Name : Md Farhan Ishmam
ID : 180041120

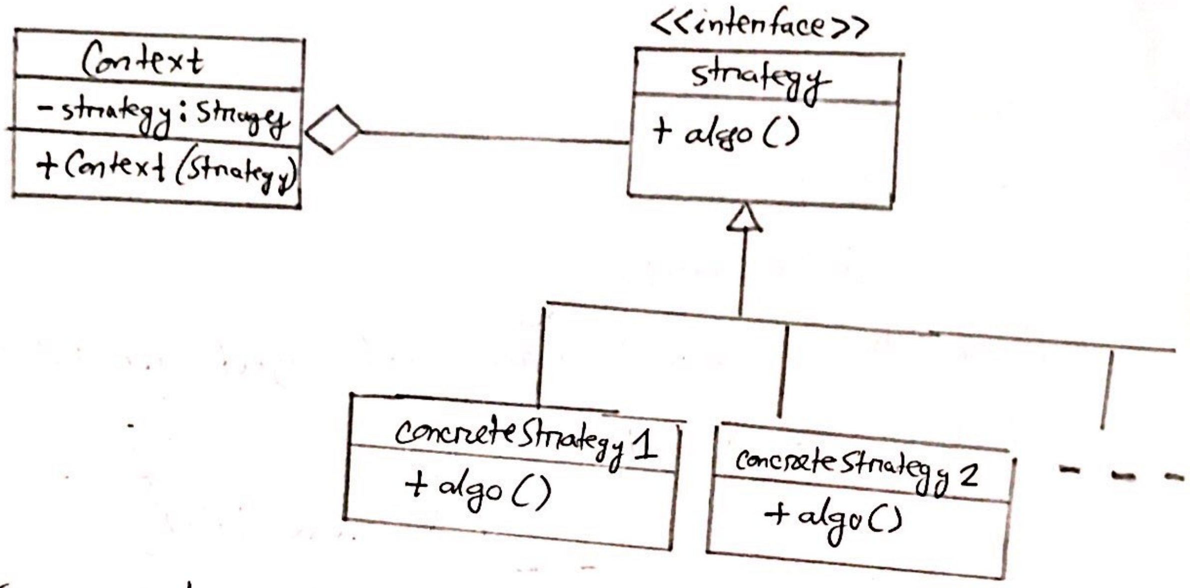
Name : Md Farhan Ishmam
ID : ~~1~~180041120
Section : CSE-1
Dept : CSE
Semester : Fifth
Course Code : CSE 4513
Course Title : Software Engineering and Object oriented Design
Date : 14-Sep-21

Ans. to Q.no. 1(a)(i)

I think strategy pattern would best support the given scenario.
Strategy pattern tells us to encapsulate the family of algorithms that change and ensure that these are interchangeable. The given system relies on a complex algorithm and it is modified/changed frequently. For such a system where algorithms change on a regular basis, strategy pattern tells us to encapsulate such algorithms and make the context class a composition of such algorithms. Due to this frequency of change of algorithms, I think strategy pattern would be the best in this scenario. Then, the algorithms can vary ~~indepently~~ independently from the way, clients use it.

Md Farhan Ishmam
180041120

Ans. to Q no. 1(a)(ii)



Key points : (i) The changes are encapsulates in strategy interface.

(ii) Context / client class is composed of strategies.

(iii) Conerete strategies implements the methods ~~#~~ defined in the strategy interface.

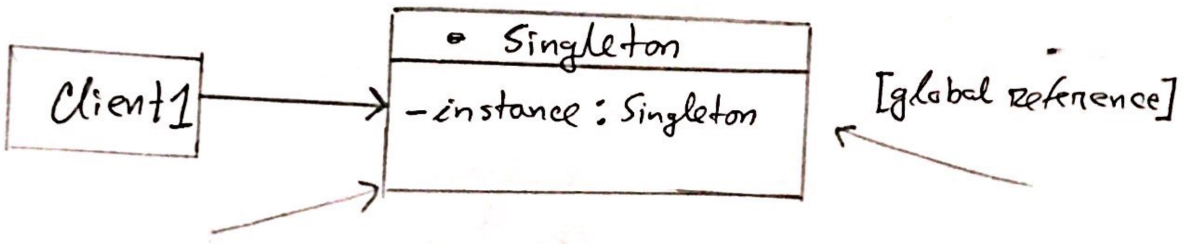
Md. Farhan Ishmam
180041120

Ans to Q.no. 1(b)

As the users can open only one Bank accaunt to make the trasaction, we should use Singleton Principle in this case. For the second case, the users will be ~~on~~ nottified by email for a transaction made, which fallows Observer Pattern.

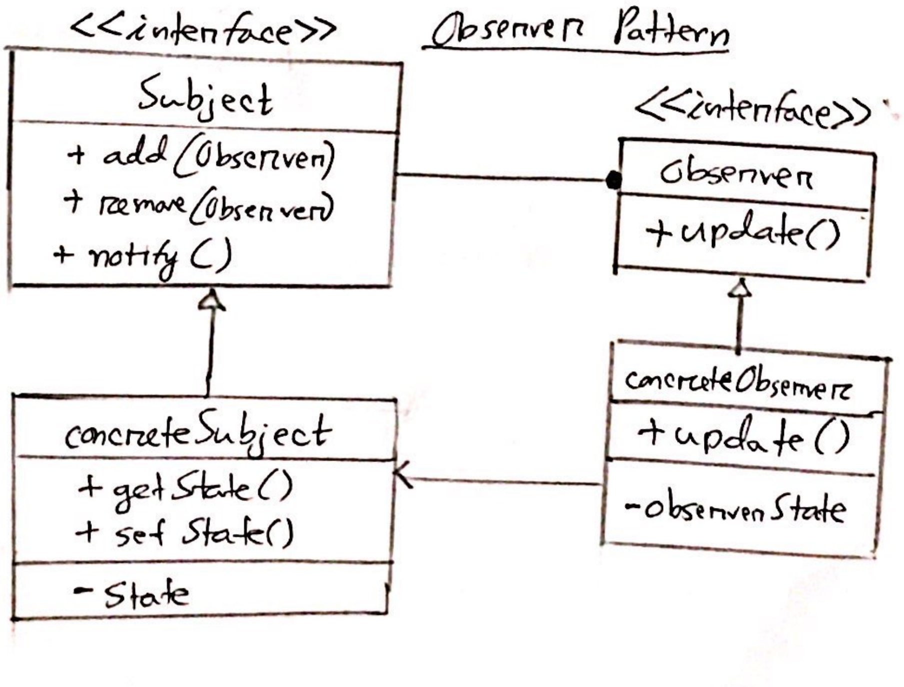
The class diagram of the ~~given~~ mentioned patterns are given below:

Singleton Principle



Key points: (i) Single instance of a class
(ii) Global reference to that single instance.

Observer Pattern



Md. Farhan Ishmam
180041120

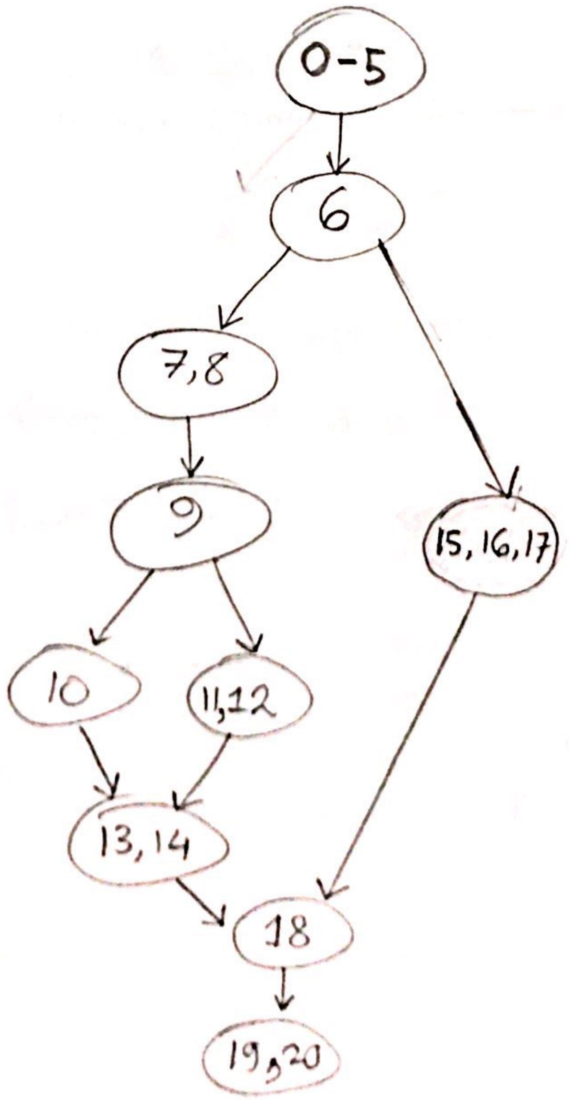
Key Points:

(i) Observer will be notified when a change occurs at subject which prevents subject overloaded by requests.

(ii) The ~~objec~~ observer and subject interfaces are made and then implemented in concrete implementations.

Ans. to Q.no. 3(a)

(i)



$$\begin{tikzcd}
 & 0\text{-}5 \arrow[d] & \\
 & 6 \arrow[dl] \arrow[ddr] & \\
7,8 \arrow[d] & & \\
9 \arrow[d] \arrow[dr] & & 15,16,17 \arrow[dddl] \\
10 \arrow[d] & 11,12 \arrow[dl] & \\
13,14 \arrow[dr] & & \\
 & 18 \arrow[d] & \\
 & 19,20 &
\end{tikzcd}$$

Md Farhan Ishmam
180041120

(ii)

The cyclomatic complexity is

$V(G) = e - n + 2P$ $\left[\begin{array}{l} e = \text{edges} = 11 \\ n = \text{no. of nodes} = 10 \\ P = \text{no. of connected components} = 1 \end{array}\right]$

$= 11 - 10 + 2 \times 1$

$= 3$

and, $V(G) = d + p$ $\left[\begin{array}{l} d = \text{no. of decision nodes} = (k-1) \text{ for each node.} \end{array}\right]$

$= (1+1) + 1$

$= 3$

and, $V(G) = \#$no. of regions

$= 3$

Ans: Cyclomatic complexity is 3.

(iii) No. of independent ~~Ans: 3~~. paths $= V(G) = 3$ (Ans.)

Ans to Q.no. 3(b)

Continuous ~~dt~~ delivery is the continuous integration practice where unit tests, integration test, acceptance tests and deployment are all done automatically without any manual intervenetion.

But, continuous deployment does the same process ~~autom~~ automatically, except deployment. Deployment has to be done manually. This is the difference between continuous deployment and continuous delivery.

Md Farhan Ishmam
180041120

Ans. to Q no. 3(c)

Re-engineering: The process in which the design of a software is ~~same~~ changed i.e. the software is re-written but functionality remains the same is called re-engineering.

Reverse Engineering: Process to achieve system specifications by analysing the product is called reverse engineering.

Reuse Process: (i) Same requirements, changing components
(ii) Same components, changing requirements.

The steps are -

a) Requirement Specification - functional and non-functional requirements are specified.

b) Design - The ~~architecture~~ system design and architecture is created.

c) Components Specifications - Segregate the system into smaller modular components studying the design.

d) Search and Reuse Components - Search component repository and find suitable components to reuse.

e) Incorporate Components - Use matched components in developing the new system.

Md. Farhan Ishmam
180041120

Ans. to Q. nr. 2(a)

(i)

A bug is registered if username less than 6 characters is accepted. The expected output should be an error message and ask for username again.

For this I think dynamic testing should be done using black box testing. Robustness test needs to be performed. The limiting value is 6 characters. The QA team must test with less than 6 characters (min⁻), exactly 6 characters (min) and more than 6 characters (min⁺). When the expected results are found from desired test case; QA team can assure bug is solved.

(ii)

Black-box testing composes of different tests ~~which~~ in ~~their~~ which the tester has no knowledge of the source code of the system. The functional requirements can be verified using black box testing.

Black-box testing doesn't need the tester to be an expert in identifying source code problems. The tests can be quickly ~~done~~ developed and executed to check if the function requirements work properly. For boundary value checking, robustness tests and worst case analysis we can ensure the system doesn't behave unexpectedly at corner-cases and unwanted inputs. Path-based testing ensures each path is error-free. Thus by removing bugs and unwanted results, a quality software is made.

Ans. to Q. no. 2(b) (i)

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Condition Stub	C1: Doctor's Office	T	F	F	F	F
	C2: V1 Hospital Visit	F	T	F	F	F
	C3: V2 Hospital Visit	F	F	T	F	F
	C4: V3 Hospital Visit	F	F	F	T	F
	C5: V4 Hospital Visit	F	F	F	F	T
Action Stub	A1: 33% reimburse	X				
	A2: 50% reimburse		X			
	A3: 66% reimburse			X		
	A4: 70% reimburse				X	
	A5: 90% reimburse					X

A visit can be one of those 5 decisions and the outcome is also one of five decisions, at a time.

(ii)

The minimum number of test cases to cover the full decision table is five. (Ans.)