CMPE 331- SOFTWARE ENGINEERING CONCEPTS

OBSERVER PATTERN

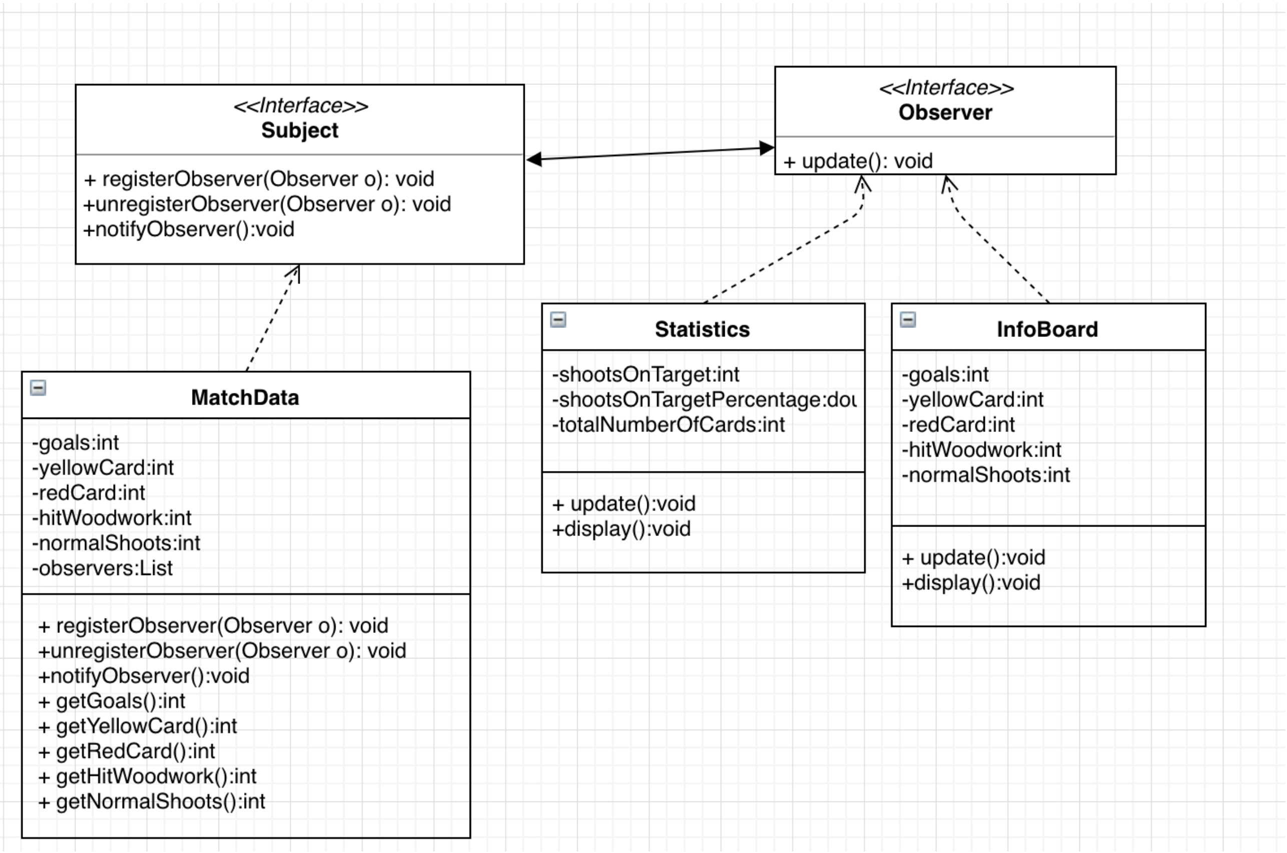
Behavioral patterns are the patterns that are most specifically concerned with communication between objects. Observer Pattern is one of the types of the behavioral design patterns.

Observer pattern is essentially a way of notifying change to a number of classes, which defines a one-to-many dependency between objects so that one object changes state, all of its dependents are notified and updated automatically.

The dependency relation is between a Subject(One) and Observer(Many) classes. First, Subject and Observer are implemented as interfaces. Subject which has the functions to register, unregister and notify all the observers and Observer is implemented to update the information and inform the observing classes with the data that is coming from the Subject class.

To give the idea of the mechanics of the Observer Pattern, it would be a good idea to explain the subject with a simple analogy. For instance, in the auctions, there are an auctioneer(Subject) and several bidders(observers). When one of the bidders raises the paddle with a number that indicates the new bid from this very person and the auctioneer accepts the bid and broadcast the most recent bid to the other bidders.

TASK: You are required to provide Statistics and InfoBoard classes' implementation of the following UML Diagram. The Subject and Observer interfaces, MatchData and PatternDemo classes are provided.



<<Interface>>
Subject
+ registerObserver(Observer o): void
+unregisterObserver(Observer o): void
+notifyObserver():void
<<Interface>>
Observer
+ update(): void
MatchData
-goals:int
-yellowCard:int
-redCard:int
-hitWoodwork:int
-normalShoots:int
-observers:List
+ registerObserver(Observer o): void
+unregisterObserver(Observer o): void
+notifyObserver():void
+ getGoals():int
+ getYellowCard():int
+ getRedCard():int
+ getHitWoodwork():int
+ getNormalShoots():int
Statistics
-shootsOnTarget:int
-shootsOnTargetPercentage:dou
-totalNumberOfCards:int
+ update():void
+display():void
InfoBoard
-goals:int
-yellowCard:int
-redCard:int
-hitWoodwork:int
-normalShoots:int
+ update():void
+display():void