation Table”, file system by Microsoft
flash storage	digital storage chip
Floppy	magnetic data medium
GMF	“Graphical Modelling Framework”, open source Java™ Framework
GPL	“General Public License” , Free Software Foundation
GUI	“Graphical User Interface”, type of user interface which allows people to interact with a computer or other media formats which employs graphic icons
HDD	“Hard Disc Drive”, non-volatile storage device which stores digitally encoded data on rapidly rotating platters with magnetic surfaces
HID	“Human Interaction Device”, a computer device that interacts with and takes input from humans, such as the keyboard, mouse, etc
hierarchical model	Data are organized into a tree-like structure
IBM	“International Business Machines Corporation” is a multinational computer technology corporation
Internet	is a worldwide, publicly accessible network of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol (IP)
Java™	object-oriented applications programming language developed by Sun Microsystems
JavaPOS™	“Java for Point of Sale” is an application interface, written in Java to provide common access to POS peripheral devices
JDBC	JDBC is an API for the Java™ programming language that defines how a client may access a database
JNI	Java Native Interface (JNI) is a programming framework that allows Java™ code running in the Java™ virtual machine (VM) to call and be called by native applications

JRE	“Java Runtime Environment”, allows a computer system to run a Java™ application
JVM	“Java Virtual Machine”
LAN	“Local Area Network”, computer network covering a small geographic area, like a home, office, or group of buildings
Layer	The target image is produced by "painting" or "pasting" each layer, in order of decreasing depth, on the virtual canvas
Linkedlist	fundamental data structures used in computer programming. It consists of a sequence of nodes, each containing arbitrary data fields and one or two references ("links") pointing to the next and/or previous nodes
Maxtor	Was an American manufacturer of computer hard disk drives founded in 1982 and acquired by Seagate in 2006
Micro ATX	is a small PC motherboard standard, with a maximum size of 9.6"×9.6" (244mm×244 mm)
Microsoft SQL Server	relational database management system (RDBMS) produced by Microsoft
Microsoft Windows 98 (Second Edition)™	Graphical operating system released on June 25, 1998 by Microsoft and the successor to Windows 95. Like its predecessor, it is a hybrid 16-bit/32-bit monolithic product based on MS-DOS
Microsoft Windows XP (Professional Edition)™	Operating System Windows XP is the successor to both Windows 2000 and Windows Me, and is the first consumer-oriented operating system produced by Microsoft to be built on the Windows NT kernel and architecture
Microsoft® Windows® 2000 (Professional Edition)™	Interruptible, graphical and business-oriented operating system that was designed to work with either uniprocessor or symmetrical multi-processor 32-bit Intel x86 computers
MSR	“Magnetic Stripe Reader”, reads magnetic stripe cards that contain data for e.g. credit cards
MySQL	is a multithreaded, multi-user SQL database management system (DBMS)

NCR	NCR Corporation (NYSE: NCR) is a technology company specializing in solutions for the retail and financial industries
network model	The network model is a database model conceived as a flexible way of representing objects and their relationships
network protocol	a network protocol is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints
NTFS	“New Technology File System” is the standard file system of Windows NT and its descendants
Oracle	one of the major companies developing database management systems (DBMS)
Primary Key	a primary key is a candidate key to uniquely identify each row in a table
RAM	“Random Access Memory”, type of data storage used in computers
Red Hat Linux (7.1)™	was a popular Linux distribution assembled by Red Hat until the early 2000s, when it was discontinued
Relational model	database model based on predicate logic and set theory
Result Set	set of rows from a database, as well as meta-information about the query such as the column names, and the types and sizes of each column
Runtime	describes the operation of a computer program, the duration of its execution, from beginning to termination
Shell	Instances of this class represent the "windows" which the desktop or "window manager" is managing
SQL-Statement	refers to the uniqueness of data values contained in a particular column (attribute) of a database table
SureMark 4610™ printer	Thermal receipt printer manufactured by IBM
SurePOS™ 500	POS computer series with touchscreen manufactured by IBM
SVGA	“Super Video Graphics Array”, a broad term that covers a wide range of computer display standards
Swing	a GUI toolkit for Java™

SWT	“Standard Widget Toolkit”, a graphical widget toolkit for the Java™ platform originally developed by IBM and maintained now by the Eclipse Foundation
TCP/IP	Transmission Control Protocol / Internet Protocol (TCP/IP) is one of the core protocols of the Internet protocol suite
Thread	is a sequence of instructions which may execute in parallel with other threads
touch screen	display overlays which have the ability to display and receive information on the same screen
TrueType-fonts	outline font standard developed by Apple Computers
UltraATA-100	Advanced Technology Attachment (ATA) is a standard interface for connecting storage devices such as hard disks and CD-ROM drives inside personal computers
UNIX®	is a computer operating system originally developed in the 1960s and 1970s by a group of AT&T employees
UPOS	or UnifiedPOS is a retailer-driven initiative to combine two existing device interface standards under one specification to allow retailers freedom of choice in the selection of Point of Service devices
USB	Universal Serial Bus (USB) is a serial bus standard to interface devices
VIA Technologies	a Taiwanese manufacturer of integrated circuits
Visual Editor	Development platform supplying frameworks for creating GUI builders
VPN-tunnel	virtual private network (VPN) is a private communications network often used by companies or organizations, to communicate confidentially over a public network

8.2 Source Code

8.3 Acknowledgement

Declaration

We declare within the meaning of section 25(4) of the Examination and Study Regulations of the International Degree Course Information Engineering that: this Bachelor Thesis has been completed by ourselves independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or works of others are made known through the definition of sources.

Hamburg, 05.06.2007 **Hamburg, 05.06.2007**

---------- **----------**

Julius Schwarzweller **Fleming Kahn**

8.4 Team work division

Chapters

1, 2.1, 2.2, 3.1, 3.2, 3.3, 3.4, 3.5, 4.1, 4.2, 4.3, 4.4, 5.1, 5.2, 5.3.1, 5.3.2, 5.4, 5.6

are written by Mr. Julius Schwarzweller

Chapters

2.3, 3.6, 3.7, 3.8, 3.9, 3.10, 4.5, 5.3.3, 5.3.4, 5.3.5, 5.5, 5.7, 6

are written by Mr. Fleming Kahn