RMIT UNIVERSITY

School of Science

CPT121 / COSC2135 Programming 1 (OUA)

Assessment 2: Assignment 2

Assessment type: Individual assignment; no group work.

Word limit: N/A

Due date: Sunday ending Week 8 (i.e. 25th April 2021) 11:59PM (AEST)

As this is a major assignment in which you demonstrate your understanding, a university standard late penalty of 10% per each working day applies for up to 5 working days late, unless special consideration has been granted.

Weighting: 20%

Overview

This assessment requires you to apply various concepts and techniques covered in weeks 1–8 of the course, in order to assemble a programmatic solution for a problem based on a simulated real-world scenario.

An important part of your development as a programmer is the ability to identify and apply appropriate techniques or programming structures when implementing the various aspects of a programmatic solution to a given problem, so in addition to the basic functional requirements of this task, you will also be assessed on your ability to select and utilise appropriate techniques and structures in your solution.

This assessment requires you to design and create a Java program to solve analysed stakeholder (client and development-team manager) requirements.

Assessment grading criteria

This assessment will measure your ability to:

- Apply the specified coding and documentation style guide
- Apply basic features of Java and Object-Oriented techniques (covered in weeks 1-8 of the course material) to solve the user and development-team manager requirements
- Produce correct output on provided test data

Course learning outcomes

This assessment is relevant to the following course learning outcomes:

- CLO 1: Solve simple algorithmic computing problems using basic control structures and Object-Oriented Techniques
- CLO 2: Design and implement computer programs based on analysing and modelling requirements
- CLO 3: Identify and apply basic features of an Object-Oriented programming language through the use of standard Java (Java SE) language constructs and APIs
- CLO 4: Identify and apply good programming style based on established standards, practices and coding guidelines
- CLO 5: Devise and apply strategies to test the developed software

RMIT UNIVERSITY

Assessment details

'Taradale Folk Festival' (TFF) is a (fictional) biennial arts festival based around the central Victorian township of Taradale. To try and provide scheduling flexibility in case of Covid-19 restrictions, the festival organisers have split the festival program into two types of event: unrestricted open air events requiring a 'venue' pass and indoor events requiring an 'experience' pass. It is hoped that if Covid restrictions prohibit indoor events, the outdoor events could continue since attendees are outdoors and could be spaced out further.

Your task is to produce, using standard Java (Java SE), a prototype record keeping system for tracking the events that are on offer and accepting bookings. For this initial proof of concept, the system will be console based.

Interviews with two TFF staff are given below: Huy, the General Manager, and Tina, the Ticketing and Front of House Manager. You should analyse the interviews to determine the functional requirements for your program. Your development team manager has provided further requirements. These requirements are set out in the three stages (*A*, *B* and *C*) below. Since each stage elaborates on the previous one, you only need to submit one program – the one implementing the highest stage you were able to complete.

For each of the stages A, B and C, complete the steps below based on the Double Diamond process:

Discover

'Go wide' in the form of reading and thinking about the supplied stakeholder requirements. Find out what the current situation is and understand user behaviours and business drivers.

Fill in the supplied 'Discover and Define' Word template with a list of different problems that could be solved.

Define

From your understanding of the overall issues/requirements, narrow down to a single problem and turn that into a problem statement. The problem statement defines what you will develop, and you may start some initial coding to start working out if you can create a solution to the problem you defined.

Fill in the supplied 'Discover and Define' Word template with a statement of the single problem you will solve.

Develop

Start coding to create an initial prototype and program logic to address the problem. You may create a few different iterations to get to the best prototype. This is a phase for trying ideas out to see if they work.

Deliver

Pick the best prototype from the 'Develop' phase and create the final version of the code, refining and making it work. The aim is to create a minimum viable product (MVP) to address the problem you have identified and defined.

Coding Style/Documentation

Your program must follow the coding and documentation style guide given in the file *StyleGuide.pdf*, available on the course Canvas under the *Assignment 2* link.

Stage A – Modelling and implementing classes to represent and manage open-air venue events

Your task here is to discover and define the problem for stage A and then develop a Java program which implements the functional requirements derived from the interviews and instructions given below.

Huy – TFF General Manager

> *"As a proof of concept, I'd like the prototype at this stage to model outdoor 'Venue' events. I'd like the prototype on start-up to ask what the maximum number of events can be, and then be capable of storing up to that many events (giving an error message if I try to exceed that limit).*
>
> *I'll need to be able to enter details about the event that is being added, be able to display the titles of all events, and be able to display all details of the event I specify. These details are: the event title, description and prices of Adult tickets, Child tickets and Concession tickets. The system should keep track of the total number of tickets sold for each event.*
>
> *My ticketing staff will also need to be able to accept customer bookings and produce confirmation tickets."*

Tina – Ticketing and Front of House Manager

> *"When an attendee wishes to book to attend an event, I find out the required event title and how many tickets are required. For each required ticket, I then find out the required ticket type (Adult, Child or Concession), record that another ticket has been sold and prepare a ticket to give to the attendee stating the name of the attendee, the event title, the ticket type and the cost of the ticket."*

Your development-team manager advises that your program must also meet the following requirements:

- The program must be console driven (i.e. use *Scanner* for input).
- Displayed output must be neatly formatted.
- ArrayLists, Java Collections Framework or other material not covered in weeks 1..8 can't be used.
- Your program for this stage should be implemented as two classes:
    - *TffEvent* stores and processes information relevant to a TTF venue event, including:
        - The void method *displayEvent()* for displaying all instance variable values, neatly formatted.
        - The boolean method *bookEvent(String ticketType, String name)* where *ticketType* indicates the type of the ticket being booked ("Adult", "Child" or "Concession"), and *name* is the name of the attendee. This method should print ticket details and manipulate the count of tickets sold to indicate that a booking has occurred. The

method returns *true*.

- *StageA* implements the *main* method. This class collects all user input and uses an array of references to objects of type *TffEvent* called *tffEvents* to help implement the functionality of stage A.

- Your program must use the following array (not ArrayLists) for the following purpose:
  - *tffEvents* : a 1D array of *TffEvent* references.
- Your program should follow the style guides given in *StyleGuide.pdf* on the Assignment 2 page.

You are expected to adhere to all relevant object-oriented programming guidelines, including:

- Visibility of instance variables and methods set appropriately
- Instance variable initialisations carried out in the constructor only
- No unnecessary accessors (setters) or mutators (getters) - only provide methods which will be needed when implementing the application class in this stage
- Parameter lists in methods should be appropriate to the task the method is performing - only accept parameters where a method requires one or more values from the caller to perform its assigned task that it does not already have access to
- Methods which need to communicate a value or result back to the caller should do by returning the value in question, not by storing it in an instance variable
- Data classes (i.e. *TffEvent*) should not prompt for input from the user

Stage B – Implementing refundable indoor 'experience' events

Your task here is to extend your console-driven Java program to implement the functional requirements derived from the edited interviews and instructions given below.

Huy – TFF General Manager

> *"In addition to our outdoor 'venue' events, our festival also hosts indoor 'experience' events. Since they are held indoors, each of these events has their own limit on the total number of tickets. I should be able to enter this information when creating the event, and it should be displayed when displaying the event. Unlike the 'venue' events, 'experience' events should also be refundable."*

Tina – Ticketing and Front of House Manager

> *"I now need to be able to book 'experience' events. I'll need the system to produce an appropriate error message if a booking would exceed the available number of tickets for the event.*
>
> *Attendees can also have their experience booking refunded. This involves them giving me their ticket, from which I obtain the event title to determine the event to refund and I then update the record of the total number of tickets booked for that event. Now this is being automated, I'd expect the system to print out a confirmation that the refund has been successful, or an error message if when there has been a problem."*

Your development-team manager advises that your program must meet all stage B requirements, with the following changes/additions:

- A new *TffExperienceEvent* class must extend the *TffEvent* class. This new class must implement the requirements of the 'experience' event.
- *TffExperienceEvent* overrides boolean method *bookEvent(String ticketType, String name)*.This method should make appropriate use of the *super* construct. The method should return *true* if a booking was successfully performed, and *false* if a booking couldn't be successfully completed.
- *TffExperienceEvent* overrides method *displayEvent*, to print out all information about the *TffExperienceEvent*. The method should make appropriate use of the *super* construct.
- Class *StageA* should be renamed as *StageB* and modified to support the requirements of stage B. This class must use the array of references to objects of type *tffEvent* (called *tffEvents*) to store references to all *TffEvent* **AND** *TffExperienceEvent* objects in the system, and use these objects and polymorphism appropriately for the functionality of stage B.

Stage C – Implementing attendee lists for refundable indoor 'experience' events

Your task here is to extend your console-driven Java program to implement the functional requirements derived from the edited interviews and instructions given below.

Huy – TFF General Manager

> *"We need to keep a list of the names of ticket holders for a 'experience' events and the price they paid, so we can verify they hold a ticket and refund the correct amount in case they don't have their ticket available when they claim a refund. I'd like the list displayed as part of displaying an 'experience' event. The ticket holder's record should be deleted once they have been refunded."*

Your development-team manager advises that your program must meet all stage B requirements, with the following changes/additions:

- Class *TffExperienceEvent* must utilise a 1D String array instance variable, called *bookings*, to store ticket booking details (i.e. the name list of ticket holder name and price paid). It can be assumed that a ticket holder's name will be unique.
- Class *StageB* should be renamed as *StageC* and modified to support the requirements of stage C. This class must still use the array of references to objects of type *tffEvent* (called *tffEvents*) to store references to all *TffEvent* **AND** *TffExperienceEvent* objects in the system, and use these objects and polymorphism appropriately for the functionality of stage C.

Referencing guidelines

What: This is an individual assignment and all submitted code must be your own. If you have used examples of code from sources other than the contents directly under the course Canvas Modules link, or the recommended textbook (e.g. an example from the web or other resource) as the basis for your own code then you should acknowledge the source(s) and give references using the IEEE referencing style.

Where: Add a block code comment near the work to be referenced and include the reference in the IEEE style.

How: To generate a valid IEEE style reference, please use the citethisforme tool if unfamiliar with this style.

Submission format

Submit your Java code file(s) on the course Canvas under the Assignments/Assignment2 link as a file upload.

Marks are awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums on the course Canvas.

Academic integrity and plagiarism

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas.

You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct.

Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the University website.

Assessment declaration

When you submit work electronically, you agree to the assessment declaration.

RMIT UNIVERSITY

Criteria	Ratings						Pts
Code style/documentation	All three *Style Guide* guidelines were implemented, no areas of improvement were identified	All three *Style Guide* guidelines were implemented, minor issues noted with one guideline	All three *Style Guide* guidelines were implemented, minor issues noted with two guidelines	All three *Style Guide* guidelines were implemented, significant issues noted with one guideline	Fewer than three *Style Guide* guidelines were implemented, or major issues were noted with more than one guideline	The *Style Guide* guidelines were not followed	
	1.5 pts	**1.2 pts**	**0.9 pts**	**0.6 pts**	**0.3 pts**	**0.0 pts**	**1.5**
TffEvent Class	TffEvent correctly implemented the requirements of 'venue' events for stage A/B/C.		TffEvent had minor issues implementing requirements of 'venue' events for stage A/B/C.		TffEvent has significant issues implementing requirements of 'venue' events for stage A/B/C.	Functionality not implemented	
	1.5 pts		**1.0 pts**		**0.5 pts**	**0.0 pts**	**1.5**
StageA/B/C class	StageC correctly uses array *TffEvents* to store and reference TffEvent and TffExperienceEvent objects, as specified. Collects user data as appropriate. Uses TffEvent and TffExperienceEvent objects and polymorphism to implement Stage C functionality appropriately.	StageB correctly uses array *TffEvents* to store and reference stage B TffEvent and TffExperienceEvent objects, as specified. Collects user data as appropriate. Uses TffEvent and TffExperienceEvent objects and polymorphism to	StageB has minor issue with one of : correctly using array *TffEvents* to store and reference stage B TffEvent and TffExperienceEvent objects, as specified, collecting user data as appropriate, using TffEvent and TffExperienceEve	StageA correctly uses array *TffEvents* to store and reference TffEvent objects, as specified. Collects user data as appropriate, uses TffEvent objects to implement Stage A functionality appropriately.	StageA has a minor issue with one of: correctly uses array *TffEvents* to store and reference TffEvent objects as specified, collects user data as appropriate, uses TffEvent objects to implement Stage A functionality appropriately.	Not implemented or major issue with StageA using array *TffEvents* to store and reference TffEvent objects as specified, or collecting user data as appropriate, or using TffEvent objects to implement Stage	

		implement Stage B functionality appropriately.	nt objects and polymorphism to implement Stage B functionality appropriately.			A functionality appropriately.	
	1.0 pts	**0.7 pts**	**0.5 pts**	**0.3 pts**	**0.2 pts**	**0.0 pts**	**1.0**
TffExperienceEvent class (stage B/C only)	TffExperienceEvent extends TffEvent, using inheritance correctly, correctly implements ticket limit, uses array *bookings* appropriately, modified *bookEvent* and *displayEvent* and uses *super* as per stage C specifications, methods correctly implemented to enable refund of ticket.	TffExperienceEvent extends TffEvent, implements ticket limit and enables refund of ticket. Minor issues with using array *bookings* appropriately, or modifying *bookEvent* and *displayEvent* and uses *super* as per stage C specifications.	TffExperienceEvent extends TffEvent, correctly implements ticket limit and enables refund of ticket. Methods *bookEvent* and *displayEvent* modified and uses *super* as per stage B specifications.	TffExperienceEvent extends TffEvent. Significant issue with correctly implementing ticket limit or refund or with modifying *bookEvent* and *displayEvent* modified and using *super* as per stage B specifications.	TffExperienceEvent extends TffEvent. Doesn't correctly implement ticket limit.	Functionality not implemented or TffExperienceEvent doesn't extend TffEvent.	
	2.0 pts	**1.5 pts**	**1.0 pts**	**0.5 pts**	**0.1 pts**	**0.0 pts**	**2.0**
Stage A/B/C functionality	Correctly implements all functionality of stage C.	Stage C submission with minor issue regarding one item of stage C functionality.	Significant issue with one item of stage C functionality, or a stage C/B submission that correctly implements all functionality of stage B	Stage B submission with minor issue regarding one item of stage B functionality or a stage A submission that correctly implements all functionality of stage B	Stage A submission with minor issue regarding one item of stage A functionality.	Functionality not implemented	

	2.5 pts	**2.0 pts**	**1.5 pts**	**1.0 pts**	**0.5 pts**	**0.0 pts**	**2.5**
Object-oriented technique	Object-oriented guidelines were correctly implemented		Minor issues were noted on one object-oriented guideline.		Minor issues were noted on two or more object-oriented guidelines, or a major issue noted on one.	Significant issues were noted on two or more object-oriented guidelines.	
	1.5 pts		**1.0 pts**		**0.5 pts**	**0.0 pts**	**1.5**

Total: **10 pts**