Department of Computer Science and Information Technology
La Trobe University

CSE1OOF/4OOF Summer 1, 2018
Assignment Part D

10%

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

This program will have extend the functionality of Assignment C.

Due Date

Due : Thursday 14th of February 2019 at 10.00 a.m.
Execution test: Friday 15th of February 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 15th of February 2019 **– in your normal lab (Week 12).** After the submit server has closed, NO assignments can be accepted. Non-attendance at the week 12 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 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 N teams (default 6), and each team can have M players (default 11).

This program will have extend the functionality of Assignment C.

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[ ] teams - final int MAX_TEAMS = 6 - int numberOfTeams; - Scanner keyboard
+ static void main(String[] args) + CricketMatch() + void run() - void displayMenu() - void process() - void addPlayer() - void addTeam() - Team searchTeam(String teamName) - int searchTeamIndex(String teamName) - Player searchPlayer(String playerName, String playerCountry) - void addTeamFromFile( ) - void removeTeam() - void displayTeam() - void displayAllTeams() - void displayPlayer() - void setTeamBattingOrder() - void match()

Team
- String teamName - Player[] players - int[] battingOrder - int numberOfPlayers - int gamesWon - int gamesLost - int gamesTied + static final int MAX_NUMER_OF_PLAYERS = 11
+ Team(String teamName) + String getTeamName () + void AddPlayer(Player player) + Player[] getPlayers() + Player getPlayer(int index) + void setBattingOrder(int[] battingOrder) + int[ ] getBattingOrder() + int getNumberOfPlayers() + 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 getName() + 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.

Task - Solution length

The code should be around 400-550 lines of code (both files combined). If your code is significantly larger, you may want to reconsider your approach.

Player.java

The Player class requires no change from Assignment C

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, an array of players, the battingOrder, numberOfPlayers.

Further, the public static final attribute MAX_NUMBER_OF_PLAYERS is set to 11. This should be used everywhere that the maximum number of players is required. As it is public this attribute can be used in CricketMatch class. If the value is changed to any number greater than 2 and recompiled, the program should still work. This will likely be tested in the real time testing.

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
- players is initialized as an array of Player s of length MAX_NUMBER_OF_PLAYERS and then each element is set to null
- battingOrder is initialized as an array of int s of length MAX_NUMBER_OF_PLAYERS and each element is set to the values 1, to MAX_NUMBER_OF_PLAYERS
  i.e. Element at index 0 will hold 1, Element at index 1 will hold 2, and so on
- numberOfPlayers is set to the default value of 0

Team(String teamName, Player[] players, int[] battingOrder, int gamesWon, int gamesLost, int gamesTied)

The constructor sets number of players to MAX_NUMBER_OF_PLAYERS, and all other attributes based on the arguments passed.

addPlayer(Player player)

The addPlayer method adds a player to players array and increments numberOfPlayers

getPlayer(int index)

The getPlayer method returns the player at the given index

getPlayers()

The getPlayers method returns the array of Players. You must prevent the memory leak by making copy of the array.

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:

The CricketMatch class contains keyboard scanner, an array of teams and number of teams

The final attribute MAX_TEAMS is set to 6. This should be used everywhere that the maximum number of team is required. If the value is changed to any number greater than 2 and recompiled, the program should still work. This will likely be tested in the real time testing.

Important: Two teams cannot have the same name.
Important: Two players cannot have the 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.

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.

int searchTeam(String teamName)

Will return the team from the teams array of the team with a matching teamName if found
Else return null

Team searchTeamIndex(String teamName)

Will return the index of the team from the teams array of the team with a matching teamName if found
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 MAX_TEAMS are already added 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 the a new Team is instantiated and added to the team array at the end. numberOfTeams is also incremented

5. Return to menu **(no need to call anything, just return from the method)**

Player searchPlayer(String playerName, String playerCountry)

Will return the player from the players array of each team with a matching playerName and playerCountry if found
Else return null

void addPlayer()

This method creates a new track 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. If the team already has MAX_NUMBER_OF_PLAYERS then an appropriate message should be printed and return
7. 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.
8. 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.
9. Return to menu **(no need to call anything, just return from the method)**

void removeTeam()

Remove the last team added from the team array, and decrement the appropriate attribute. Display appropriate messages to the user if team can not be removed.

void displayTeam()

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 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 of all the teams that are added. Each team should be printed on a separate line, in the order 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 setTeamBattingOrder()

This method first should check if there is at least one team to set the batting order for. The user is asked for the team, and the team must have all MAX_NUMER_OF_PLAYERS else an appropriate message is to be displayed and return.
A list of all players is displayed to the screen as shown on the example output. Then the batter for each position is entered by the user. Note the batters are listed from 1 to MAX_NUMER_OF_PLAYERS , not from 0 as this is for the user to understand (not a programmer). The same goes for the batting order, from 1 to MAX_NUMER_OF_PLAYERS.
Eg. If player 4 is to be bat in position 1, then 4 will be stored in index 0 of the battingOrder array. Player 4 is in index 3 of the Players array. This may be confusing, but we need to keep it simple for the user, the complexity behind is what we need to deal with.
You also need to check that a valid player number was entered, values should be between 1 and MAX_NUMER_OF_PLAYERS, and no value should be entered twice. You can do this after reading all the values and print a message, not set the batting order and return if any invalid values were entered.
Use the appropriate method to set the batting order for the team if valid values entered.

void addTeamFromFile( )

Teams added from files must have MAX_NUMBER_OF_PLAYERS. This means that teams from file can not be partially filled with players. The file includes the the numberOfPlayers, teamName, numberOfPlayers, battingOrder, gamesWon, gamesLost, gamesTied and each players information. You will need to study the example files to decide on the appropriate order to get the information.
The overload constructor Team(String teamName, Player[] players, int[] battingOrder, int gamesWon, int gamesLost, int gamesTied) must be used.
The file always includes the number of players on the first line, so you can test if it is suitable, if not you should print an appropriate message to the screen and return.
The method should also check if MAX_TEAMS are already added and if so the program should print “Maximum teams reached” and return
The user enters the file name and will always exist and be populated correctly

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.

Two innings are played. Where innings 1 played (batted) by team1 and inning 2 by team2

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.
Every player **except the last** bats in a game (See example ouutput)

When the user presses enter on the keyboard, the next inning is printed to the screen (see example)
The team that won after the 2 innings is printed with the number of runs they won by.

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.