Module 326

Your Gradebook

Design Phase

Tamanna Rajput, Anushya Nagulanandan
TBZ

Inhaltsverzeichnis

This Documentation

In this Documentation we have documented how we went on with the planning and have also attached everything that was needed or made. For the planning phase we mostly used the application word on a local laptop and the online application draw.io.

In this documentation you will find the following things attached:

- ➔ Project description
- ➔ Use cases (written and diagram)
- ➔ CRC Cards
- ➔ Domain Model
- ➔ Class diagram

Steps

1. Project Description

Before the planning phase we came up with an idea and also wrote a detailed description so that the application is easy to understand.

Our idea is to program an application called "Your Gradebook". In this application you can keep an overview of all your marks. If you go to two different schools (BMS/ABU) then you also have the option to work with more than one gradebook. But you can't look into two gradebooks at once. At the starting it asks for your name and school and opens exactly that gradebook. Then a small menu pops up which asks you if you want to add a mark or look at the overview. If you choose to add a mark then it first asks for a subject. In the gradebook you can add your subjects and name them however you want. There is no fixed number of subjects so you can add as many as you please. After choosing a subject you can a mark and also give in an evaluation for each mark, that means how much that mark weighs for your average. And of course, you can also delete a mark if needed. If you choose the option "overview" in the menu then it will show you all your subject and the average of your subjects this semester.

2. Use cases

As the use cases add value to a program we decided to write them. This helps us explain how the system should behave during the process. Our use cases are more user based. The use cases help us to get a list all of the steps a user can do. We didn't only write use cases but also made a use case diagram to visualize it all. This helped us to brainstorm what could go wrong. It helps us to get a better idea of how the user input would look like.

Our Use Cases

Use Case (1)	Choose Gradebook
Pre-Condition	The application should be started before this action can be done
Description of Use Case in detail	The user has to give in his/her name and which school he/she goes to
Post-Condition	After the action the system remembers the name and school and saves it in the background to load the correct data. There will be a menu that pops up with the options “Overview”, “Add mark” and “Exit”
Exceptions	If a grade book to the name and school is not present then it automatically creates a new gradebook or loads the existing gradebooks data in the background

Use Case (2)	See Overview
Pre-Condition	The user should have given his name and school before performing this action
Description of Use Case in detail	In the menu that will be shown the user can choose the option “Overview”
Post-Condition	After this action the system show the user a overview of all the existing marks and subjects
Exceptions	- No Exceptions here –

Use Case (3)	Add mark
Pre-Condition	The user should have given his name and school before performing this action
Description of Use Case in detail	In the menu that will be shown the user can choose the option “Add marks”
Post-Condition	After this action the user will see a list of existing subjects on the screen which the user can choose from.
Exceptions	In case the wanted subject isn’t there in the list there is also an option called “Create subject” where you can add a new subject to the list

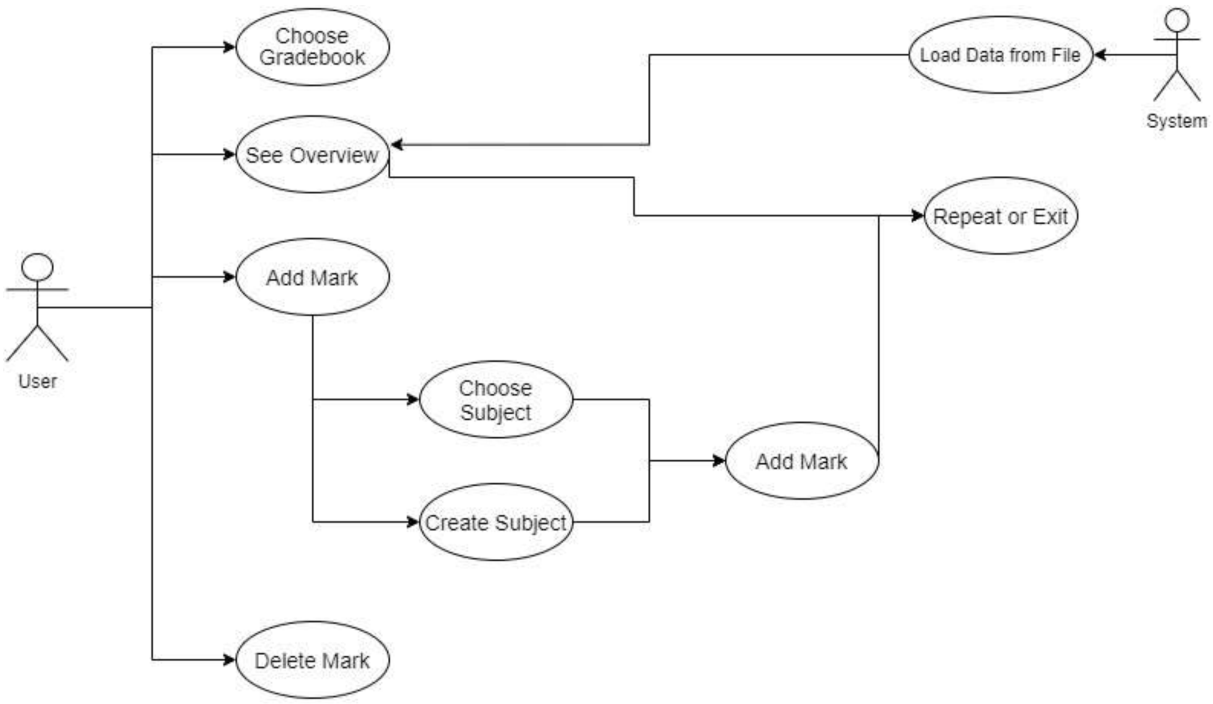
Use Case (4)	Create subject
Pre-Condition	The option “create subject” has to be chosen before this action can be done
Description of Use Case in detail	The user has to give in a name for the new subject
Post-Condition	The subject will be saved and then added to the list
Exceptions	If the user forgets to give in a name the message “Please type a subject name” will pop up

Use Case (5)	Choose Subject
Pre-Condition	The option “add mark” has to be chosen before this action can be done and will see a list of all the subjects
Description of Use Case in detail	The user can choose a subject now and just type in a mark and an evaluation he/she wants
Post-Condition	The mark to the correct subject will be saved in the data file
Exceptions	If the user forgets to give in a marl on evaluation the message “Please give in the needed data” will pop up

Use Case (6)	Delete mark
Pre-Condition	There should be an existing mark in the user’s gradebook before this action can be done
Description of Use Case in detail	The user can just choose the mark and then delete it
Post-Condition	The mark will no longer be shown on the screen and will also be deleted in the data file
Exceptions	- No Exceptions here -

Use Case (7)	Repeat or exit
Pre-Condition	The user should have completed the use case 2 or 5 to be able to finish this usecase
Description of Use Case in detail	The user will be asked if he wants to stay on this page or want to go to the menu
Post-Condition	According to the user choice the user will either stay on the page or go back to the menu
Exceptions	- No Exceptions here -

Our Use case Diagram



3. CRC Cards

We also wrote CRC Cards as it really helps us to get a better idea of how you can code later on. A Class Responsibility Collaborator model is basically a simple collection of index cards but are divided into three specific sections. Class Name, Responsibility and Collaborators. The Class Name is self-evident. It's the name of the class. The responsibility kind of explains what the class is capable of and has to do. The collaborators are a list of classes/interfaces which the current class uses in the future. You can easily call the CRC Cards a very superficial class diagram or a domain model.

Class Starter **Responsibilities** - Call all Methods	**Collaborators** - UserInput

Class UserInput	**Collaborators** - Starter - Calculator

Responsibilities - All User Inputs Name School Menu Selection Select/add Subject Name of test Marks Weight	- Subject - Marks - Serializer

Class	**Collaborators**
Calculater **Responsibilities** Calculating the weight Calculate Avrg Show Overview	UserInput Marks

Class	**Collaborators**
Subject **Responsibilities** - Saves all information of a subject	- UserInput - Marks - SerializbleMarks

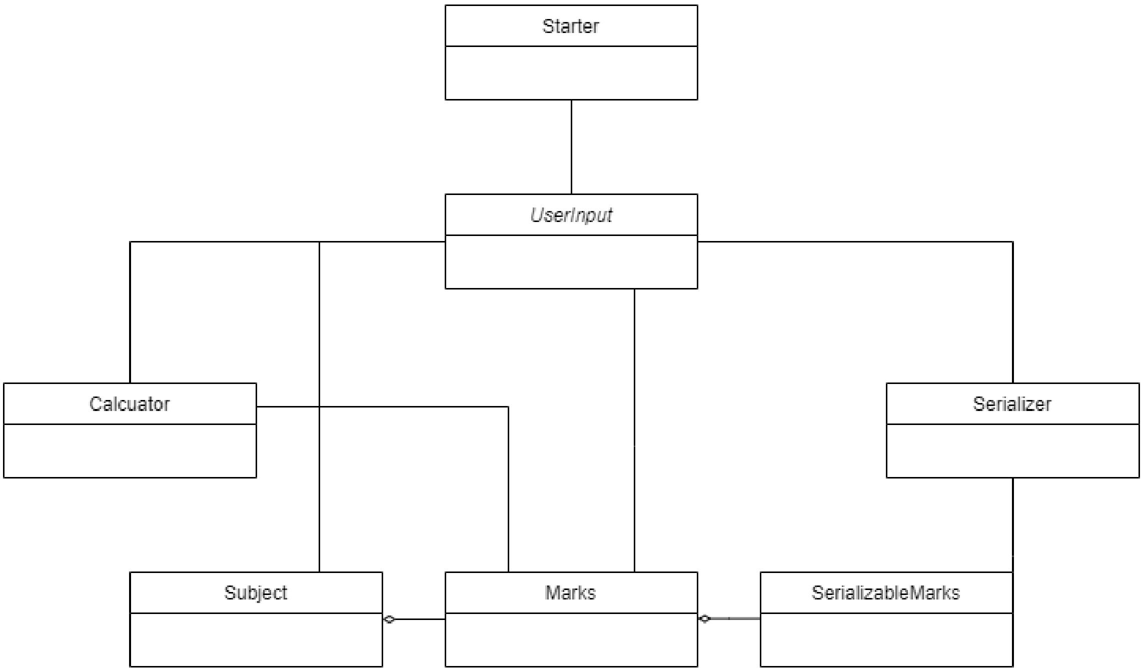
Class	**Collaborators**
Marks **Responsibilities** - Saves all information of a subject	- UserInput - Subject - SerializbleMarks

Class	**Collaborators**
Serializer **Responsibilities** - Serialize and deserialize Marks	- UserInput - SerializableMarks

Class	**Collaborators**
SerializableMarks **Responsibilities** - Adds all Information to a List - This class is serializable	- Serializer - Marks - Subject

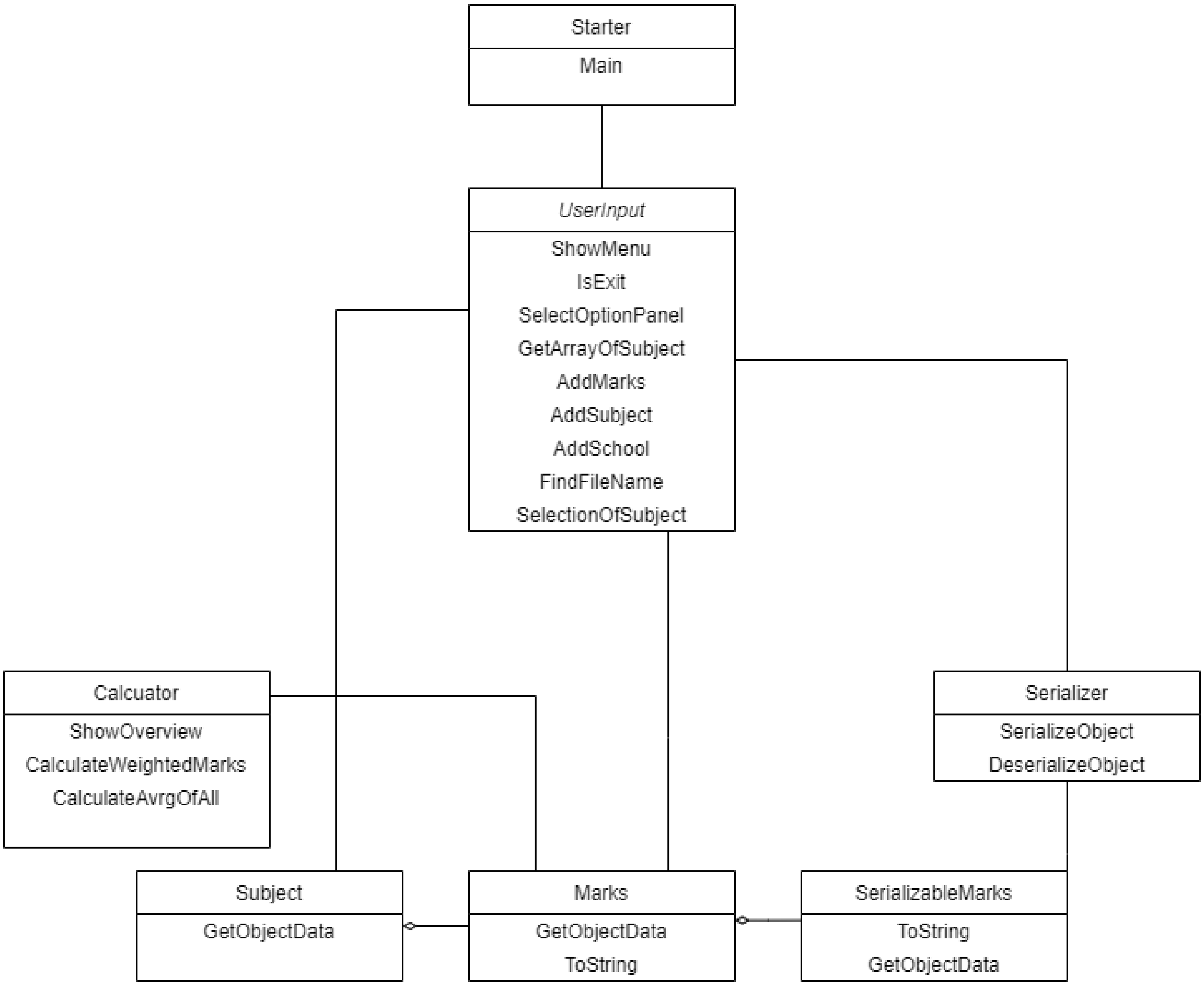
4. Domain Model

The Domain Model was also a very important part in our planning phase. We did this based on our CRC Cards. The domain model basically is a structured visual representation of our classes or objects in a domain of interest. You can see the domain objects or the conceptual classes. The associations between the classes is a very important part. We tried our best visualizing everything so that in the future we won’t struggle much with the coding part.



5. Class Diagram

A class diagram helps us map out the structure of our system. We model classes, object, fields and methods. The relationship between classes is also represented. First, we made a class diagram how we think we want to implement it. The class diagram helped us think about a lot of things that we didn't realize before. We had a good preview of our first class diagram and our system was represented well. Of course, for the class diagram we used the existing CRC Cards and the domain Model which helped us be faster and more detailed.


Starter
Main
UserInput
ShowMenu
IsExit
SelectOptionPanel
GetArrayOfSubject
AddMarks
AddSubject
AddSchool
FindFileName
SelectionOfSubject
Calcuator
ShowOverview
CalculateWeightedMarks
CalculateAvrgOfAll
Serializer
SerializeObject
DeserializeObject
Subject
GetObjectData
Marks
GetObjectData
ToString
SerializableMarks
ToString
GetObjectData