Foundations of Software Engineering – Assignment 1

Gilad Chen - 207168568
Igor Alikin – 322081241

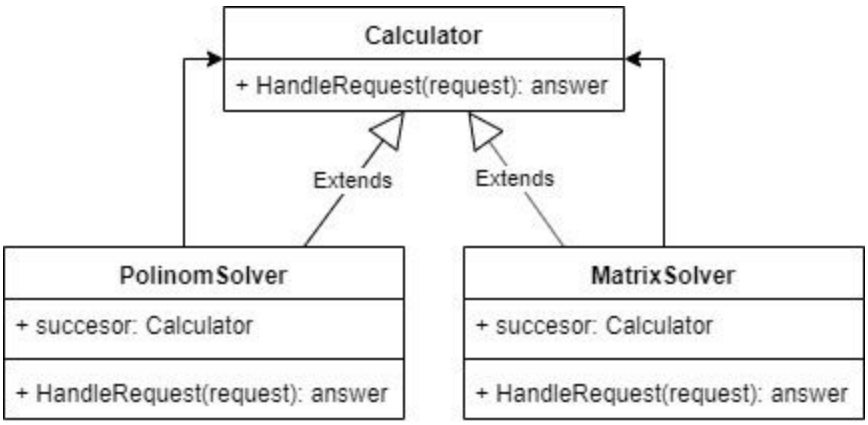
Question 1:

The code for this task is included with a readme. In the readme, the needed addition to the tests' tearDown() function is described.

Question 2:

a. Class "MatrixSolver" -> "PolinomSolver" ->"Calculator".

b.



Calculator - is the base class and he is the handler
Polinom and Matrix - are the concrete handlers

The "COR" design pattern solves our problem in this example by assigning responsibilities for each object and linking them in a chain, if one object can't handle some requests he will transfer it to the next one until it reaches the top object and he will decide if the request is manageable at all.

c.

Alike	Different
Forwarding requests, in decorator to the next class and in “COR” to the successor.	Decorator takes class and “wraps” him inside another class that gives him more responsibilities while in “COR” each class takes care of his own responsibilities
Both have flexibility in assigning responsibilities to objects	In the chaining process decorator does something and passes while in COR its or do or pass not both.

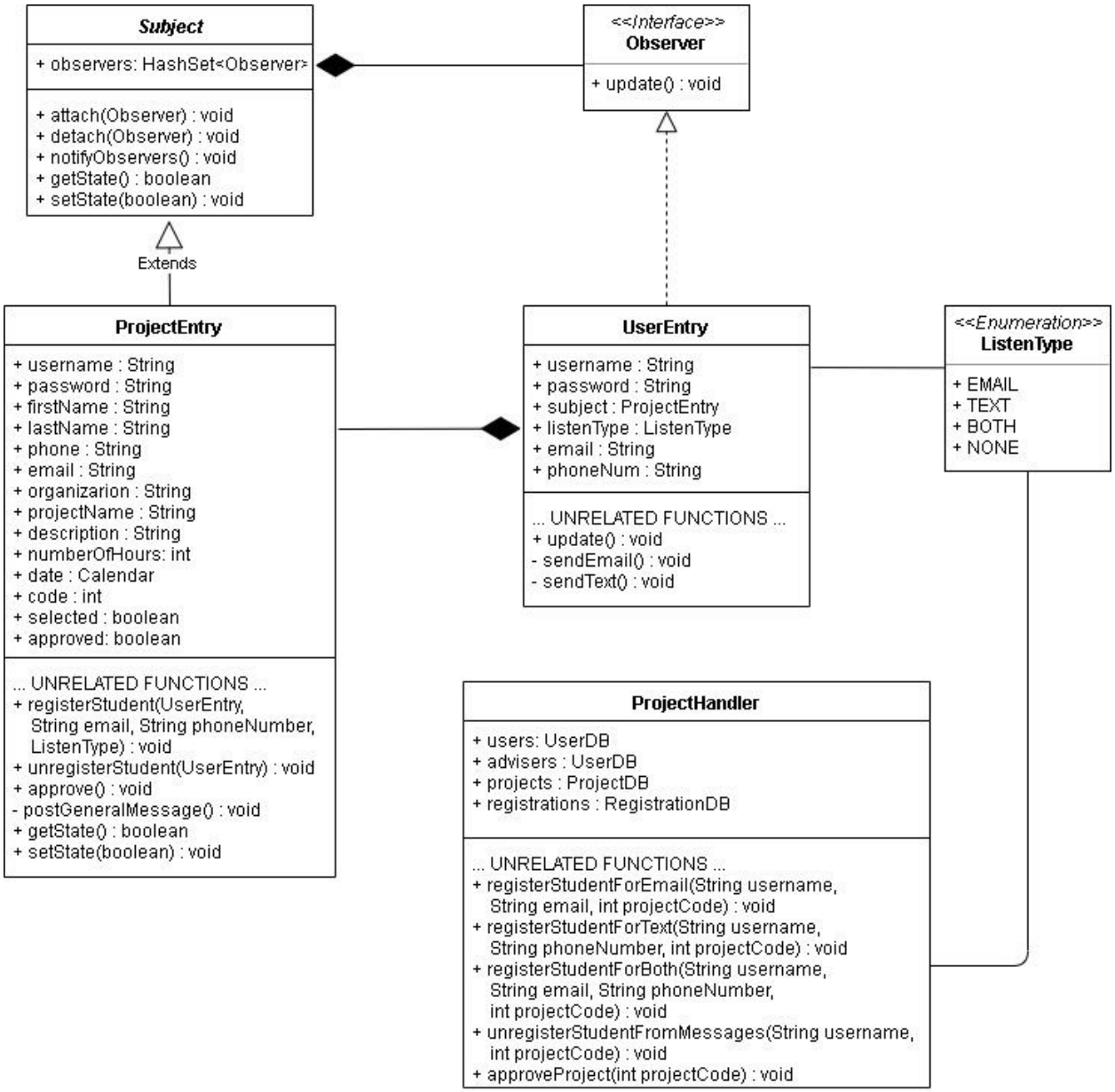
d.

Like the example from the class material where we want to show plain text in scrollable window and with borders. The example shows we decorate the plain text with scrolls and then decorate with borders. In “COR” it wont work because the classes have some connection between them while in “COR” only one object will be able to handle the request.

Question 3:

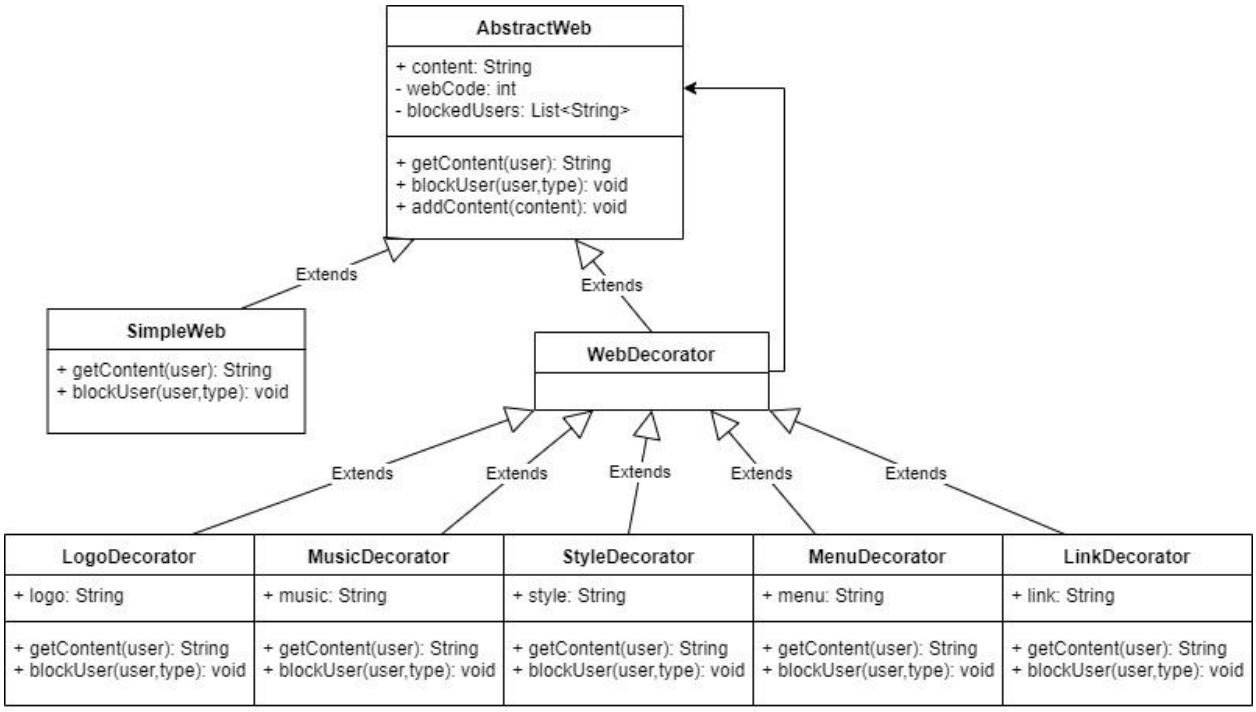
a. We will use the Observer design pattern.
In order to send messages to students who registered to a project's messaging list, there is need to notify them on the project's approval. This fits perfectly with the notifyObservers() method of the Observable.

b. Observer: UserEntry. Upon update a corresponding message will be sent to the user in accordance to the type of service he registered to.
Subject: ProjectEntry. A project will keep a list of all the observers and upon approval, notify all of them.
Class diagram:



Question 4:

a. We will use the Decorator design pattern, because we need to add functionalities for our web page and it fits exactly for the decorator pattern.
b. We will have the “Web” abstract class and the “SimpleWeb” will extend it, this will be our simple web page. Alongside “SimpleWeb” the “WebDecorator” will also extend the web. And we will have for each decoration option a class that will extend from the decorator.
c.



AbstractWeb - our interface for every web page.
SimpleWeb - our simple web page.
WebDecorator - interface for the decorators.
Decorators - Concrete decorators.