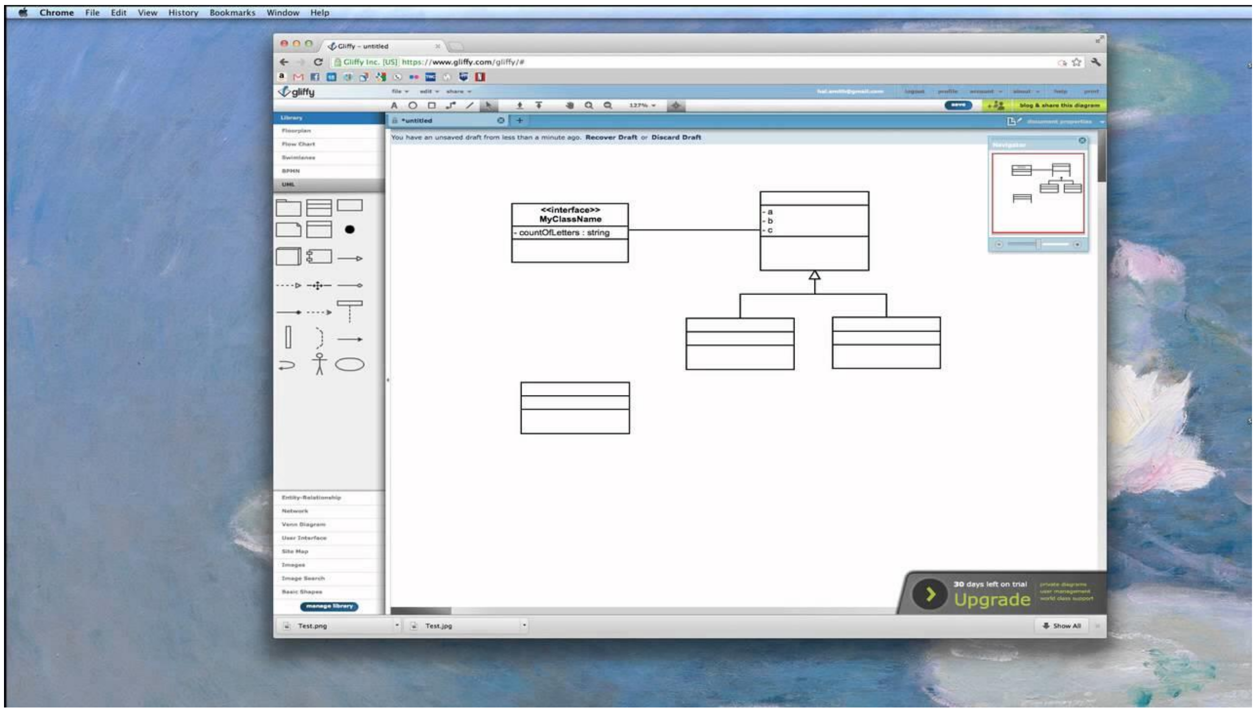
P09 – Pattern Based Software Designs

Creation of Design Patterns

Gliffy will be used to create design patterns for the selected scenarios. Since Gliffy provides a graphical user interface which is user friendly, it allows to develop UML patterns very easily.

Gliffy allows to generate JPG or PNG format pictures after completing the design. This is helpful as it enables these documents to be used effectively.



You have an unsaved draft from less than a minute ago. Recover Draft or Discard Draft

Necessary shapes can be dragged and dropped on to the design pane. This allows to create diagrams easily. Since Gliffy provides access to Google drive, we can collaborate and share documents with the group members.

Pattern Based Software Designs

.

Selected Design Patterns

i. Singleton
ii. Object Pools
iii. Observer
iv. Mediator
v. Adapter

Singleton

In a program where only one instance of a particular object is required, but when the point of invocation is not known, singleton can be used. This method ensures that only one instance of an object is created adhering to the object oriented principles.

Intent

- Ensures that only one instance of an object is created
- Provides a global point of access to the object

Implementation

In our program, a server object will store data temporarily for access. Since the first point of access is somewhat unclear, singleton will be used as follows.

```
Public class Server

{

        Private static Server instance;

        Private Server(){

                //Other initializations required

        }

        Public static synchronized Server getInstance() {
```

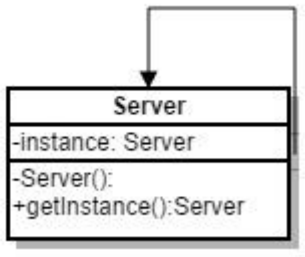
```
        If (instance == null )

                Instance = new Server ();

                Return instance;

}

}
```



<u>Object Pools</u>

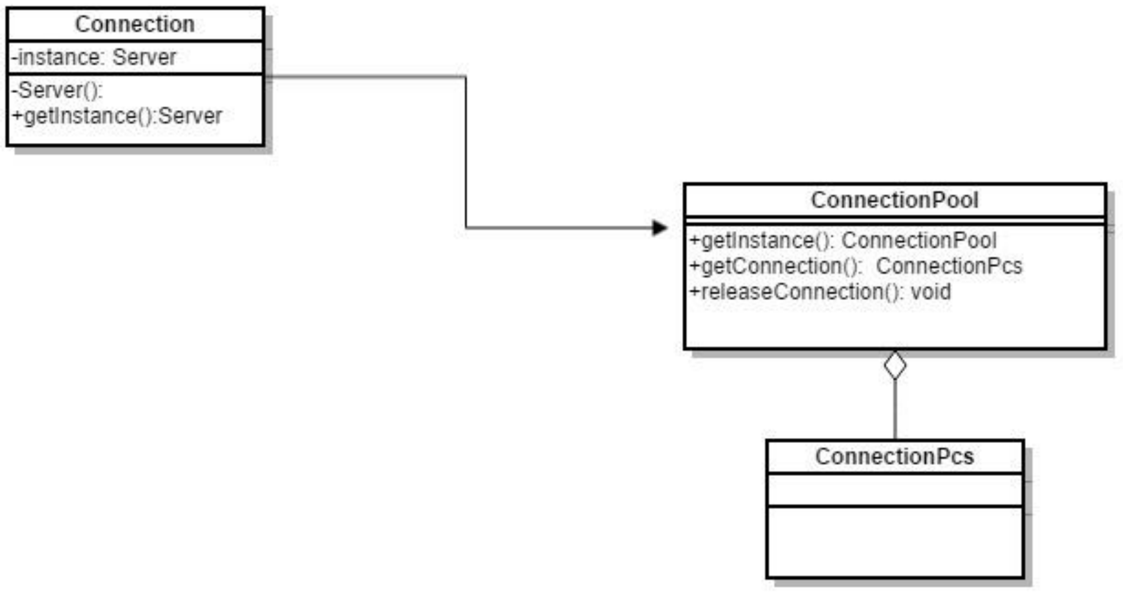
Object pooling allows to clone objects which are expensive to create. Clients of an object pool feel like they are the owner of the service although the service is shared among many other clients.

<u>Intent</u>

Reuse and share objects that are expensive to create.

<u>Implementation</u>

This will be used for database connections, since it’s expensive to create database access objects all the time.


Connection
-instance: Server
-Server():
+getInstance():Server
ConnectionPool
+getInstance(): ConnectionPool
+getConnection(): ConnectionPcs
+releaseConnection(): void
ConnectionPcs

<u>Observer</u>

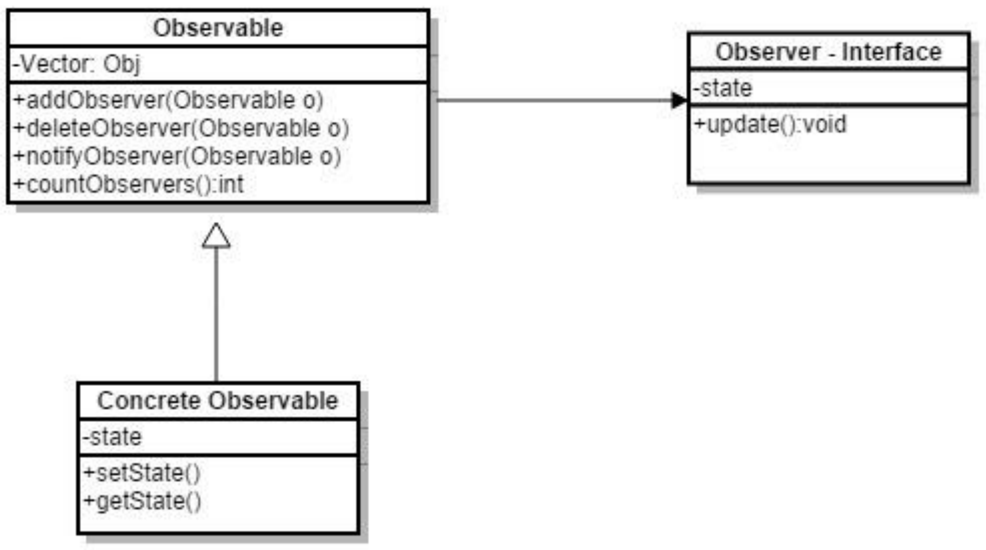
Object oriented programming is about objects and their dependencies. In a good design objects should be decoupled as much as possible. Object design patterns can be used when there's an observable object that needs to be monitored by several observers.

<u>Intent</u>

When the state is changed of one object, it's notified to the others.

<u>Implementation</u>

The business logic is contained in separate classes, and results need to be displayed on the GUI. Therefore the interface class should be an observer and the processing should be done within the Observable.

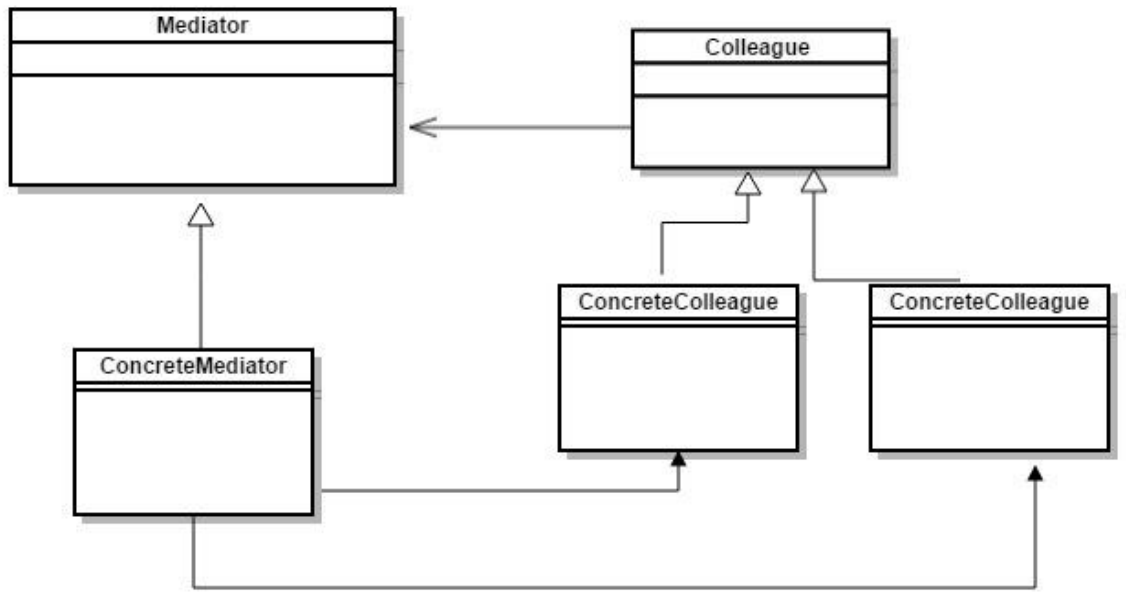


Mediator

In an object oriented design many classes are required for processing. When there are many object interactions, it might lead to tight coupled systems. Mediators allows to conceal the existence of some objects from others.

Intent

An object encapsulates how the set of objects interact. Loose coupling can be achieved because the objects are referring each other explicity.



<u>Adapter</u>

When the interface between objects need to be change, there may be situations where multiple changes are required thorough out the whole program. But instead an adapter class can be used to incorporate the changes.

<u>Intent</u>

- Converts the interface
- Classes work together

<u>Implementation</u>

Although there's no need of adapters at the moment, it is expected that with the development of software, adapter classes will be required to incorporate changes.

TRUE CODE TECHNOLOGIES

Group Members

Malith Jayaweera:	**120271A**
Randika Navagamuwa:	**120418H**
Buddhika Pathirana:	**120095L**
Indika Wijesooriya:	**120723M**
Tharindu De Silva:	**120026E**