Assignment No. 1

Name	**Ahmad Jalal**
Roll No	**20L-1305**
Section	**BSSE-7A**
Course	**Software for Mobile Devices**

Problem related to the class structure of question no. 1.

Diamond problem:

In object-oriented programming, the phrase "diamond problem" is used to refer to a specific form of ambiguity that can happen in languages that enable multiple inheritance. The diagram of the inheritance hierarchy, which is in the shape of a diamond, gave rise to the term "diamond problem" for this issue.
In our case, it is a class hierarchy where **AllRounder** extends both **Batsmen** and **Bowler**, which both extend a common base class **Cricketer.**

When the diamond problem occurs following are the major challenges that we can face:

Ambiguity in Method Resolution:
There may be ambiguity in method resolution if the **Batsmen** and **Bowler** classes both specify a method with the same name, and **AllRounder** inherits from both. For instance, executing **hitBall**() on an **AllRounder** object can be unclear if both Batsmen and Bowlers have the function **hitBall**().

In order to overcome this, we can either provide the conflicting function a particular implementation in the **AllRounder** class or use method overriding to make it clear which implementation to use.

Data Member Ambiguity:
Uncertainty may result if both the **Batsmen** and the **Bowler** have data members with the same name. To explicitly access the data members from the parent classes, we need either make sure the data members have unique names or utilize access modifiers (such as super).

Constructor Ambiguity:
When building an **AllRounder** object, we could get into confusion if both Batsmen and Bowler have constructors with arguments. To fix this, add a constructor to the **AllRounder** class that calls the **Batsmen** and **Bowler** constructors appropriately.