ELEARNING SYSTEM DESIGN USING SOFTWARE DESIGN PATTERNS

Submitted by:

Doha Basem

Submitted To

Dr.Abdelaziz A.Abdelahmid

12/27/2015

1.Creational Design pattern

The report is about a complete design for an e-learning system in which software design pattern strategy is followed in designing different modules in the system.The used design patterns in this system are as follows: Abstract Factory,Builder,Factory Method,Prototype,Singleton,Adapter,Bridge,Composite,Decorator,Façade,Flyweight, Proxy,

Chain of Responsibility ,Command,Interpreter,Iterator,Mediator,Memento,Observer ,State,Strategy,Template and Visitor Design Pattern .The system contains submodules ,so each of those submodules are described and designed using one of the above mentioned patterns .Along with the description of each submodule there is a UML diagram and any further illustrations if required.

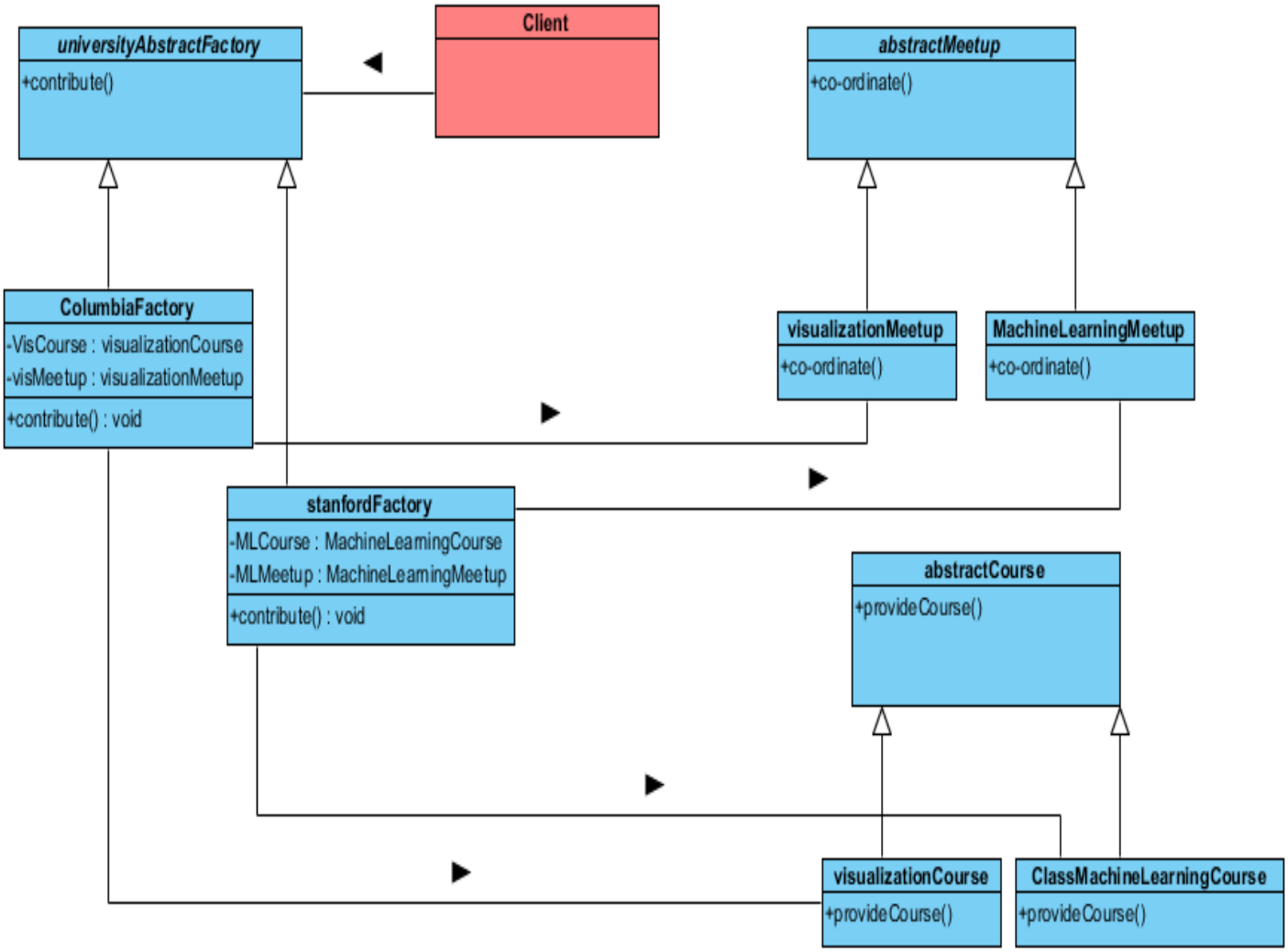
ELearning System Design using software design patterns

1.CREATIONAL DESIGN PATTERN

1.1|Abstract Factory Design Pattern

In the e-learning system to be designed ,many universities shall contribute to the portal with a set of services ,including courses and academic online meet ups .The system shall provide those services to the user ,so the proposed design for this module is one that uses the abstract factory design pattern in which the services are the products to be created and the universities are the abstract factories.

Design pattern components	System components
Abstract factory	universityAbstractFactory
Abstract product	abstractCourse/abstractMeetup
Concrete factory	stanfordFactory/columbiaFactory
Concrete product	visCourse/MLCourse/visMeetup/MLMeetup

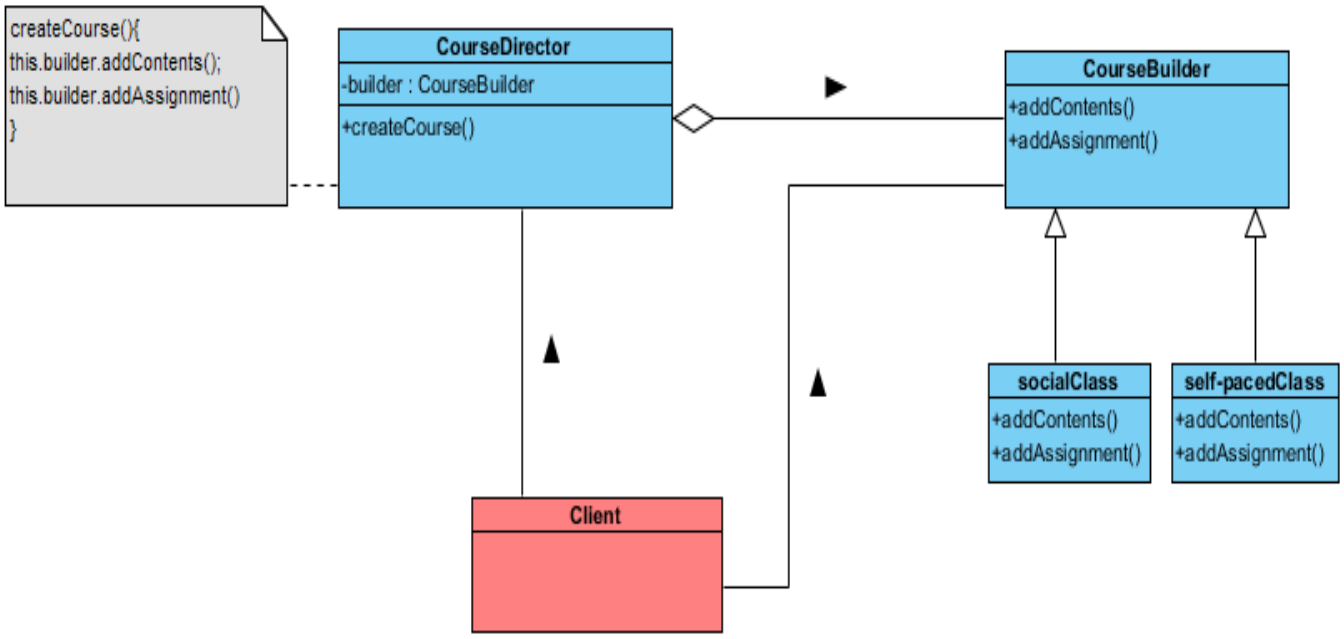


Code in project: abstractFactory.java

1.2|Builder Design Pattern

The system is planned to support different learning environments, including the social classes and the self paced classes .The construction process for each learning environment shall be the same,i.e. to create each class, the designer should add the contents and add the assignments .However the contents and the assignments shall be added differently for each class .So builder design shall be a got fit for designing such requirements ,as the system is required to build a complex object with different representations.

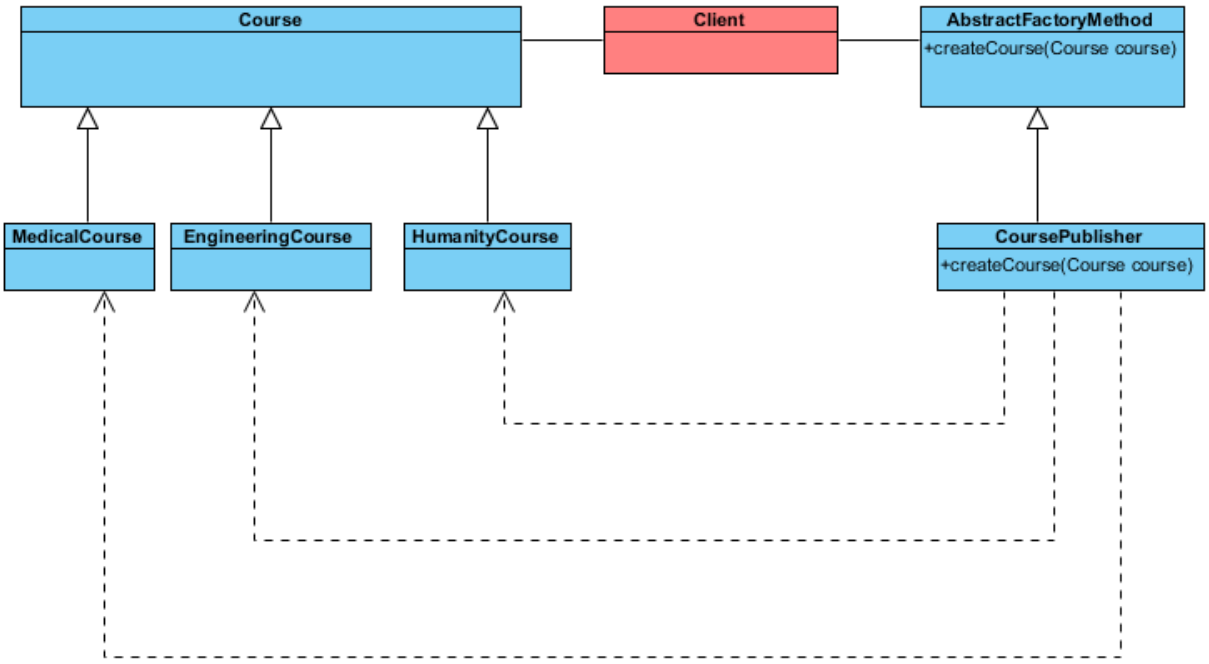
Design Pattern Components	System Components
Director	CourseDirector
Builder	CourseBuilder
Concrete builder	SocialClass/self-pacedClass



Code in project: builder.java

1.3|Factory Method Design Pattern

Factory method design pattern is used for the creation of courses. An object of the required course is passed to the factory method as a parameter and accordingly the course is created.

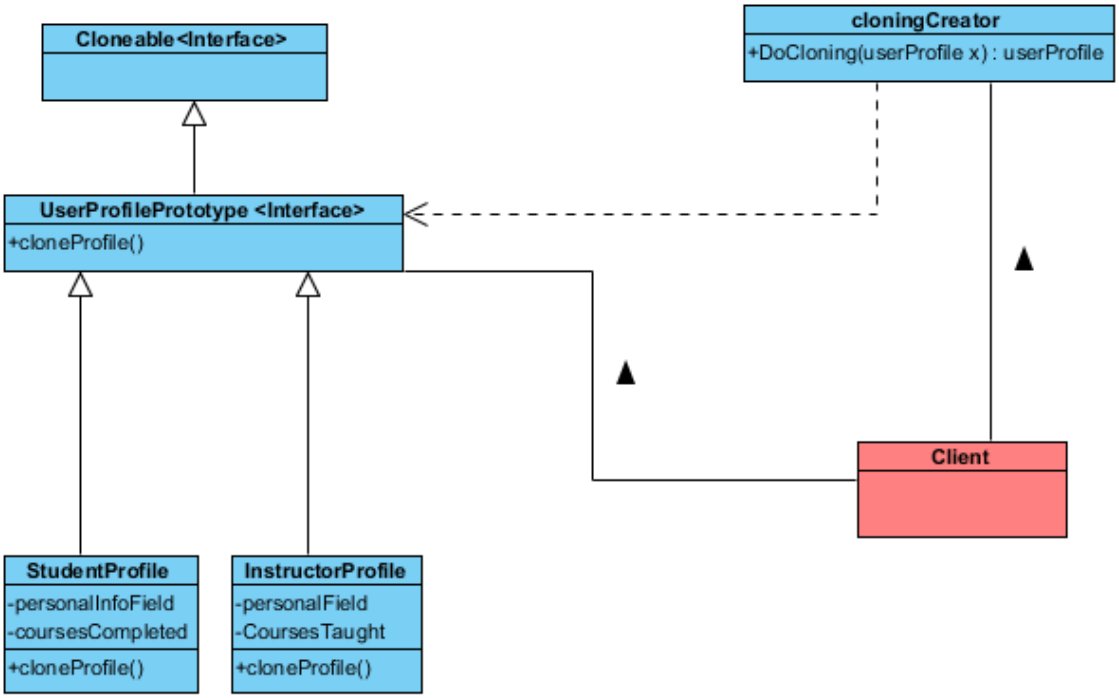


Code in project :factoryCreator.java

1.4|Prototype Design Pattern

To personalize the interaction between the users, each user has his profile that will act as an interface for communication with other users.

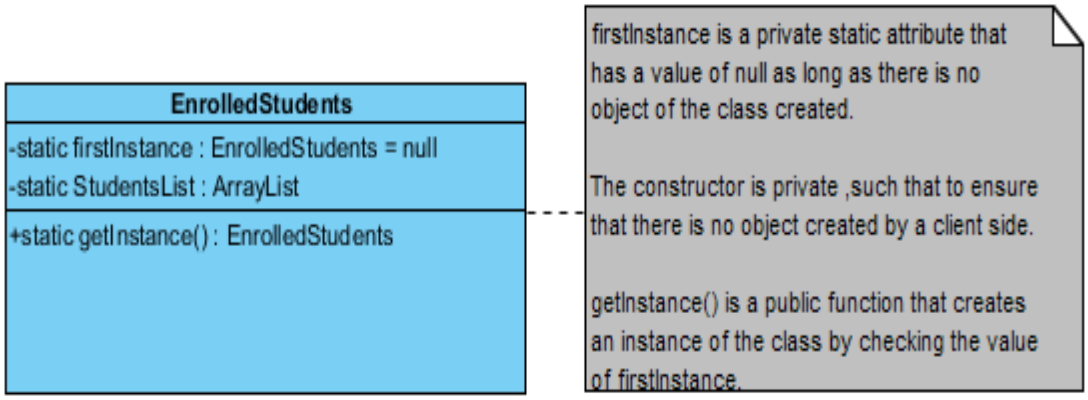
All the learners' profiles have the same design , and all teachers profiles also have the same design . So a good fit for the design of a module that creates users' profiles shall be the prototype design pattern ,in which only one profile is created and the other profiles are just clones for that profile .So instead of having many profile objects in the system ,there is only one object for each profile type .



Code in project :prototypePattern

1.5|Singleton Design Pattern

In the system there should be a single list that holds all the users with some attributes required for further analysis . Analysis is done only on this list,so in the design we should ensure that there is a single list created that contains all the relevant information .So the singleton deign pattern is used.

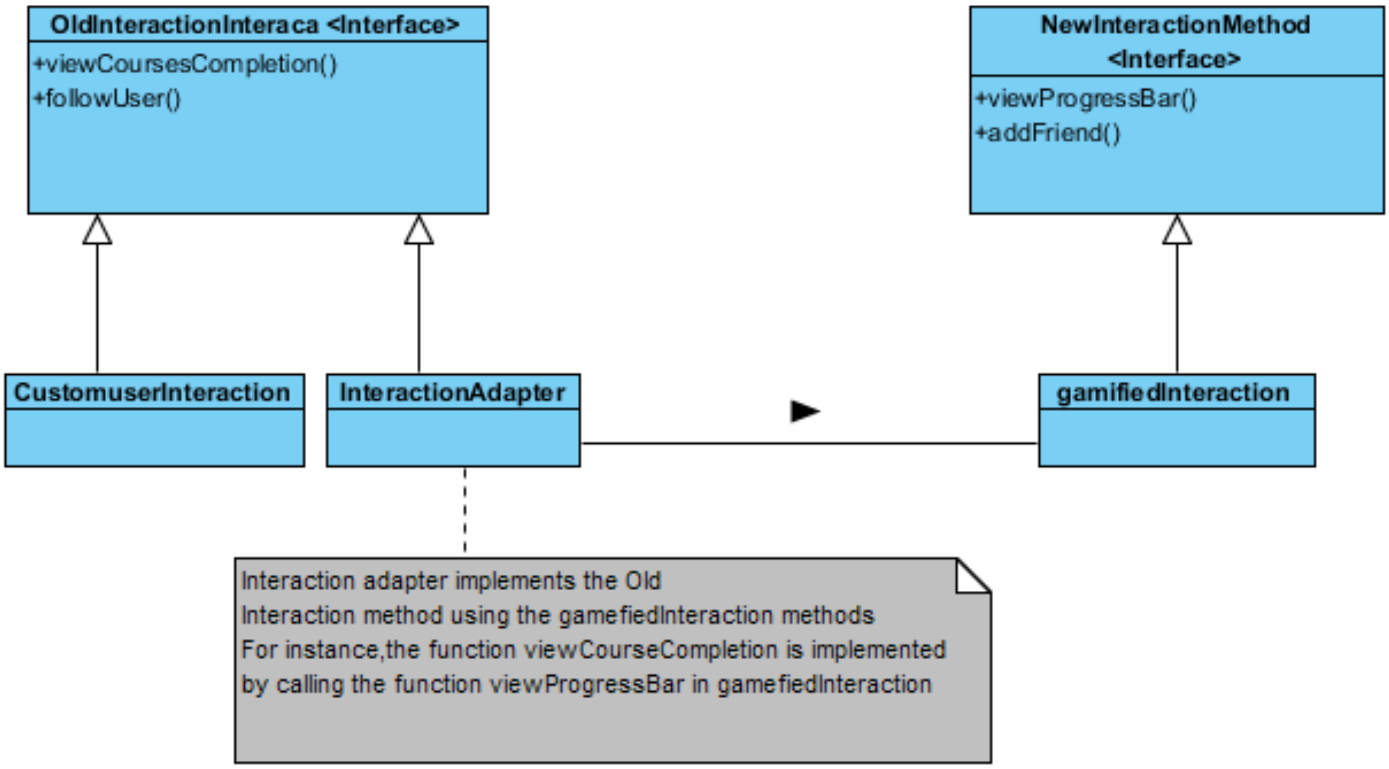


Code in project:EnrolledStudentsSingleton.java

2.STRUCTURAL DESIGN PATTERNS

2.1|Adapter Design Pattern

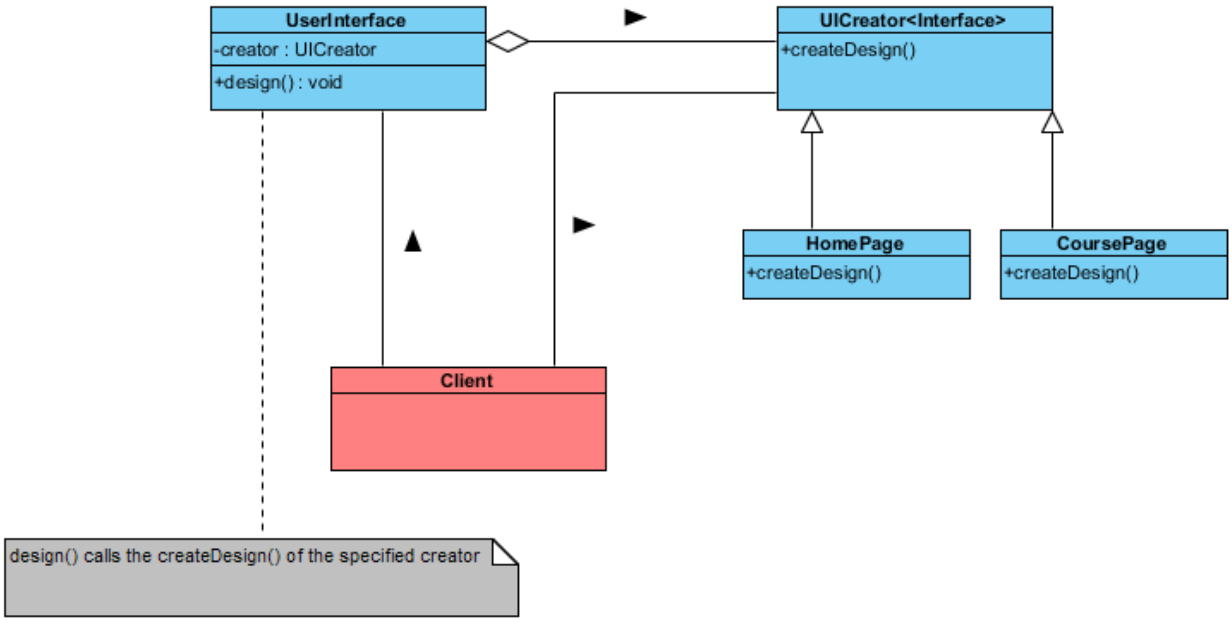
Gamefying some user interactions would be a great idea. So to convert the already designed interactions from a conventional way to a gamefied way , adapter design pattern shall be used as it converts the interface of a class into another desired interface.



Code in project:AdapterPattern.java

2.2|Bridge Design Pattern

Each page in the system has a different way in displaying its contents.For example the home page has a different user interface from that of the profiles pages and different from that of the courses pages . So to display pages differently , instead of subclassing the UserInterface abstract class into all the available pages ,the bridge design pattern could be used in which the bridge implementer is the UICreator interface and the abstraction using the implementer is the UserInterface class.The bridge design pattern decouples and abstraction from its implementation, so to create different pages the same abstract class method is used but with different implementations.

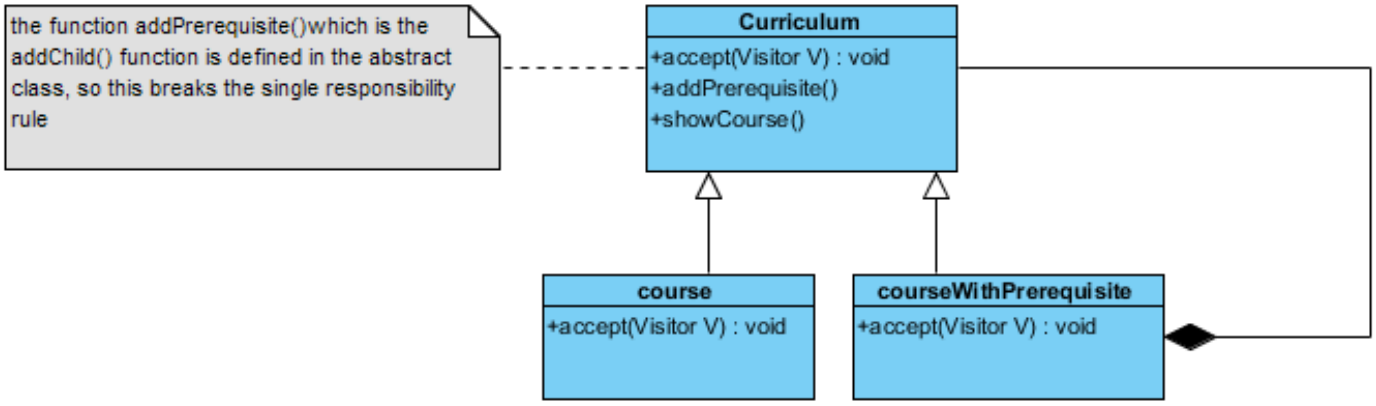


Code in project :bridgePattern.java

2.3|Composite Design Pattern

The composite design pattern is used to design and manipulate the course tree ,i.e. to manipulate the courses and their prerequisites.

Using the composite design pattern to manipulate such hierarichal structure ,we will not have to query the type of object before attempting to manipulate it.

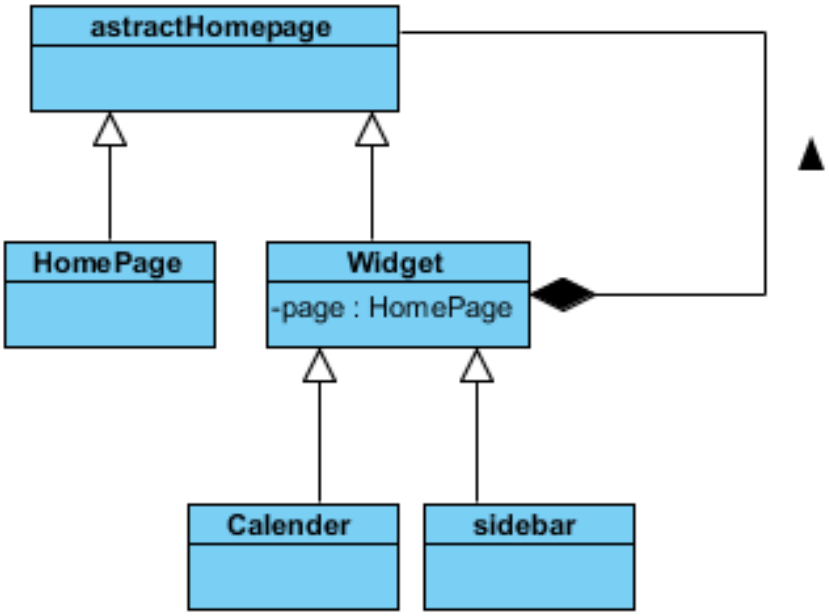


Code in project :Composite.java

2.4|Decorator Design Pattern

The user's default home page consists only of a menu showing the

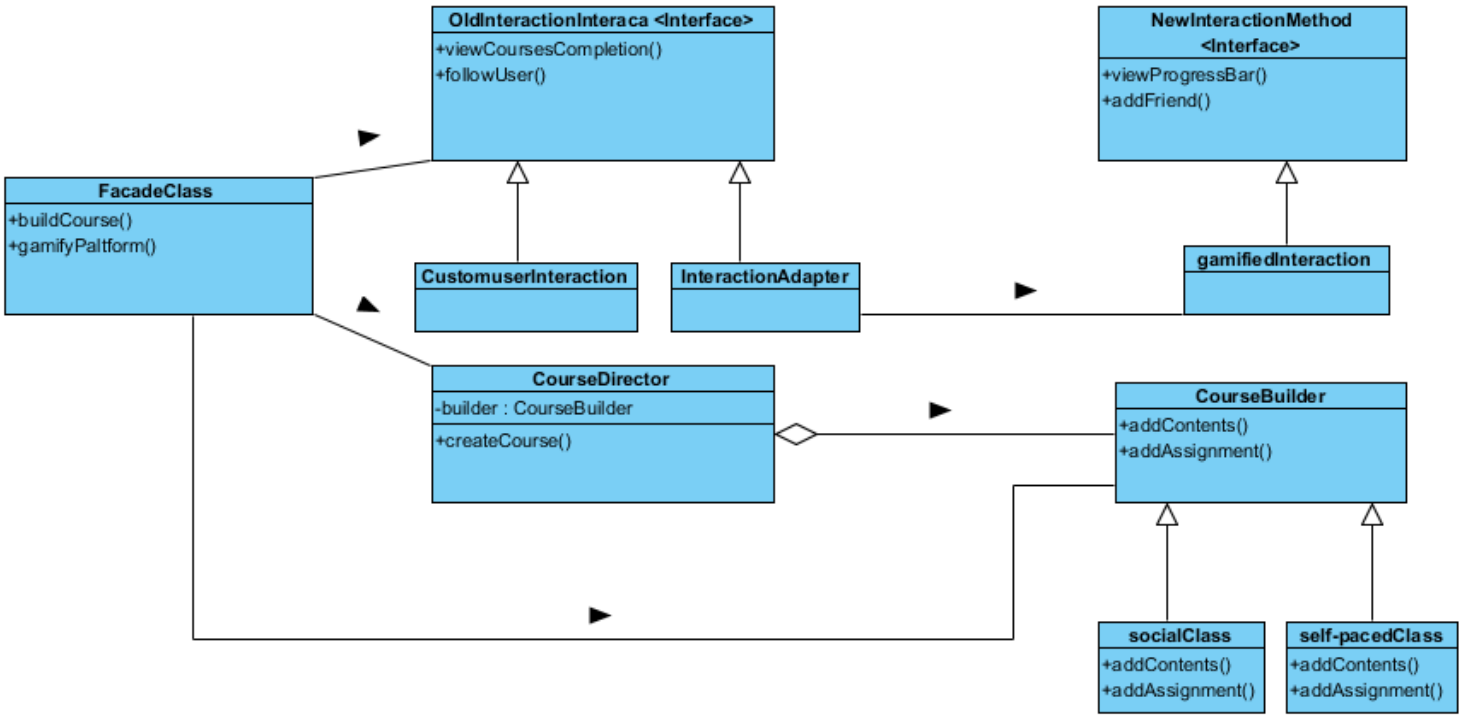
Courses the user is enrolled in .However allowing the user to add to his home page other widgets such as a calendar or a side bar that shows him his friends activities would make a better user experience .To have this functionality in the system the decorator design pattern could be used where the extra widgets are considered the decorators .



Code in project:classCreator.java

2.5|Façade Design Pattern

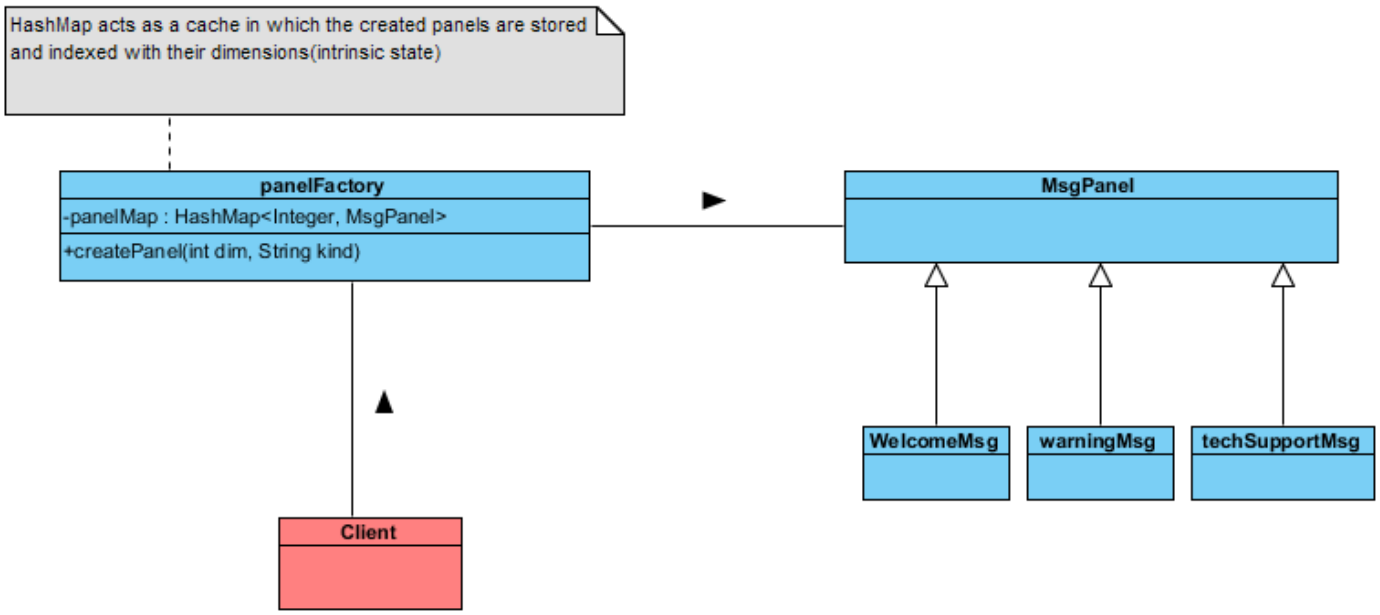
The façade pattern gives a unified interface for creating a class and updating its platform to a new gamified platform by providing a common interface for the interfaces that used in the builder design pattern and adapter design pattern .



Code in project: façadePattern.java

2.6|Flyweight Design Pattern

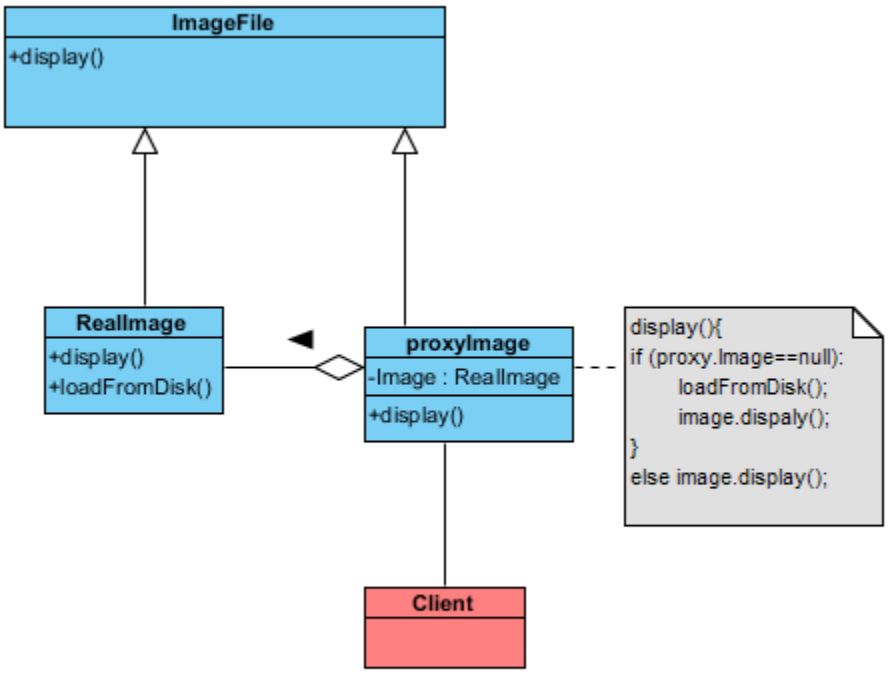
A user encounters many message panels while using the system, including welcome messages,helping messages,warning messages and many other .So the flyweight design pattern is used to o reduce the number of objects created and to decrease memory footprint and increase performance. In the pattern ,the dimensions of the panel is the intrinsic (shared)state which is stored in the flyweight object while the contents of the message,its color and the icon used within it are the extrinsic state.So the message panels don't have to be rendered every time a message pops up to the user. As there is a set of standard dimensions for the messages ,the flyweight design pattern will be used.



Code in project:FlyWeightPattern

2.7|Proxy Design Pattern

Our systems requires image Files from remote resources to be loaded .As loading such resources is expensive and very consuming, those files are loaded once and then kept in the proxy to be loaded to the system when needed instead of instantiating them from their remote resources.So proxy design pattern shall be used in which the client asks the proxy image class and inside the implementation of the proxy image ,the method of displaying the image first checks if the image is already in the proxy.If the image is not in the proxy it is loaded from the disk before being displayed.

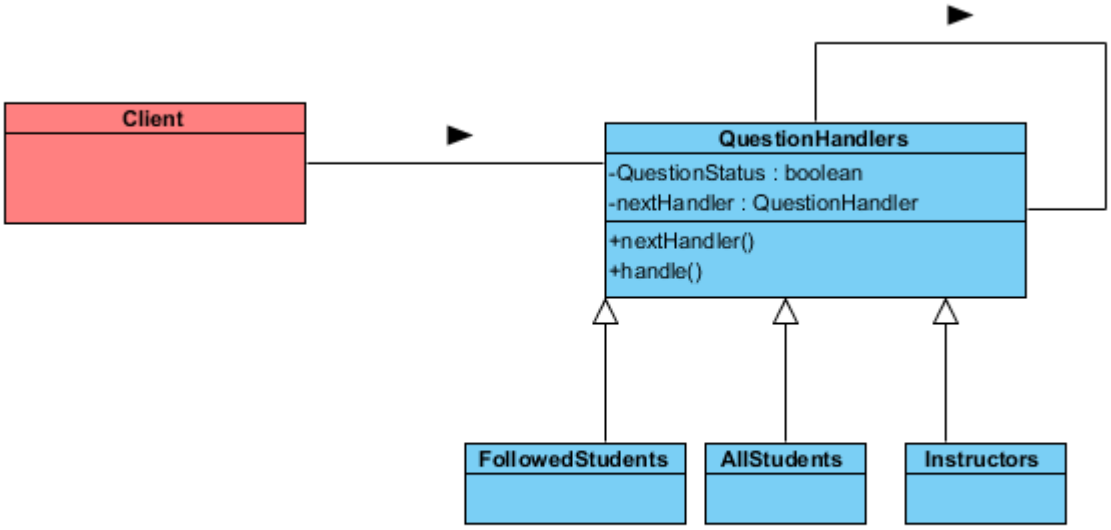


Code in project:proxyPattern.java

3.BEHAVIORAL DESIGN PATTERNS

3.1|Chain Of Responsibility Design Pattern

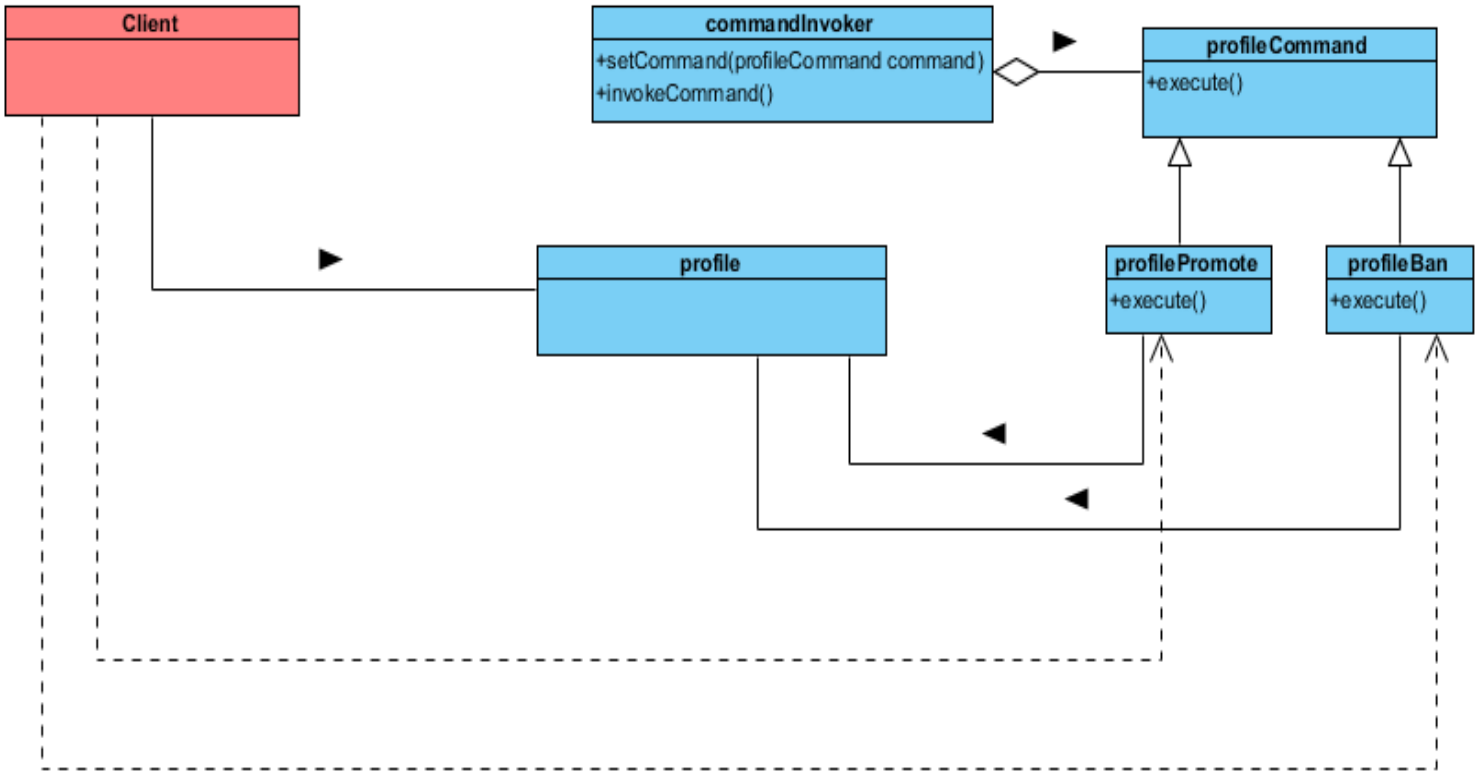
When a student posts a question .The first recipients to the question are those people he is following .Then if the question is not answered within 24 hours ,the next recipients to the question are the other fellow students registering in the system .Again if no answer is received within 24 hours the instructors receive the question to answer it. This behavior is best designed with a chain of responsibility design pattern.



Code in project: chainPattern.java

3.2|Command Design Pattern

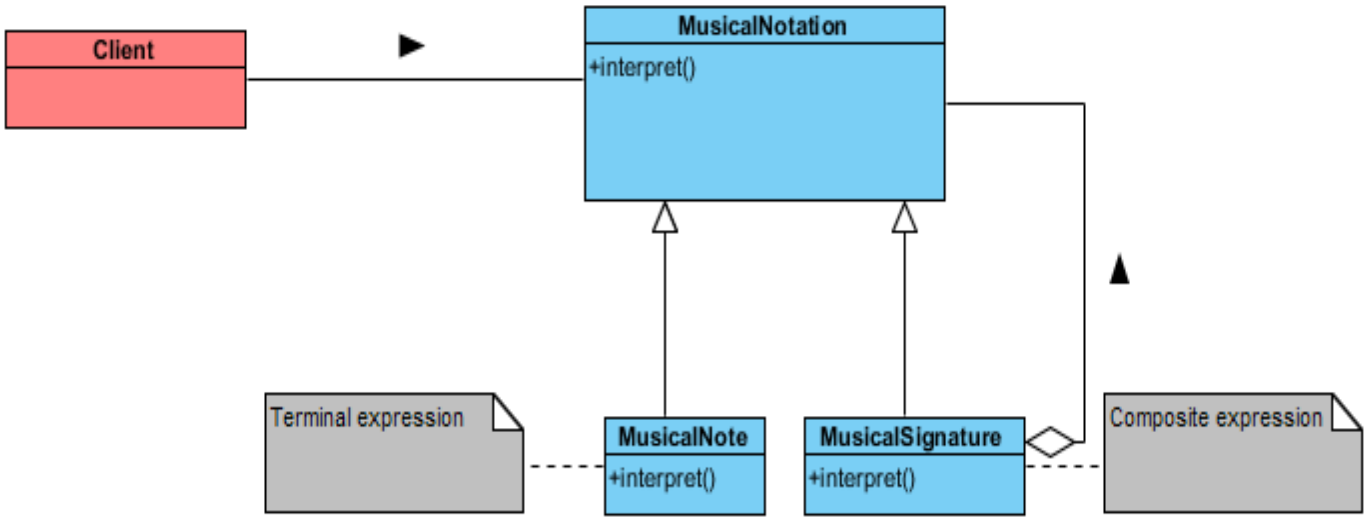
In our elearning system there is a kind of social interaction between users, as users are able to ask questions or answer questions. The user's profile could be promoted if he/she is initiative or helpful for other users in the system .While the profile could be banned if the user issues offensive context or unfollow the set . So to encapsulate a command ,such as the promote and ban commands , command design pattern is used.



Code in project:commandPattern.java

3.3|Interpreter Design Pattern

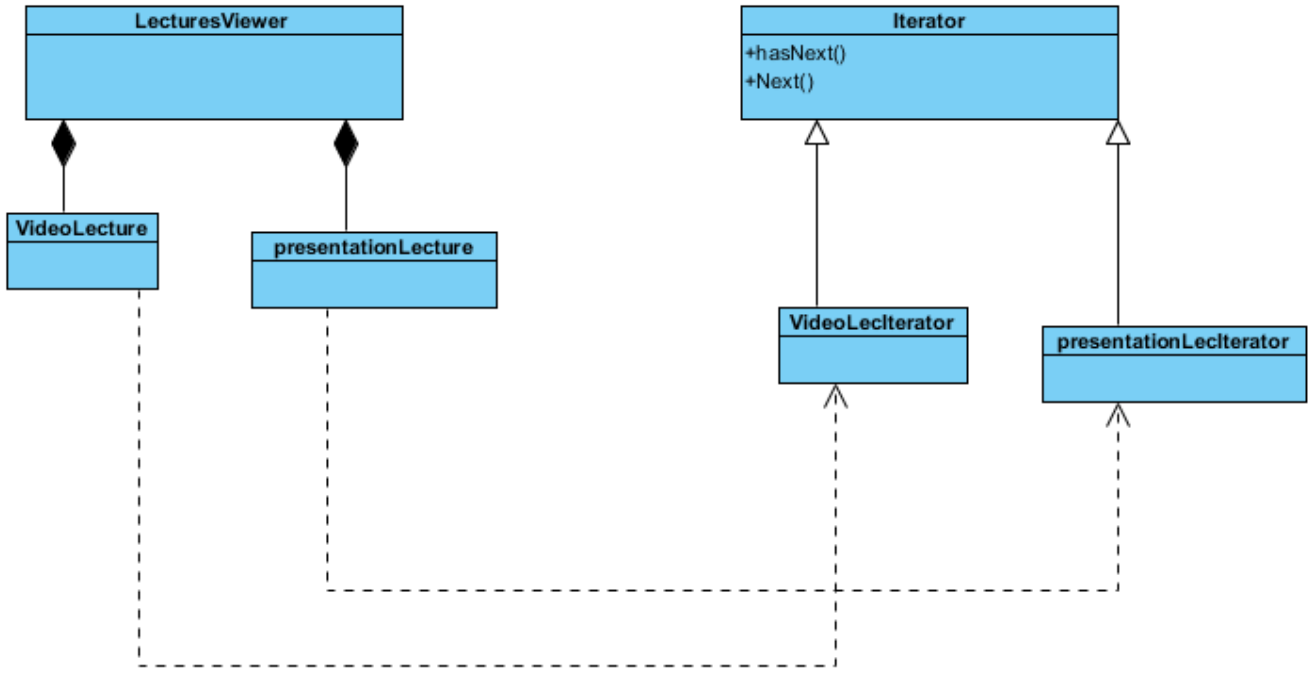
In music online classes to get things more enjoyable ,an interactive musical board is used where learners are supposed to drag and drop some given notes to make a musical time signature ,then the learner is supposed to click on the play button to hear the music reproduced. To design such module the interpreter design pattern could be used.



Code in project :interpreter.java

3.4|Iterator Design Pattern

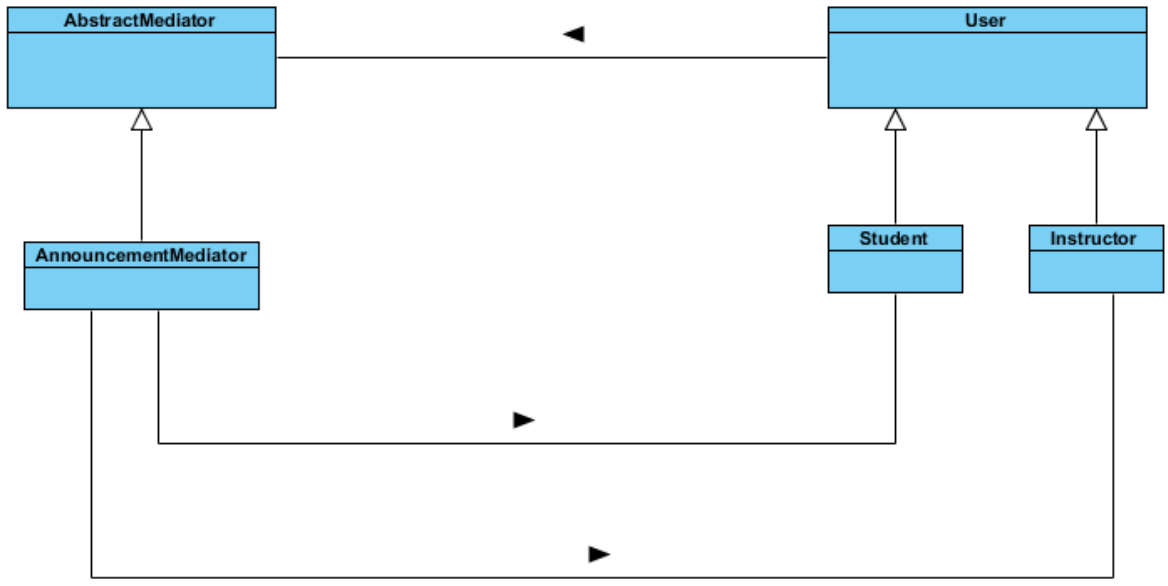
The courses are to be displayed in menus for students to choose from .One problem is that the containers in which the courses are saved vary with the media of the course contents .So video lectures are not saved in the same type of containers as presentations lectures .However to get over this problem and iterate over all the courses uniformly,iterator design pattern is used.



Code in Project:IteratorCreator.java

3.5|Mediator Design Pattern

Just as in real classrooms ,in virtual classrooms there should be kind of interaction between students or between students and their instructors so one kind of interaction is when an instructor wants to make an announcement to the students enrolled in his course. Mediator design pattern could be used to model such interaction in which the announcement could be broadcasted to other users without direct interaction between the instructor and his students to avoid the over coupling between classes .Instead a mediator class is responsible for the interaction ,and the users interact with the mediator class instead of interacting with each other.

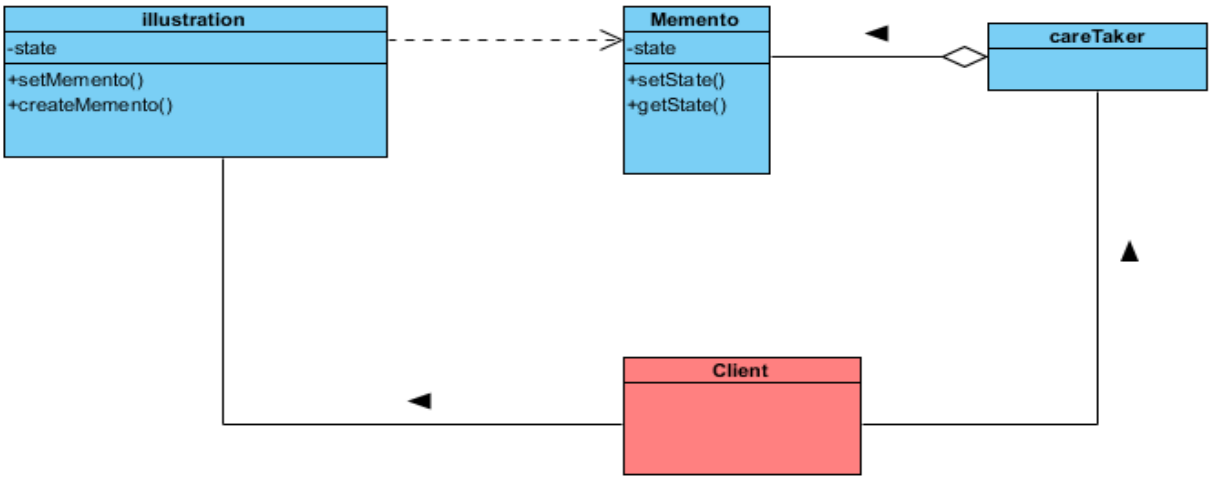


3.6|Memento Design Pattern

Just like in real classes ,some topics require diagrams to be displayed for the learners and sometimes they need to collaborate on these diagrams to be able to communicate and absorb the concept efficiently. But to sumup and to avoid confusion ,the diagrams are restored to specific states where the illustrations are clear and correct after all these interactions from other users.So the Memento design pattern is used in such case .When the diagram reaches a point where illustrations are clear and correct ,the instructor makes the caretaker save the current state of the originator , the whiteboard in our system,to the memento before proceeding to other state by the users interaction.

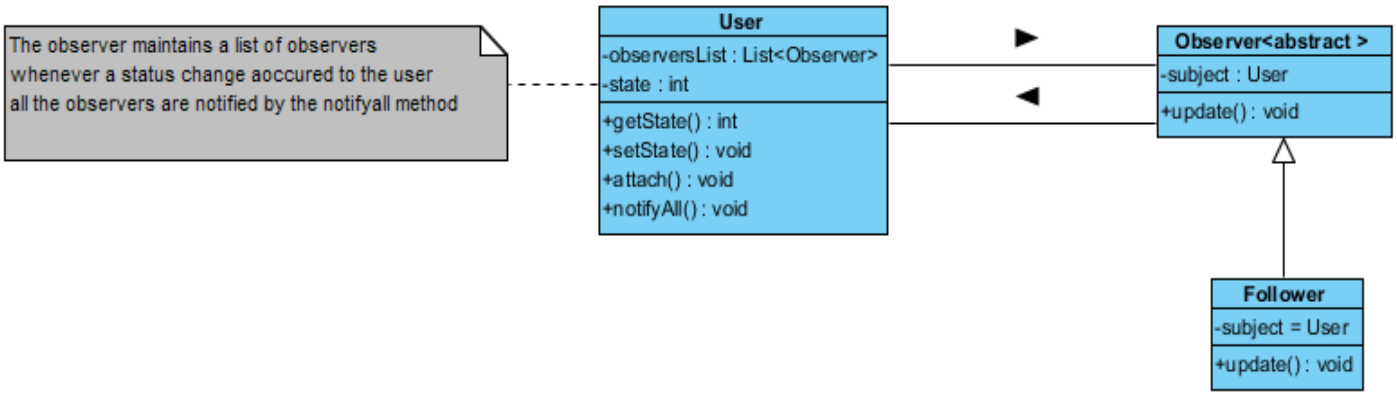
The Memento design pattern defines three distinct roles:

1. *Originator* - the object that knows how to save itself.
2. *Caretaker* - the object that knows why and when the Originator needs to save and restore itself.
3. *Memento* - the lock box that is written and read by the Originator, and shepherded by the Caretaker.



3.7|Observer Design Pattern

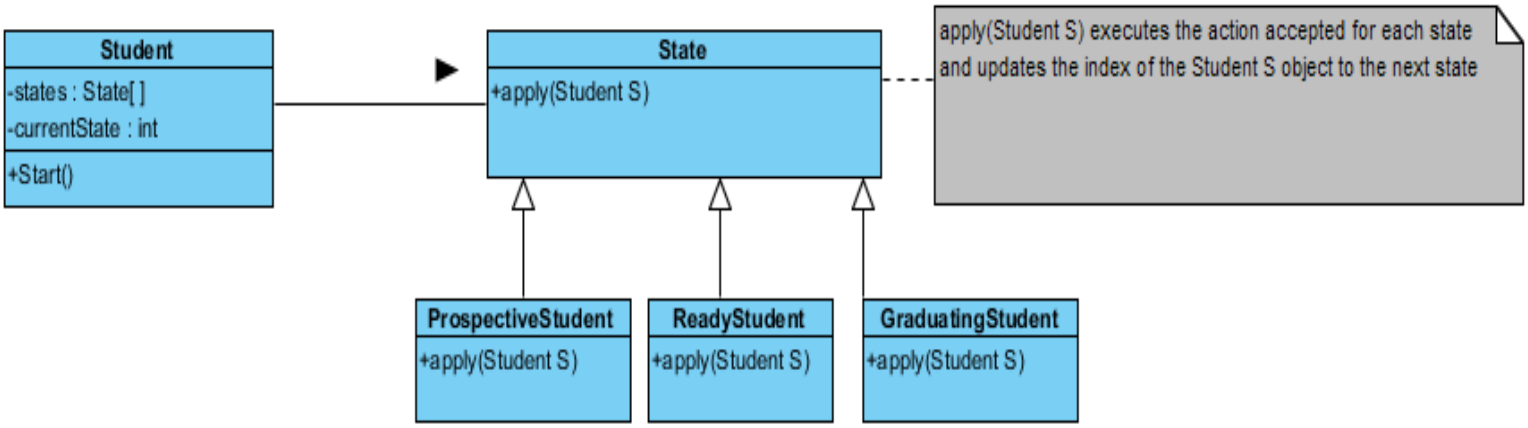
As the proposed e-learning system follows social learning strategy , users can follow each others public activities . As in any social network followers are notified if the user they are following commenced a public activity ,so to model this notification system, observer design pattern is used.



Code in project:observerPattern.java

3.8|State Design Pattern

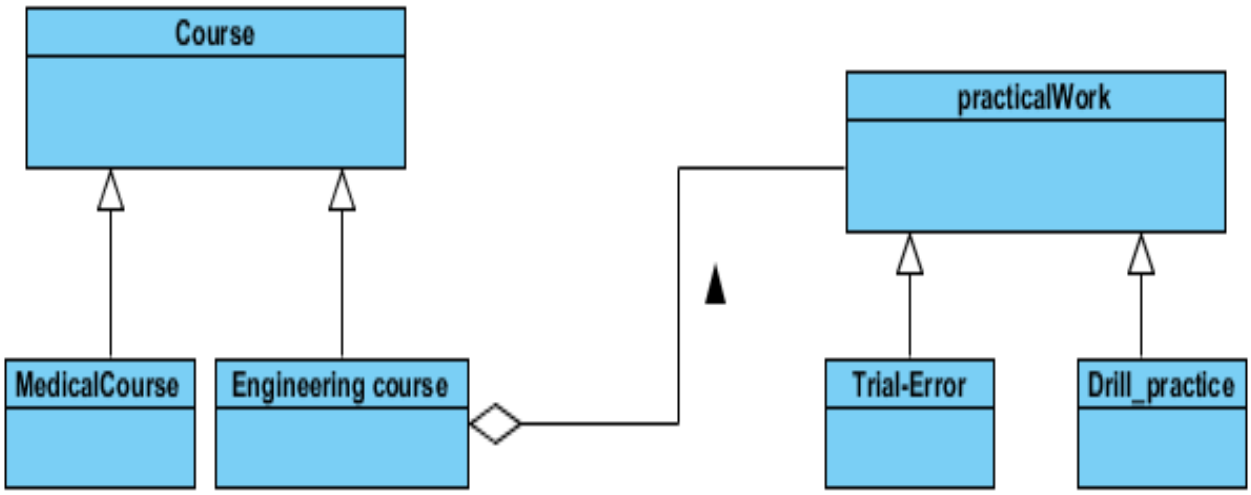
The enrolled students in a class are through 3 different states from the time they choose to enroll to the class to the time they complete the course .The first state is the “prospective student “state which is the state after user just accepts the terms of service ,then the user takes an online assessment to ensure that the prerequisites for the course are satisfied and moves on to the “Ready Student ” state ,then after completing the course the user moves on to the final state which is the “Graduating Student” state.



Code in project:statePattern.java

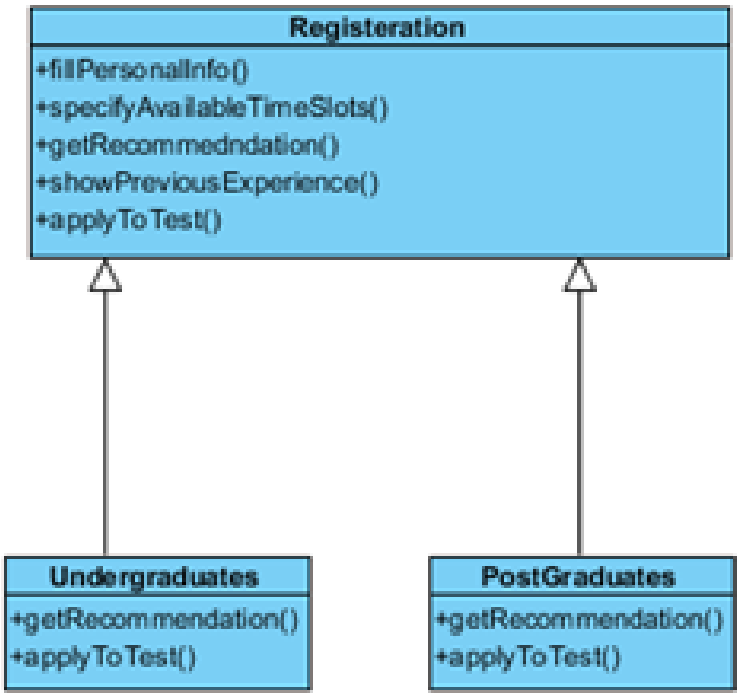
3.9|Strategy Design Pattern

Each course could be presented to the learner differently according to its content and to the kind of objective the course should deliver.So strategy design pattern is used to give the flexibility for each class to have its own strategy.



3.10|Template Design Pattern

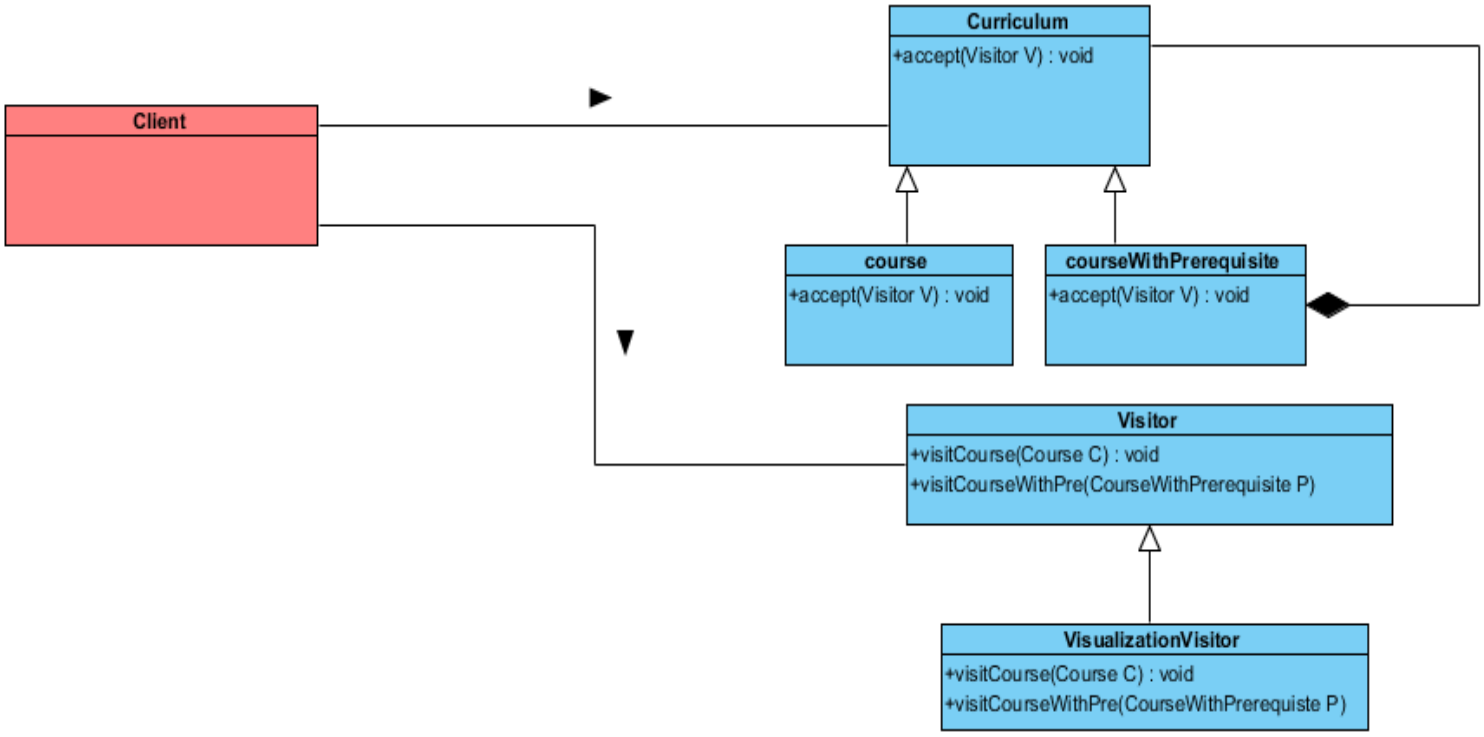
The registration process for both the graduates and undergraduates is similar for both except in some aspects.The different processes are abstracted such as those of the getRecommendation process,for the postgraduate ,the recommendation is required to be from an internship or real work experience while for an undergraduate it could be just an academic recommendation.Also applying for the test is different for both candidates.



Code :Template.java

3.11|Visitor Design Pattern

It would be easier for a student to visualize the hierarchy of the course tree and see the relation between the courses .To structure this hierarchy the ,the composite design pattern has been used to visualize the structure. The visitor design pattern has been used to avoid much modification for the main structure when modifying the visualization functionality.



CODE:VISITORPATTERN.JAVA