Department of Computer Science and Information Technology
La Trobe University

CSE4OOF Semester 1 2019
Assignment Part C

12%

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 : Wednesday 22th of May 2019.
Execution test: Week 11 - 22 May to 24 May 2019

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 in your normal lab (Week 11).** After the submit server has closed, NO assignments can be accepted. Non-attendance at the **week 11** 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 CSE4OOF 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 do not 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 does not 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 (O.Mahdi@latrobe.edu.au) to arrange another time.

Marking summary

This assignment is worth **12%** 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()**
Olympics.java Sport.java Competitor.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, Math, 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 MOF Olympics.java
submit MOF Sport.java
submit MOF Compeitor.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 11**.

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 sports and competitors, displaying information on sports and Competitors, and playing a sport in a simplified Olympics. In this task there is a maximum of 3 sports, and each sport can have 3 competitors.

Sports can have more than one Competitor but must the required number of competitors before competing.

It is not possible to have a competitor without there being a Sport first. It is possible to have a Sport without a Competitor.

When a Sport is first created it has no Competitors.

When a Sport is played Competitors can earn gold medals, the number of medals is stored for each competitor. The Competitor that comes first gets one gold medal.

Each sport has individual competitors from various countries compete. There are no team sports in this problem.

Background – UML diagrams and files

You will have three classes for this assignment, **Olympics, Sport** and **Competitor**

The minimal UML class diagrams for each are given below

Olympics
- Sport sport1 - Sport sport2 - Sport sport3 - Scanner keyboard
+ static void main(String[] args) + Olympics() + void run() - void displayMenu() - void process() - void addCompetitor() - void addSport() - Sport searchSport(String sportName) - void removeSport() - void displaySport() - void displayAllSports() - void compete()

Sport
- String sportName - Competitors competitor1 - Competitors competitor2 - Competitors competitor3
+ Sport(String sportName) + void addCompetitor(Competitor Competitor) + Competitor getCompetitors(int competitorNumber) + String getSportName() + String toString()

Competitor
- String competitorName - String country - double Skill
+ Competitor(String name, String country, double Skill) + String getCompetitorName() + String getCountry() + double getSkill() + void incrementNumberOfGoldMedals() + int getNumberOfGoldMedals() + 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 Olympics class has been partially written for you.

This file can be copied from the csilib area:

```
cp /home/student/csilib/cse4oofbu/assignBu19/assignC/Olympics.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 400-550 lines of code (both files combined). If your code is significantly larger, you may want to reconsider your approach.

Competitor.java

The Competitor class contains information on the Competitors name, country and skill

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

Competitor(String competitorName, String country, double skill)

The attributes are set based on the arguments passed. The number of gold medals is set to 0.

toString()

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

Sport.java

The Sport class contains information on the sports name, an array of Competitors, the required number of competitors, and the actual number of Competitors.

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

Sport(String sportName)

The constructor sets:

- the sport name based on the argument passed
- Each Competitor is set to null

addCompetitor(Competitor Competitor)

The addCompetitor method adds a Competitor to the sport. The competitor1 should be added first, then competitor2 and then competitor3.

getCompetitor(int competitorNumber)

If competitorNumber is 1, then return competitor1, if 2 then competitor2, if 3 then competitor 3. If any other number is passed, then return null.

toString()

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

Olympics.java

The Olympics class will allow the user to create, remove sports, add competitors, display and run a sport competition

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

Information:

sport1, sport2 and sport3

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

Important: Two sports cannot have the same name.
Important: Two Competitors 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. Options 3, 6 and 8 will be used in Assignment D (Do not add them now)

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 searchSport(String sportName)

If sport1 is added and has the same sport name as the argument, then return the object reference for sport1.
Else if sport2 is added and has the same sport name as the argument, then return the object reference for sport2.
Else if sport3 is added and has the same sport name as the argument, then return the object reference for sport3.
Else return null

void addSport()

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

1. The method should check if all sports are already added (1,2 and 3) and if so the program should print "Maximum sports reached" and return
2. The method should ask the sport name as shown in the example outputs.
3. The method should check using the searchSport method if there is already a sport with the same name, and if there is the program should print "Sport already exists" and return
4. Then the appropriate Sport constructor is called and a new Sport is instantiated, and set to the appropriate object reference:
    a. If all three sports are not added then sport1 should be used
    b. If only sport1 is added then sport2 should be used
    c. And if both sport1 and sport2 are added then sport3 should be used
5. Return to menu **(no need to call anything, just return from the method)**

void addCompetitor()

This method creates a new competitor 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 sport to add a Competitor to, if not an appropriate message should be printed and return
2. The user then is prompted to enter the Competitors name and country
3. The user then is prompted to enter the sports name
4. If the sport does not exist an appropriate message should be printed and return
5. The user then is prompted to enter the skill value, a number (double) between 0 and 1. If the user enters an invalid value, the user is prompted until they enter a valid value.
6. The appropriate Competitor constructor is called and a new Competitor is instantiated, this object reference is passed to the sport using the addCompetitor method from the Sport class.
7. If the addCompetitor method returns false, then the competitor could not be added and a message should be printed
8. Return to menu **(no need to call anything, just return from the method)**

void removeSport()

This method should remove (set object reference to null) the last sport loaded, if no sports have been loaded then the program should print "No sport to remove"

void displaySport()

This method first should check if there is at least one sport to display, if not the message "No sports exist" is printed. The user should be asked to enter the sport name, if the sport exists then the

program prints the String returned by the sports toString method. If the sport does not exist then the program should print “Sport does not exist”.

void displayAllSports()

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

(Get everything else working first before attempting the below)

void compete()

This method implements the competition of the sport between competitors. All three competitors are required to compete.

1. Ask the user for the name of the sport
2. If the sport does not exist or all the competitors of the sport have not been added print an appropriate message and return
3. The sports name is printed to the screen
4. The score of each competitor is calculated by taking the competitors skill multiplying it by Math.random() and then multiplying by 50, and converted to an int.
5. The highest of the three scores is the competitor who won the gold medal, the name and country needs to be printed for this competitor. The number of gold medals for the winning competitor needs to be incremented.

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 does not compile.

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