Department of Computer Science and Information Technology
La Trobe University

CSE1OOF/4OOF Summer 1, 2018
Assignment Part C

7%

This is an individual Assignment. You are not permitted to work as a Pair Programming partnership or any other group when writing this assignment.

Due Date

Due : Thursday 24th of January 2019 at 10.00 a.m.
Execution test : Friday 25th of January 2019 at 1.00 p.m

Delays caused by computer downtime cannot be accepted as a valid reason for a late submission without penalty. Students must plan their work to allow for both scheduled and unscheduled downtime. **There are no days late or extensions on this assignment as execution test marking will begin on** Friday 25th of January 2019 **– in your normal lab (Week 9).** After the submit server has closed, NO assignments can be accepted. Non-attendance at the week 9 lab you have signed up for on the LMS page will also result in your assignment being awarded 0, except as detailed below.

Copying, Plagiarism

Plagiarism is the submission of somebody else's work in a manner that gives the impression that the work is your own. The Department of Computer Science and Information Technology treats academic misconduct seriously. When it is detected, penalties are strictly imposed. Refer to the subject guide for further information and strategies you can use to avoid a charge of academic misconduct.

Assessment Objectives

- To practise using the **Objects and Classes**.

Submission Details and marking

Full instructions on how to submit electronic copies of your source code files from your latcs8 account are given on **page 2**. All assignments in OOF are marked, face to face, in the lab, in an execution test. This means that we mark running code. Your code must compile and display a result to the screen. Regrettably, we don't have the time or resources to look at code. The smallest amount of code that produces and displays a correct result will gain more marks than lots of code that doesn't compile, run or display something to the screen. If you cannot attend the lab you have signed up for on the LMS page, please email me (m.felicetti@latrobe.edu.au) to arrange another time.

Marking summary

This assignment is worth **2%** of your final mark in this subject.
Implementation **(Execution of code) 90%, explanation of code 10%**

Instant zeros or heavily reduced marks	**Not submitting code** **Not attending marking session** **Not able to explain code that has not been taught yet** **Does not run on latcs8** **Uses System.exit()**
CricketMatch.java Player.java Team.java	**100% (90% Execution and 10% Explanation)**

Using code not taught in OOF

Please also note carefully that whilst we encourage innovation and exploring java beyond what has been presented in the subject to date, **above all, we encourage understanding**.

All of the Tasks that follow can be solved using techniques that have been presented in lectures, lecture / workshops and labs so far.

These are the techniques and knowledge that we will later be examining in the Real Time Test (20 marks) and the exam (60 marks).

Code and techniques that are outside the material presented will not be examined, of course.

For demonstrating the knowledge taught in this subject, you can only use the standard classes below and user defined classes.

String, File, Scanner, Math and System

Submitting an assignment with code outside what has been presented so far and not attending the marking execution test will result in an automatic mark of 0, regardless of the correctness of the submission.

Electronic Submission of the Source Code

- Submit all the Java files that you have developed in the tasks above.
- The code has to run under Unix on the latcs8 machine.
- You submit your files from your latcs8 account. Make sure you are in the same directory as the files you are submitting. Submit each file separately using the **`submit`** command.

```
submit OOF CricketMatch.java
submit OOF Player.java
submit OOF Team.java
```

After submitting the files, you can run the following command that lists the files submitted from your account:

verify

You can submit the same filename as many times as you like before the assignment deadline; the previously submitted copy will be replaced by the latest one.

Please make sure that you have read page 1 about the submission close off date and time and the compulsory requirement to attend the execution test in Week 4

Failure to do both of these things will result in your assignment be awarded a mark of 0, regardless of the correctness of the program.

Execution test marks are provisional and subject to final plagiarism checks and checks on the compliance of your code to this assignment document.

As such, final assignment marks may be lower or withdrawn completely.

Background - Task

This program will implement a user interface for adding teams and players, displaying information on teams and players, and playing a *simplified* game of cricket.
In this task there is a maximum of 3 teams, and each team can only have one player.

This program will have limited functionality but large portions of this will be used in Assignment D.

Background – UML diagrams and files

You will have three classes for this assignment, **CricketMatch, Team** and **Player**

The minimal UML class diagrams for each are given below

CricketMatch
- Team team1 - Team team2 - Team team3 - Scanner keyboard
+ static void main(String[] args) + CricketMatch() + void run() - void displayMenu() - void process() - void addPlayer() - void addTeam() - Team searchTeam(String teamName) - Player searchPlayer(String playerName, String playerCountry) - void removeTeam() - void displayTeam() - void displayAllTeams() - void displayPlayer() - void match()

Team
- String teamName - Player player - int gamesWon - int gamesLost - int gamesTied
+ Team(String teamName) + String getTeamName () + void addPlayer(Player player) + Player getPlayer() + int getGamesWon() + int getGamesLost () + int getGamesTied() + void incrementGamesWon() + void incrementGamesLost() + void incrementGamesTied() + String toString()

Player
- String playerName - String country - double battingSkill
+ Player(String name, String country, double battingSkill) + String getPlayerName() + String getCountry() + double getBattingSkill() + String toString()

You may add further methods and attributes if you like, but at a minimum both classes must have these methods and attributes.

Background – Base program

The CricketMatch class has been partially written for you.

This file can be copied from the csilib area:

```
cp /home/student/csilib/cse1oof/assignSum18/assignCsum18/CricketMatch.java .
```

Some methods have been written for you.
You will need to edit the main method to add your name and your student number.

Task - Solution length

The code should be around 300-400 lines of code (both files combined). If your code is significantly larger, you may want to reconsider your approach.

Player.java

The Player class contains information on the players name, country and batting skill

Create the Player class based on the UML class diagram and information below. (Not all methods explained)

toString()

The toString() method should return a String such that all attributes on a single line (see example)

Team.java

The Team class contains information on the teams name, games won, games lost, games tied and the player

Create the Player class based on the UML class diagram and information below. (Not all methods explained)

Team(String teamName)

The constructor sets the team name based on the argument passed, the games won, lost and tied are all set to the default value of 0, and the player is set to the default value of null

AddPlayer(Player player)

The addPlayer method adds a single player by assigning it to the player attribute

toString()

The toString method should return a String such that all attributes are on separate lines (see example)

CricketMatch.java

The CricketMatch class will allow the user to create, remove teams, add players, display and play a cricket match

Create the CricketMatch class based on the UML class diagram, information below and the file provided

Information:

team1, team2 and team3

Each team is of the class Team, when the CricketMatch constructor is called, all three Team object references are set to null. This Indicates that the teams have not been added yet. You can check if the team has not been added using the Boolean expression teamN == null, where N is the team (1,2 or 3). A new team instance is created by getting the team information from the user, using the menu system. The teams are added in the order 1, 2 and 3. The teams can also be removed, by setting the specific Team variable to null. The maximum number of teams that can be added is 3.

Important: Two teams cannot have the same name.
Important: Two players cannot have both the same name and country.

Hint: Do these in order.

void displayMenu()

This method should print to the screen a menu. This is already provided. Case 3 and 8 should not be added (These are for Assignment D)

void process(int choice)

This method accepts an int choice, this int contains the users choice based on the menu.
This method should call the appropriate methods based on the choice. If the user decides to exit, then no method should be called. If an invalid choice is made, the message “That is not a valid choice” is printed. This is already provided.

Team searchTeam(String teamName)

If team1 is added and has the same team name as the argument, then return the object reference for team1.
Else if team2 is added and has the same team name as the argument, then return the object reference for team2.
Else if team3 is added and has the same team name as the argument, then return the object reference for team3.
Else return null

void addTeam()

This method creates a new team instance where the attributes passed to the constructor are based on the users input.

1. The method should check if all teams are already added (1,2 and 3) and if so the program should print “Maximum teams reached” and return
2. The method should ask the team name as shown in the example outputs.
3. The method should check using the searchTeam method if there is already a team with the same name, and if there is the program should print “Team already exists” and return
4. Then the appropriate Team constructor is called and a new Team is instantiated, and set to the appropriate object reference:
    a. If all three teams are not added then team1 should be used
    b. If only team1 is added then team2 should be used
    c. And if both team1 and team2 are added then team3 should be used
5. Return to menu **(no need to call anything, just return from the method)**

Player searchPlayer(String playerName, String playerCountry)

If team1 is added and team1 has a player, and the player has the same name and country as the argument, then return the object reference for the player from team1
Else if team2 is added and team2 has a player, and the player has the same name and country as the argument, then return the object reference for the player from team2
Else if team3 is added and team3 has a player, and the player has the same name and country as the argument, then return the object reference for the player from team3
Else return null

void addPlayer()

This method creates a new Player instance where the attributes passed to the constructor are based on the users input.

1. This method first should check if there is at least one team to add a player to, if not an appropriate message should be printed and return
2. The user then is prompted to enter the players name and country
3. If a player with the same name and country already exists an appropriate message should be printed and return
4. The user then is prompted to enter the teams name
5. If the team does not exist an appropriate message should be printed and return
6. The player then is prompted to enter the batting skill, a number (double) between 0 and 1. If the user enters an invalid value, the user is prompted until they enter a valid value.
7. The appropriate Player constructor is called and a new Player is instantiated, this object reference is passed to the team using the AddPlayer method from the Team class.
8. Return to menu **(no need to call anything, just return from the method)**

void removeTeam()

This method should remove (set object reference to null) the last team loaded, if no teams have been loaded then the program should print “No team to remove”

void displayTeam()

This method first should check if there is at least one team to display, if not the message “No teams exist” is printed. The user should be asked to enter the team name, if the team exists then the program prints the String returned by the teams toString method. If the team does not exist then the program should print “Team does not exist”.

void displayAllTeams()

This method should print all the team information (using toString()) of all the teams that are added. Each team should be printed on a separate line, in the order 1, 12, or 123 (depending on the number added). If no teams are loaded, “No teams exist” should be printed.

void displayPlayer()

This method first should check if there is at least one team to display, if not the message “No teams exist”. The user should be asked to enter the player name and country (NOT THE TEAM NAME), if the player exists then the program prints the String returned by the players toString() method. If the player does not exist, then the program should print “Player does not exist”.

(Get everything else working first before attempting the below)

void match()

This method begins by getting two team names from the user. If either do not exist or either do not have a player, then an appropriate message should be printed and return.

Four innings are played. Where innings 1 and 3 are played (batted) by team1.

The team that is batting is printed at the start of each inning.
The number of runs the player (batter) made in the inning is calculated by the following:

Players skill level multiplied by a random number from Math.random() and then multiplied by 50, and the result converted into an int (using casting)

The players name is printed next to the number of runs, and the total number of runs is stored per team. When the user presses enter on the keyboard, the next inning is printed to the screen (see example)
The player that won after the 4 innings is printed with the number of runs they won by. If there is a tie, then both teams are printed to the screen.
The games won, lost and tied are incremented for each team based on the results.

Output examples

Output examples given in separate files on LMS

Final notes

There will be consultation sessions for the assignment, the times will be posted on LMS, if you have problems come to consultation.

Do a little bit every night; before you know it you will be finished. The assignment is marked with running code, so you are better to have 1 or 2 Tasks completed that actually compile and run, rather than a whole lot of code that doesn't compile.

The execution test is done on latcs8 so please make sure that your code runs on latcs8 before you submit.