Geschwister-Scholl-Schule FIU1

Bismarckstr. 207

51373 Leverkusen

**Projektdokumentation**

**Erstellen einer Buchungssoftware für das Busunternehmen Sonnenschein**

Verfasser: Thomas Trinklein, Philipp Willms

Fach: Anwendungsentwicklung

Lehrer: Dr. Genc

Bearbeitungszeitraum: 24. Februar 2012 – 16. März 2012

Abgabetermin: 16.03.2012

Inhaltsverzeichnis

[Kick-Off-Meeting 1](#_Toc319482946)

[Projektauftrag 1](#_Toc319482947)

[Lastenheft 1](#_Toc319482948)

[Arbeitspakete 1](#_Toc319482949)

[Projektablaufplan 1](#_Toc319482950)

[Klassendiagramm vor der Implementierung 1](#_Toc319482951)

[Literaturverzeichnis 1](#_Toc319482952)

[Erklärung 1](#_Toc319482953)

## Kick-Off-Meeting

Im Kick-Off-Meeting am 28.02.2012 ist festgelegt worden, dass der Source Code über einen Subversion-Server geteilt wird. Außerdem wurde der Projektleiter ausgewählt. Weiterhin wurde die Arbeit auf die Mitarbeiter verteilt. Philipp Willms ist für die Utils zuständig und Thomas Trinklein ist für die Models zuständig. Die Klasse Reiseverwaltung wurde von Thomas Trinklein eingepflegt und in Kooperation mit Philipp Willms verbessert und verändert.

|  |  |
| --- | --- |
| Projektauftrag | |
| Projektname | BSFBS – Bus-Software für das Busunternehmen Sonnenschein |
| Projektleiter | Philipp Willms |
| Projektanlass | Das Busunternehmen wünscht eine Buchungssoftware, mit der Reisen und Kunden einfach verwaltet werden können |
| Projektziele | Siehe Lastenheft |
| Zu erarbeitende Ergebnisse | Funktionsfähige Software zur Buchung von Städtereisen gemäß den Anforderungen im Lastenheft |
| Randbedingungen | * Es werden vier Reisen angeboten * Diese Reisen finden in den Sommerferien statt * Die Busse fahren drei Wochen lang im Wochenrhythmus * Jeder Bus hat einen festgelegten Starttag |
| Termine und Meilensteine | 24. Februar: Projektstart und Kick-Off Meeting  02. März: Festlegen der Klassen  09. März: Testfälle aufstellen und durchlaufen  16. März: Die Software wird den Busunternehmen vorgestellt |
| Unterschriften | Auftraggeber Auftragnehmer  (Name, Unterschrift) (Name, Unterschrift) |

## Lastenheft

* Es wird eine Software für einen PC-Arbeitsplatz benötigt, über die die Belegung der vier Busse möglich ist. Eine Platzreservierung wird nicht gewünscht.
* Für jeden Bus sollen die Anzahl der zur Verfügung stehenden Plätze angezeigt werden.
* Eine neue Buchung soll automatisch mit einer neuen Buchungsnummer versehen werden.
* Eine Überbuchung soll nicht möglich sein. Dies soll durch eine Fehlermeldung angezeigt werden.
* Alle Teilnehmer einer Reise sollen in alphabetischer Reihenfolge aufgelistet werden können.
* Es soll eine Möglichkeit zur Stornierung einer Buchung bestehen. Auch dieser Vorgang soll automatisch einer fortlaufenden Nummer zugeordnet werden.
* Eine Unterdeckung soll verhindert und durch eine Fehlermeldung angezeigt werden.
* Mögliche Eingabefehler in den Buchungsdaten sollen korrigiert werden können.
* Alle Buchungen und Stornierungen sollen in einer log-Datei gespeichert werden.
* Die Daten sollen mit Excel und Word weiter verarbeitet werden und dazu im CSV-Dateiformat abgespeichert und geladen werden.

## Arbeitspakete

Modellierung von Klassen:

Thomas Trinklein

Implementation der Nutzerinteraktion:

Philipp Willms

Thomas Trinklein

Dokumentation:

Thomas Trinklein

Philipp Willms

Testen:

Philipp Willms

Datenverwaltung:

Philipp Willms

## 

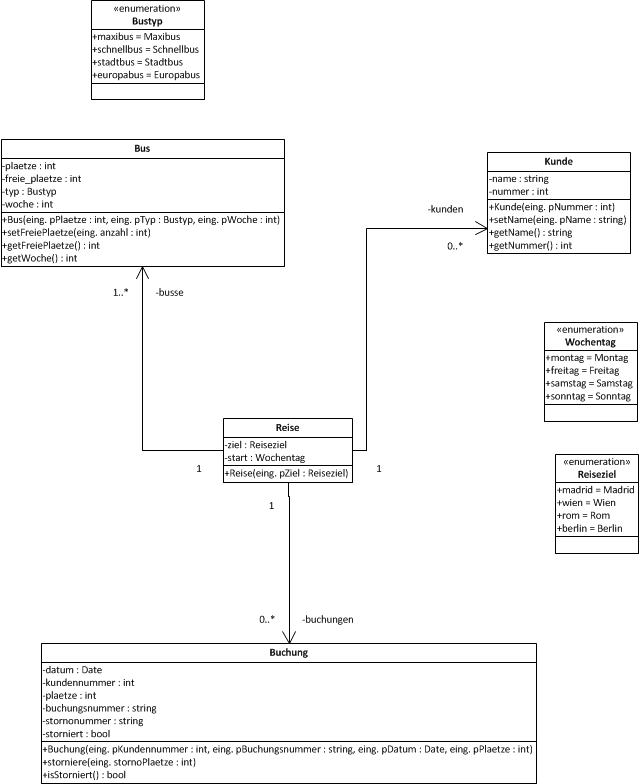
## Projektablaufplan

|  |  |  |  |
| --- | --- | --- | --- |
| Vorgangsnummer | Vorgangsbezeichung | Dauer (Tage) | Vorgänger |
| 1 | Erstellung eines Klassendiagramms | 1 | - |
| 2 | Implementation der Klassenfunktionalität | 7 | 1 |
| 3 | Implementation der Datenverwaltung | 8 | 1 |
| 4 | Javadoc | 2 | 2,3 |
| 5 | Testprotokoll erstellen und durchführen | 2 | 4 |
| 6 | Projektdokumentation | 2 | 5 |

## 

## Klassendiagramm vor der Implementierung

Vor der Implementierung wurde dieses Klassendiagramm angelegt:



Das gesamte Programm wird später über die Reise gesteuert. Hierbei hat jede Reise null bis unendlich Buchungen, solange noch Platz im Bus ist. Außerdem hat jede Reise einen bis drei Busse, die an verschiedenen Starttagen losfahren, was in der Enumeration „Wochentag“ verwaltet wird. Weiterhin haben die Busse ein unterschiedliches Reiseziel, welches in der Enumeration „Reiseziel“ verwaltet wird. Außerdem gibt es verschiedene Bustypen, welche in der Enumeration „Bustyp“ aufgelistet sind.

Zu jeder Reise gibt es auch Kunden, jedoch gibt es maximal so viele Kunden, wie die Reise Plätze zur Verfügung stellt.

Der Konstruktor „+Bus(eing.pPlaetze : int, eing.pTyp: Bustyp, eing.pWoche : int);“ initialisiert die Klasse „Bus“.

Der Konstruktor „+Kunde(eing.pNummer: int);“ initialisiert die Klasse „Kunden“.

Der Konstruktor „+Reise(eing.pZiel: ReiseZiel);“ initialisiert die Klasse „Reise“.

Der Konstruktor „+Buchung(eing.pKundennummer: int, eing.pBuchungsnummer: String, eing.pDatumg: Date, eing.pPlaetze: int);“ initialisiert die Klasse „Buchung“.

Die Methode „+storniere(eing.pStornoPlaetze: int);“ storniert eine gewisse Anzahl an Plätzen.

Die Methode „+isStorniert(): bool“ findet heraus, ob die Buchung bereits storniert wurde.

Die anderen Methoden sind Getter- und Setter-Methoden, welche Schreib- und Lesezugriff auf die privaten Attribute ermöglichen.

## Projektdurchführung

### Beschreibung zur Verwaltung der Daten

Es gibt bei diesem Projekt grundsätzlich zwei relevante Ansammlungen von Daten.

Einerseits müssen die Kundendaten (Kundennummer, Name, Vorname, Adresse, Telefonnummer) in einer Art „Kundenstamm“ abgelegt werden. Somit ist gewährleistet, dass, wenn mehrere Buchung zu demselben Kunden durchgeführt werden, dieser nicht immer neu angelegt wird.

Die Stammdaten werden in einer CSV-Datei abgespeichert. Bei der Implementation erwies sich als das komfortabelste, da nicht nur ein einfacher Schreib- und Lesezugriff möglich ist, sondern die Daten obendrein mithilfe der Software **Microsoft Excel** angezeigt werden können.

Letzteres sieht dann konkret für die Datei „Bussoftware\_Kunden.csv“ so aus:

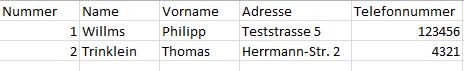


Abbildung 1: Datenaufbereitung im Kundenstamm

Andererseits müssen alle Buchungs- und Stornierungsvorgänge zu den jeweiligen Reisen verwaltet werden. Dies dient insbesondere dazu, bei jedem Programmstart alle bereits durchgeführten Buchungen und Stornierungen zur Verfügung zu haben. Dadurch bleibt die Belegung der Plätze in den Bussen immer auf dem neuesten Stand.  
Auch für diesen Fall wird eine CSV-Datei angelegt. Darin werden die Vorgänge mit ihren jeweiligen Nummern, dem zugehörigen Kunden und den Reisedaten (Ziel, Woche, Plätze) geschrieben.

Der folgende Screenshot der Datei „Bussoftware\_Log.csv“ soll dies veranschaulichen. Anhand einer solchen detaillierten Auflistung kann das Programm die freien Plätze eines Busses auslesen und alle Teilnehmer einer Reise ausgeben.

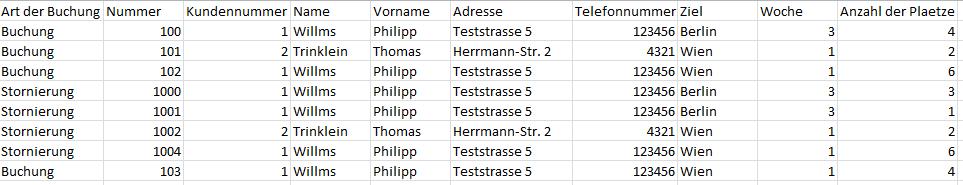


Abbildung 2: Logdatei zur Buchungssoftware

### Suchverfahren – Lineare Suche

Für den Suchvorgang nutzen wir die lineare Suche, da sie einen entscheidenden Vorteil in unseren Fall gegenüber der binären Suche bietet.

Die Daten müssen bei der linearen Suche nicht geordnet vorliegen, wodurch wir bei einer Datenbank viele Rechenschritte sparen, wenn wir die Daten erst sortieren müssen. Da die Datenbank nicht allzu groß wird, ist eine Sortierung und eine anschließende Suche mit der binären Suche nicht erstrebenswert, da der Aufwand der linearen Suche bei einer kleiner Datenbank einen maximalen Suchaufwand von n aufweist, wobei n die Anzahl der Elemente in der Datenbank ist.

### Sortierverfahren - BubbleSort

Bei diesem Sortierverfahren werden bei jedem Durchlauf das aktuelle Element und sein Nachfolger verglichen. Wenn der Nachfolger kleiner ist, wird er mit dem aktuellen Element vertauscht. Dies wird solange wiederholt, bis ein Durchlauf keine Vertauschungen mehr aufweist.

Bei diesem Projekt war ein Sortierverfahren erforderlich, da das Busreiseunternehmen alle Teilnehmer einer Reise in einer lexikographisch aufsteigend sortierten Liste anzeigen lassen möchte. Dabei standen neben dem BubbleSort auch die Verfahren InsertionSort und SelectionSort zur Auswahl. Da im Endeffekt alle eine Laufzeit von besitzen, entschieden wir uns für den BubbleSort, da dieser schnell und unkompliziert zu implementieren war.

## Literaturverzeichnis

## Erklärung

Wir erklären, dass wir das Projekt ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benützt haben.

Bei Benutzung der elektronischen Quellen haben wir eine vollständige Zugangsadresse mit dem Datum im Literaturverzeichnis aufgelistet, an dem die Seite zuletzt von uns aufgerufen wurde.

……………………………., den …………………….. ………………………….

Ort Datum Unterschrift