**SDJ 2 Assignment 1**

**Team 4**

Table of Contents

[1 Introduction 4](#_Toc513101604)

[2 Requirements 4](#_Toc513101605)

[2.1 Functional requirements 4](#_Toc513101606)

[2.2 Nonfunctional requirements 4](#_Toc513101607)

[3 Time schedules 5](#_Toc513101608)

[3.1 Burndown Chart 6](#_Toc513101609)

[4 Class Diagram 7](#_Toc513101610)

[5 Implementation 7](#_Toc513101611)

[5.1 Design patterns 7](#_Toc513101612)

[5.2 Code implementation 9](#_Toc513101613)

[6 Testing 10](#_Toc513101614)

**Abstract**

This document’s aim is to inform the reader about the different methods and techniques used to fulfill the main objectives for this project. For this case the objectives solved where linked to the business case given by Vipassana which were struggling with their daily activities given that they had limited technology in use.

As a result, different technical choices where made to produce a satisfactory outcome for the given report. This document will proceed with a small description of the current client and the struggles they face in their organization.

Based on an interview taken at the costumers headquarters a set of requirements have been subtracted and are crucial to get a better understanding of the client’s needs.

# 1 Introduction

Vipassana is a spiritual center that focuses on Buddhist spiritual practices and insight awareness.

The company does not practice any religious beliefs but rather focus their attention on dream interpretation and healing practices. They are responsible for organizing different types of events and adding new members as they sign up. Vipassana is also holding lectures about certain topics where members can also register for.

The current system of keeping track of all the mentioned events is slow and inefficient to say the least, relying on pen and paper to execute and it lacks reliability since sometimes employees misplace certain records and forget where they stored them originally.

As up lately some members have not been paying their membership fee which is causing a financial deficit for the company. For that reason, Vipassana would like a solution for this problem which can inform them which member has not played their fees.

The client mentioned in the interview case that they would like it if this project will focus on a client server approach since this would allow users to see directly and efficient the list of their debtors.

The project planning method used for this report is SCRUM and in the following section the project backlog or requirements shall be presented.

# 2 Requirements

Based on the interview case with Vipassana, a set of requirements have been taken out and are split in two categories: functional and nonfunctional.

As mentioned earlier these requirements are crucial for this step since they provide a good understanding of the costumer’s needs and will help in the development of this assignment.

## 2.1 Functional requirements

* The system must allow the employees to list all members.
* The system must allow employees to search for members that have not paid their membership fee.

## 2.2 Nonfunctional requirements

* The system must provide a client server connection.
* At least 4 java design patterns must be implemented.

# 3 Time schedules

For this report the managing method chosen is called SCRUM and it provides an agile way of organizing the work load.

The work for this report has been conducted in a team of 5 people and as such tasks had to be split accordingly. For that reason and the fact that there was a 2-week deadline to complete, a time schedule has been created to organize the activities in an efficient manner.

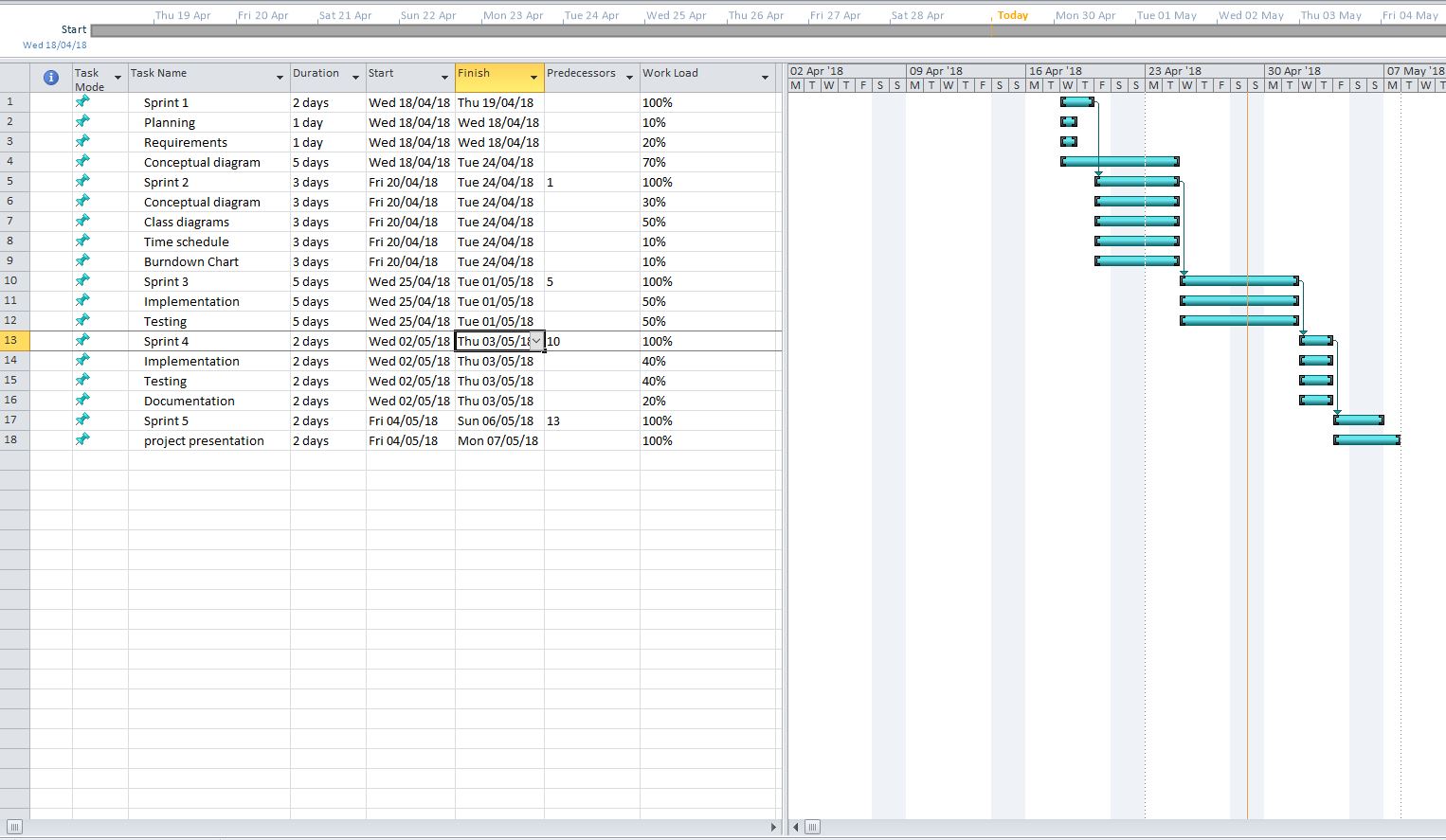
The assignment started on the 18th of April 2018 and ended on 4th of May 2018.

Due to such a short time period 4 sprints where created and instead of just having one for each week the group had 2 every Wednesday and Friday for the given time. A fifth sprint has also been created but it only serves as a meeting where the presentation for this report will be made and is not affected by the deadline.

Each of the 4 sprints have been given several requirements to complete, some being higher priority than others thus having a bigger percentage of time allocated for their completion.

A record of these sprints has been taken and is presented in form of a Gantt Chart. The Chart was an essential tool that helped organize and keep track of each task of for the given assignment.

***Figure 1 Gantt Chart for SDJ Assignment 1***



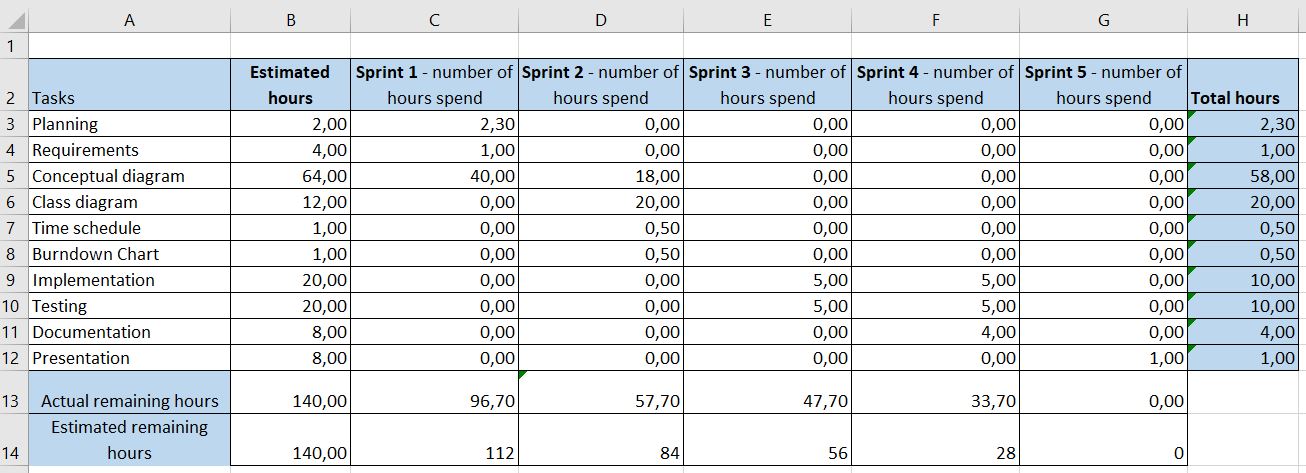
## 3.1 Burndown Chart

In this section of the chapter the Burndown Chart shall be discussed. This tool is essential to keep track of the schedule established earlier and informs the team if it is behind or ahead schedule.

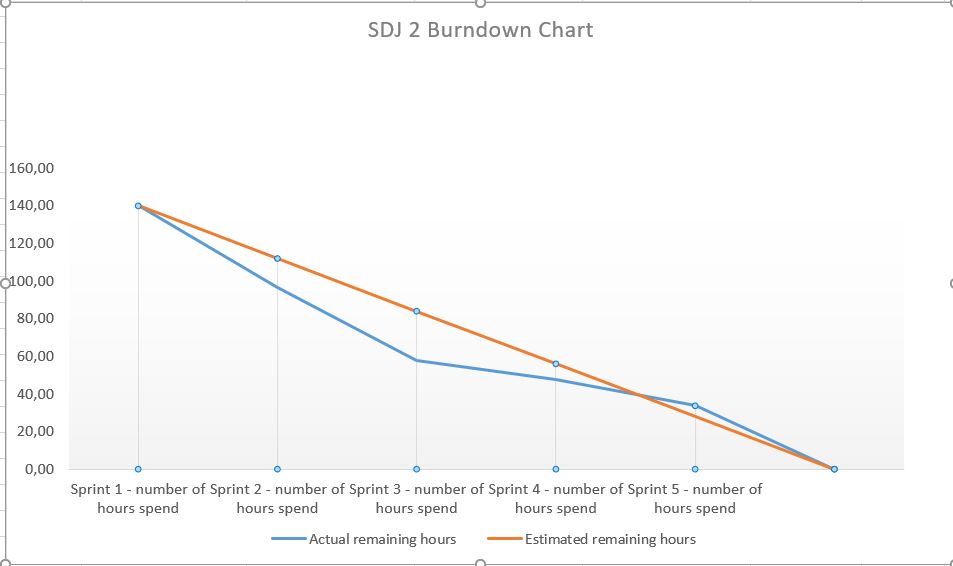
The chart is based on each task in each sprint for 5 sprints in total. It has two main components which consist of estimated time given for an individual task and an actual time spent for it.

This helped the team to reorganize the time allocated for this project in case it fell behind on certain activities or plan forward if it was ahead.

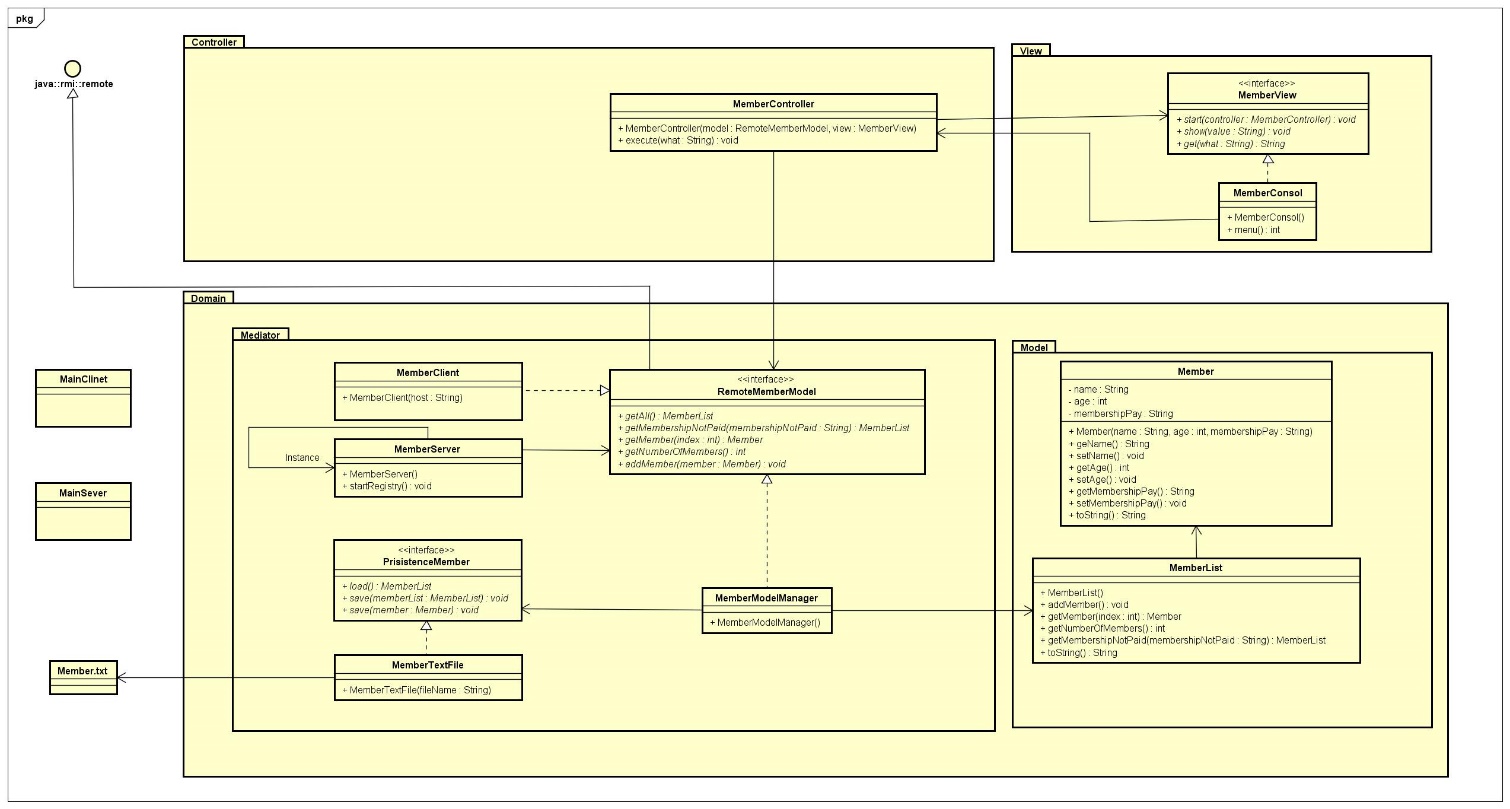
***Figure 2 Burndown Chart Table***



***Figure 3 Burndown Chart***



# 4 Class diagram



The class diagram consist of 3 main packages: Domain, View and Controller. To make it easier for the developer and to simplify coding the domain package is divided into two more packages: Model and Mediator. These packages were created following MVC design pattern.

Explanations of all the classes with methods would be in next paragraphs,

Member

This class provides a constructor which creates member objects. This class provides methods: getName, getAge, getMembershipPay and etc.

MemberList

This class provides methods an ArrayList to which multiple members can be added. In addition this class provides methods for adding member, getMembershipNotPaid, getNumberOfMembers and etc.

MemberModelManeger

This class is responsible for initializing the load() method to retrieve information regarding members.

RemoteMemberModel

This interface contains the following methods: getAll(), getNumberOfMembers(),getMembershipNotPaid(), getMember(), addMember()

PersistanceMember

This interface gathers information from the user and stores that information in a ArrayList and stores that information on a .txt file.

MemberTextFile

This class is responsible for creating a .txt file which holds information regarding created member objects. It has a method load(), with which the user can retrieve member information from the .txt file.

MemberServer

This class is responsible for creating a server on the PC on which the source code was started on. It has a non-argument constructer that has a try-catch clause to avoid exceptions when starting the server. It has override methods from the MemberList class: getAll(), getMamebershipNotPaid()

MemberClient

This class is used to connect to the server. It uses a non-argument constructer and a try- catch clause to avoid exception when running the source code. It has some inherited methods like: getAll(), getNumberOfMembers(),getMembershipNotPaid(), getMember(), addMember();

MemberConsol

This class provides methods that start when the server has started.

MemberView

This interface contains the following methods: start() and show();

MemberController

This class provides instruction on what happens when a certain input is received.

It has a method execute() which uses cases to help guide the user through the system’s interface.

# 5 Implementation

In this chapter the discussion will be about the 4 design patters chosen to create the actual code needed for the client server application.

## 5.1 Design patterns

For this report the following design patters have been chosen:

* Singleton
* MVC
* Adapter
* Façade

**Singleton** is a design pattern which ensure that a java class has only one instance and provides a global access for it.

***MVC* or model view controller** is a design pattern primarily used for the construction of a GUI and it is composed of 3 parts.

*Model*represents the actual data that exists, and the different rules used for its interaction.

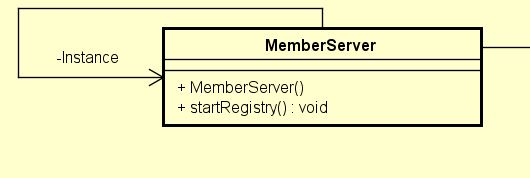
*View* shows the content of the model and tells how the data should be shown.

*Controller* creates a link between the user and the view in which the model executes the required operations.

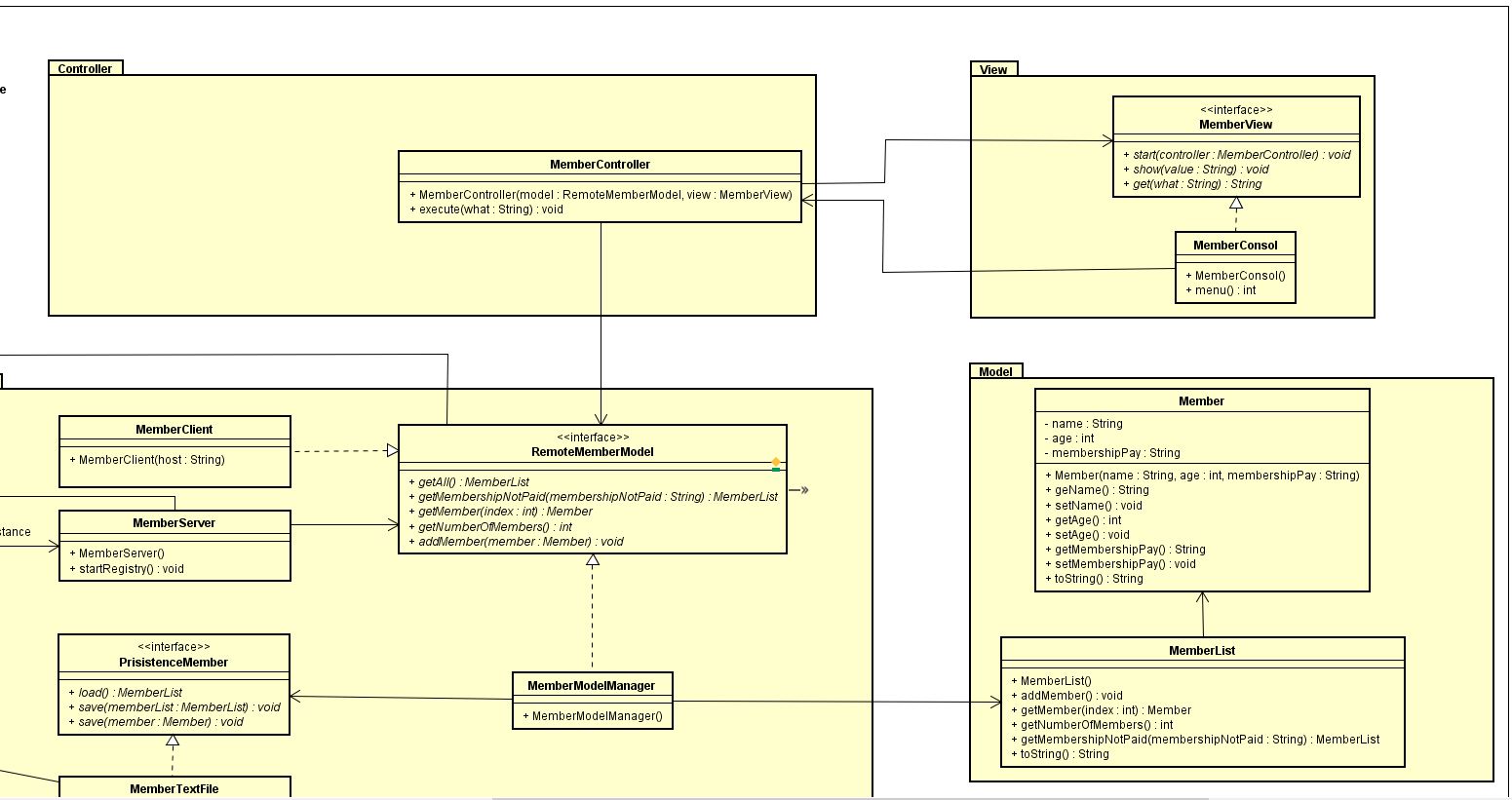
**Adapter** another design pattern that was chosen for this project. The adapter helps in the connection of two separate interfaces which without it would not be able to connect otherwise due to incompatibility issues.

**Façade** is the final design pattern used in this report. The façade design pattern is a wrapper interface that connects multiple interface together to help the client’s application interact with the system.

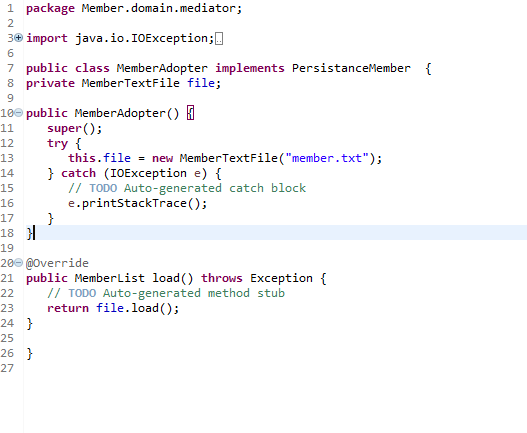
***Singleton example***



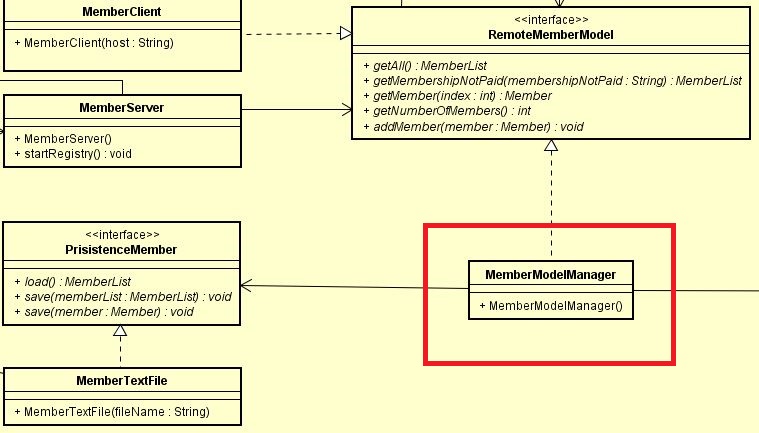
***MVC example***



***Adapter example***



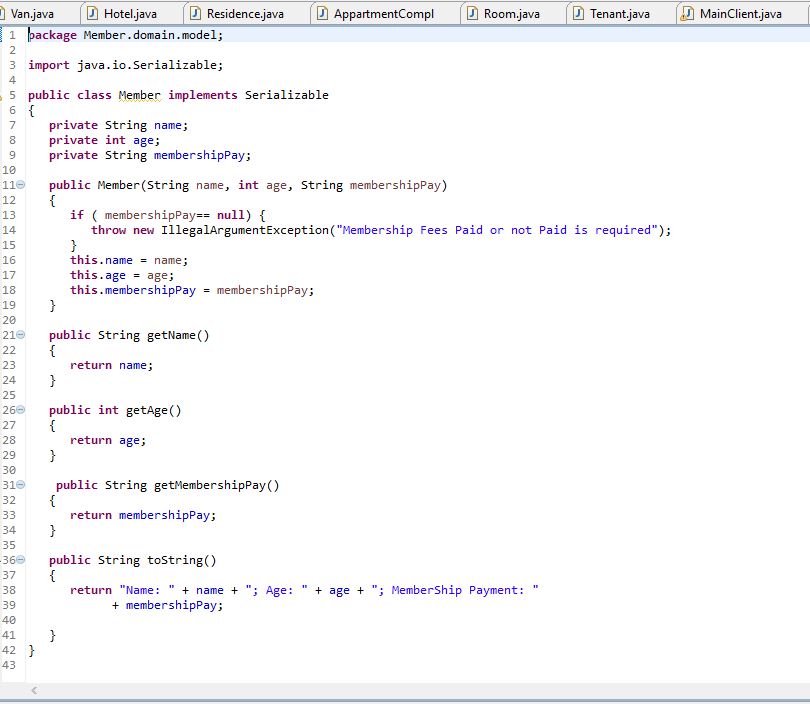
***Façade example***



## 5.2 Code implementation

In this section of the project a part of the code will be displayed and explained to get a better understanding on how the system works and how the client side is implemented in java.

***Figure 4 Member class***



The first thing that can be observed in this class on line 5 there is implements Serializable. This is used to convert objects from this class into bytes thus compressing them and makes the transportation over a network much easier.

The member class is composed of a constructor which contains 3 arguments of name type string age of type integer and membershippay of type string.

The ***if statement membershippay==null*** states that if the membership has no value inserted it will display an illegal argument exception of Membership payed or not required for the operator. The class also contains get methods for each of the constructor’s arguments and returns the specific object.

And finally, the ***public string toString()*** methods is implemented to return all the member information needed like name, age and membership status.

# 6 Testing

In this chapter there will be a discussion about the way the code has been tested and what method was used to perform such a task. The method used to test the code is called j unit testing and has been implemented for the current code.

Testing is the documentation of the result to make sure it fulfils all the requirements. The system functionality has been tested on the base of the requirements. In Model package, Member and MemberList has been tested with unit root test and most of the classes have same methods as in Model. To check the functionality all methods have been tested by Junit test, but discussion is here on Get Members who did not paid membership fees. Test code should be as

public class MemberListTest {

Member member1 = new Member("Taha", 26, "paid");

Member member2 = new Member("Alex", 24, "not paid");

Member member3 = new Member("Oskar", 22, "not paid");

MemberList list = new MemberList();

@Test

public void testGetsMembershipNotPaid() {

list.addMember(member1);

list.addMember(member2);

list.addMember(member3);

assertEquals(member2, list.getMembershipNotPaid().getMember(0));

assertEquals(member3, list.getMembershipNotPaid().getMember(1));

}

Three members object are created and added in the list. AssertEquals is used to test the equality of expected and actual list. So member2 and member 3 are the expected member who did not paid, and the actual is list.getMembershipNotPaid(). So, unit root passes the test and it means that method is functioning.