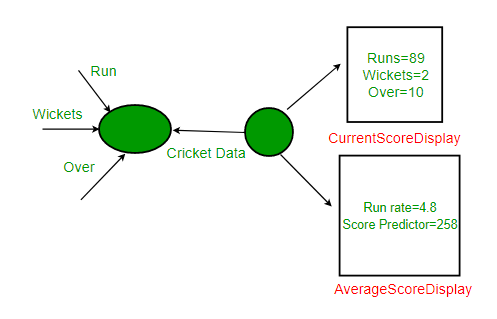
Exercise:

Suppose we are building a cricket app that notifies viewers about the information such as current score, run rate etc. Suppose we have made two display elements CurrentScoreDisplay and AverageScoreDisplay. CricketData has all the data (runs, bowls etc.) and whenever data changes the display elements are notified with new data and they display the latest data accordingly.[](https://media.geeksforgeeks.org/wp-content/uploads/ObserverPatternSet-1.png)

Open exercise solution “ObserverSolutionWithoutPattern”.

After watching our first solution we identify some problems, such as:

* CricketData holds references to concrete display element objects even though it needs to call only the update method of these objects. It has access to too much additional information than it requires.
* This statement “currentScoreDisplay.update(runs,wickets,overs);” violates one of the most important design principle “Program to interfaces, not implementations.” as we are using concrete objects to share data rather than abstract interfaces.
* CricketData and display elements are tightly coupled.
* If in future another requirement comes in and we need another display element to be added we need to make changes to the non-varying part of our code(CricketData). This is definitely not a good design practice and application might not be able to handle changes and not easy to maintain.

Apply the observer pattern to solve this problems.

To get the solution in detail visit:

[Observer Pattern | Set 2 (Implementation) - GeeksforGeeks](https://www.geeksforgeeks.org/observer-pattern-set-2-implementation/?ref=lbp)