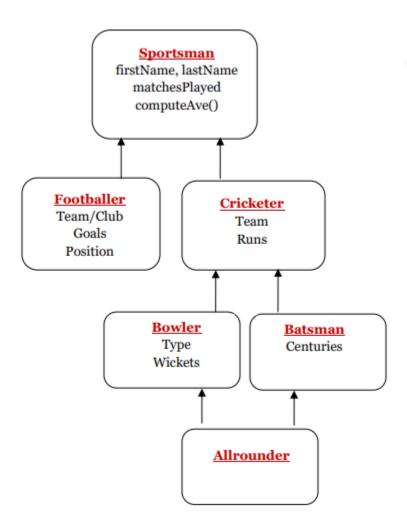
Object Oriented Programming

Task 5 Report

Problem Statement:

Implement the following class hierarchy in C++. Write a driver to test the hierarchy by displaying data of your favorite players.



Objectives:

- To show how to create a clean and well-organized object oriented program that uses most of the concepts studied so far
- To show how to use separate compilation so that each class can be stored in a separate header file and all functions of all classes can be defined in just one cpp file.
- To show how different operations in a problem statement can be converted into class member functions.
- To identify which member functions of base class can be turned into pure virtual functions to allow dynamic (runtime) polymorphism
- To show how to use multiple and multi-level class inheritance to meet specified requirements
- To show how to avoid ambiguity (virtual inheritance) when dealing with the Diamond problem in class inheritance
- To show how to use an array of pointers of an abstract class to store objects of derived classes
- To show how to use file handling to read data into an array of objects
- To show how to overload "<<" and ">>" operators to display class objects

UML diagram:

Sportsperson

playerType : string
recordNum : int
name : string
gender : char
matchesPlayed : int

+ Sportsperson()

+ Sportsperson(a: string, b: int, c: string, d: char, e: int)

+ virtual computeAvg() = 0 : float

+ virtual readData(line : string, array[] : string) = 0 : void

+ virtual printData() = 0 : void

+ <<friend >> operator << (out : ostream&, s : Sportsperson&) : ostream&

+ <<friend >> operator >> (out : ostream&, s : Sportsperson&) : ostream&

Footballer

team : stringposition : stringgoals : int

+ Footballer()

+ Footballer(a : string, b : int, c : string, d : char, e : int, f : string, g :

string, h: int)

+ computedAvg() : float

+ readData(line : string, array[] : string) : void

+ printData(): void

Cricketer # team : string # runs : int + Cricketer() + Cricketer(a: string, b: int, c: string, d: char, e: int, f: string, g:int) + returnRuns(): int + computeAvg(): float + readData(line : string, array[] : string) : void + printData(): void Batter # centuries : int + Batter() + Batter(a : string, b : int, c : string, d : char, e : int, f: string, g: int, h: int) + readData(line : string, array[] : string) : void + printData(): void Bowler # centuries : int + Bowler() + Bowler(a : string, b : int, c : string, d : char, e : int, f : string, g: int, h: string, i: int) + readData(line : string, array[] : string) : void + readData(x: int, line: string, array[]: string): void + printData(): void + printData(x : int) : void

Allrounder		
+ Allrounder()		
+ Allrounder(a : string, b : int, c : string, d : char, e : int, f :		
string, g: int, h: int, i: string, j: int)		
+ readData(line : string, array[] : string) : void		
+ printData() : void		

Tools Used: Visual Studio 2019, Microsoft Word

Sample Program Outputs:

Welcome screen:



Start screen:



Data output:

C:\Users\Naeem\source\repos\oopjourno2\Debug\oopjourno2.exe Player Info: Record number: 1 Player type: All-rounder Player name: Sana Asif Gender: F Matches: 78 Current team: Pakistan Total runs: 1400 Centuries: 7 Bowler type: Spinner Total wickets: 167 Average: 17.9487 Record number: 2 Player type: All-rounder Player name: Junaid Akram Gender: M Matches: 62 Current team: UAE Total runs: 1190 Centuries: 5 Bowler type: Pacer Total wickets: 120 Average: 19.1935 Record number: 3 Player type: Footballer Player name: Critiano Ronaldo Gender: M Matches: 500 Current team: Juventes Plays at: Forward Total goals: 1000 Average: 2 Record number: 4

Exit screen:



Conclusion:

It can be seen that it is indeed possible to create such an information system using UML diagrams, classes, text files, operator overloading, multilevel, multiple and virtual inheritances, pure virtual functions and dynamic polymorphism.