CricInfo has introduced a new service whereby cricket fans can stay up to date with the score of a match being currently played so that they don’t have to check the website each time they need an update. The organization managed to get hold of a cool JavaWorld programmer and asked him to implement this simple program in Java. This is what the programmer came up with:

class Mobile{
  // Some code here
  public void scoreUpdated(String player, String score){
    //Do Something Here
  }
}

class Radio{
  // Some code here
  public void updatedScore(String player, String score){
    //Do Something Here
  }
}

class ScorePublisher {
  public Mobile mobileObj;
  public Radio radioObj;

  public ScorePublisher() {
    mobileObj = new Mobile();
    radioObj = new Radio();
  }

  incrementScore(String player, String score) {
    mobileObj.scoreUpdated(player, score);
    radioObj.updatedScore(player, score);
  }
}
public class Driver(){
  public static void main(String[] args){
    ScorePublisher sp = new ScorePublisher();
    for (int i = 0; i < 10; ++i) {
      pl = JOptionPane.showInputDialog("Enter Player name");
      sc = JOptionPane.showInputDialog("Enter Score");
      sp.incrementScore(pl, sc);
    }
  }
}

The basic idea is to have the objects of all the people that want to get the score updates in the ScorePublisher class. Every time there is a change in score, the methods of all the people are being called. So basically, whenever a new subscriber needs to be added, slight modifications are made in the code and the code is recompiled.

The Driver class prompts the user for the player’s name and the new score 10 times (Just a random loop set to test the program)
PART II - Issues with Current Implementation

Some smart student of CS 391 highlighted certain flaws in the issue:

- If a new subscriber needs to get registered to the system then that would require some code redo and the code will also need to be recompiled.

The student proposed the following solution:

Maintain an ArrayList in the class ScorePublisher and add the subscribers to that ArrayList. The modified class looked somewhat like this:

class ScorePublisher {

    public ArrayList subscribers;

    public ScorePublisher() {
        subscribers = new ArrayList();
    }

    public void incrementScore(String player, String score) {
        int listSize = subscribers.size();
        for (int i = 0; i < listSize; ++i) {
            Object tempS = subscribers.get(i);
            tempS.updateScore();
        }
    }

    public void subscribeUser(Object newUser) {
        subscribers.add(newUser);
    }
}


public class Driver(){

    public static void main(String[] args){
        ScorePublisher sp = new ScorePublisher();
        Mobile m = new Mobile();
        Radio r = new Radio();
        sp.subscribeUser(m);
        sp.subscribeUser(r);

        for (int i = 0; i < 10; ++i){
            pl = JOptionPane.showInputDialog("Enter Player name");
            sc = JOptionPane.showInputDialog("Enter Score");
            sp.incrementScore(pl, sc);
        }
    }
}

• Why is an Object type being passed in the subscribeUser method?

• Can you identify any issue with the for loop in the incrementScore method
PART III

Now all this seems pretty good, but there are still some glitches.

- Objects are being added to the ArrayList. Now when the objects are being taken out of the Array in the for loop, with what object type do we cast them? (Mobile Class or Radio Class).

- Even if the object casting issue is resolved, which method should be called on the object (since Mobile Class has the method scoreUpdated() and Radio class has the method updatedScore()).

What is the possible solution that you can think of in this situation?
The student then enlightened CricInfo that there is yet another smarter solution to the problem at hand. She was smart enough to highlight that a contract needs to be established by the subscribers with the program. She proposed the use of interface to deal with the problem.

```java
interface UpdateScore{
    public void updatedScore();
}
```

The new implementation of the classes would look somewhat like this:

```java
class Mobile implements UpdateScore {
    // Some code here
    public void updatedScore(){
        //Bhow Bhow
    }
}
class Radio implements UpdateScore {
    // Some code here
    public void updatedScore(){
        //Bhow Bhow
    }
}
```

The class ScorePublisher would then look somewhat like this:

```java
import java.util.*;
public class ScorePublisher{
    public ArrayList subscribers;

    public ScorePublisher() {
        subscribers = new ArrayList();
    }

    public void incrementScore() {
        int listSize = subscribers.size()
        for (int i = 0; i < listSize; ++i){
            UpdateScore tempS = (UpdateScore)subscribers.get(i);
            tempS.updatedScore();
        }
    }
}
```
public void subscribeUser(updateScore newUser) {
    subscribers.add(newUser);
}

public class Driver() {

    public static void main(String[] args) {
        ScorePublisher sp = new ScorePublisher();
        Mobile m = new Mobile();
        Radio r = new Radio();
        sp.subscribeUser(m);
        sp.subscribeUser(r);

        for (int i = 0; i < 10; ++i) {
            pl = JOptionPane.showInputDialog("Enter Player name");
            sc = JOptionPane.showInputDialog("Enter Score");
            sp.incrementScore(pl, sc);
        }
    }
}