COMP1161 PROJECT PART 1

Due February 25, 2023.

OVERVIEW

The **Ministry** of Transport of the hypothetical island of Jamrock is convinced that a managed, on-demand bus service is essential to for the country to function, given the heavy reliance of it's citizens on unreliable "robots" , which transport persons without authorization. **Ministry** protocols have therefore been established that regulate operation of the on-demand service. The current protocols that require your attention as a programmer, involve the implementation of a **ApprovalRequest** process for various types of **Bus**es. Use of the process requires that a **Planner** who intend to organize **Trip**s identify an appropriate **Bus** and then instruct the **Bus** crew to submit a request to the **Ministry** of Transport for approval. If the approval is successful, the **Planner** pays the **Bus** for the running the **Trip**. Your immediate task is to implement the detailed design for classes **Date**, **SportBus**, **TrainingBus** and **PartyBus**, then complete some logic inside **Planner**.

SUBMISSION INSTRUCTIONS

Your submission will be made on OurVLE, . All files in your submission will be placed in a directory of the name <yourIDNumber>.Project1 and then the directory will be zipped. The zip file will be submitted. As an example, if your ID number is 620011011, the zip file should be named **620011011.Project1.zip**

==========

PART A. Business Rules

==========

The main entities driving the process are instances of **Ministry**, **Bus**, and **Planner**. A **Planner** plans a **Trip**, then assesses **Bus**es for suitability, availability, and affordability. A request to assign the **Trip** to lowest cost suitable **Bus** is then submitted to the **Ministry**, and is approved if the daily maximum number of travellers would not be exceeded(for the comfort level). If the **Ministry** approves the request, an approval Id is returned to the **Planner**, otherwise the value -1 is returned, and a message printed. The process uses a simple implementation of a **Date**.

==========

PART B. Starting code

==========

Code the following classes have been provided: **Bus, Ministry, ApprovalRequest, Trip, Planner,** and **Date**. Some methods are not fully implemented. Incomplete methods are described in section C. Methods available are described below

THE BUS CLASS

The public methods required of the Bus class involve

1. **Bus()** - A default constructor
2. **Bus(String name, double size, double basePrice, int lev)** – Initializes instance.
3. **boolean available(Date date)-** Returns whether the Bus is available on a specified date. A bus can run one trip per day in this model. Availability is determined by looking through the list of approved trips to see if the date on any trip matches the date in the argument.
4. **int getId()** - Returns the id of the bus

5. **String getName()** – Returns the name of the bus
6. **double getSize()** – Returns the size of the bus
7. **boolean isSuitable(String type)** – Returns true if the bus is suitable for a trip type.
8. **String toString()** – Return the name of the bus.
9. **int getEstimate(String type, int numPersons, int level)** – Returns the base price.
10. **boolean canHold(int numPassengers, int comfortLevel)-** Returns true if the bus can hold the specified number of passengers at the desired comfort level.
11. **void promoteTrips()** – Prints a header for the Bus followed by the toString method of each approved **Trip**s for the Bus
12. **int reserve (Trip trip,double availBal, Ministry mny)** – attempts to reserve a trip with the Ministry. If the reservation is successful, a positive approval ID , obtained from the Ministry is returned. If not the method returns -1.

THE MINISTRY CLASS

A **Ministry** exposes one public method, namely checkApproval(), which has the following behaviour: **int checkApproval(ApprovalRequest ar)** assesses an approval request to see if an trip should be run based on prevailing rules, and returns the id of the approval request if it is determined that the trip should be run. If the trip should not be run the method returns -1.

The method determines the maximum number of travelers for the day based on the comfort level and returns a positive integer if the trip would not result in the maximum being exceeded. Otherwise the method returns -1.

THE APPROVALREQUEST CLASS

The ApprovalRequest class exposes the following public methods:

1. **ApprovalRequest(Trip trip, Bus bus)** – instantiates an approval request, and assigns a reference to an Trip and a Bus. It then assigns a unique consecutively increasing id. The first id has value 0.
2. **int getId()** – returns the id of the ApprovalRequest
3. **Trip getTrip()** – returns the Trip associated with the ApprovalRequest
4. **Bus getBus()** – returns the Bus associated with the ApprovalRequest

THE TRIP CLASS

The Trip class manages trip information

Public methods available include

1. **Trip(String name, String type, int numPeople, Date date**)- Initializes a local instance of a trip by storing submitted data
2. **String getName()** – returns the name
3. **String getType()** – returns the associated type of trip
4. **int getNumPeople()** – returns the number of people who are expected on the trip
5. **Date getDate()** – returns the date of the trip
6. **Bus getBus()** – returns the bus to be used for the trip –returns null of no bus is assigned.
7. **void setBus(Bus b)** – sets the bus associated with the trip to b
8. **String toString()** – returns a string of the format
   "Date:"+**this**.getDate()+";Name:"+**this**.getName()+";Type:"+**this**.getType()+";Bus:"+**this**.getBus()+":numPeople:"+**this**.getNumPeople();

THE PLANNER CLASS

The Planner class represents an trip organizer who comes up with a trip idea and initiates the process of getting it approved. Public methods include

1. **Planner (String name, double budget, Ministry mny, Bus[] buses)** – Initializes an instance of a promoter, by setting a name and a budget. An association to the ministry as well as set of buses is also established.
2. **double getBudget()** – returns the budget associated with the planner
3. **String getName()** – returns the name associated with the planner
4. **void payFor(Bus bus, Trip trip , int numPassengers, int comfortLevel)** –Pays for the bus and reduces the budget by the cost
5. ******int planTrip(minBus,tripType, numPassengers, comfortLevel)** – METHOD INCOMPLETE. CODE TO BE WRITTEN DESCRIBED IN SECTION C.

THE DATE CLASS

Most code for the date class is to be implemented as described in section C.

THE DRIVER CLASS (Code provided)

The Driver class contains a main method that can be used to test the classes described above. The version provided instantiates a set of Planners, an instance of Ministry, and a set of Buses. It then allows planners to plan trips of various types and sizes over a period of five days..txt

==========
PART C. Code you are required to write
==========

You are required to write code to complete the classes **Date** and **Planner**. You will also implement the classes **SportBus, TrainingBus,** and **PartyBus**. Do not change the given starting code for other given classes. Otherwise you are free to select the implementation strategies that meet the requirements.

THE DATE CLASS*(5 marks)*

The Date class, used to represent a date in the system, exposes the following public methods

1. **Date(int day)** – instantiates a Date object, by setting it's internal value to day
2. **void setDay(int day)** set's the internal value to day
3. **int getDay()** – returns the internal value
4. **Date next(int day)** – returns a date corresponding to argument incremented by 1
5. **String toString()** – returns a string representation of the internal value

(Hint if you haven't picked it up yet – it's ok to use a simple integer to represent the internal value- at least for this implementation)

THE SPORTBUS CLASS *(15 marks)*

The **SportBus** class is a specialized **Bus.**

It stores private integer data items that represent competitorArea and spectatorArea, as well as an ArrayList of strings, each of which represent a sport for which the bus will be configured.
Methods of SportBus are as follows

1. **SportBus(String name, int basePrice, int lev, Ministry mny, int competitorArea, int spectatorArea, String sportList)**
   – initialize instance
   -Set the size of the associated parent class is as the sum competitorArea+spectatorArea;
   - append " ,SPORT" to the protected tripTypes attribute of the superclass
   - split the sportList into an array, then populate the list of sports from the array *(Hint. splitting a delimited string can be achieved using String.split())*
2. **double getCompetitorArea()** – returns the area available for competitors
3. **double getSpectatorArea()** – returns the area available for spectators
4. **int getBasePrice()** – returns 3 * the base price of the superclass
5. **int getEstimate(String type, int numPersons, int level)** – returns an estimated cost to run the trip. The cost is calculated as getBasePrice() * 10 * (competitorArea+spectatorArea) /competitorArea.

THE TRAININGBUS CLASS*(15 marks)*

The **TrainBus** class is a specialized **Bus.**
It stores private integer data items that represent teacherArea, studentArea and wifiRange, as well as an ArrayList of strings, each of which represent a course for which the bus will be configured.
Methods of **TrainingBus** are as follows

1. **TrainingBus (String name, int basePrice, int wifiRange, int lev, Ministry mny, int teacherArea, int studentArea, String sportList)**
   –initialize instance–
   -Set the size of the associated parent class is as the sum teacherArea + studentArea;
   - append " ,TRAINING" to the protected tripTypes attribute of the superclass
   - split the sportList into an array, then populate the list of sports from the array *(Hint. Splitting a delimited string can be achieved using String.split())*
2. **double getTeacherArea()** – returns the area available for teachers
3. **double getStudentArea()** – returns the area available for students
4. **int getBasePrice()** – returns 2 * the base price of the superclass
5. **int getEstimate(String type, int numPersons, int level)** – returns the 5 x teacherArea x the getEstimate method of the superclass, divided by the number of courses managed.

THE PARTYBUS CLASS*(40 marks)*
Some persons were concerned over whether a ministry should be getting involved with an activity that could be considered non-productive. To appease detractors, the ministry came up with a solution that every party should have a purpose. The effect on your implementation is that your **PartyBus** will be EITHER a specialized **SportBus** or a specialized **TrainingBus**. Note also that the Ministry of Transport wants to evaluate multiple potential solutions, so has prescribed that the pool of programmers (which include you), submit a solution that meets the following criteria:

1. If the first letter of your surname is M or smaller, a **PartyBus** is a specialized **SportBus**.
2. If the first letter of your surname is N or larger, a **PartyBus** is based on a **TrainingBus**.

Please be very careful to submit an appropriate solution. Failure to do so will produce an automatic zero for this section.

The PartyBus has attributes that represent a barArea, a foodArea and a securityCrew. It' s constructor initializes the instance, and appends ", PARTY" the protected tripTypes attribute. The PartyBus ~~overloads~~ overrides the getEstimate method. Again, the ministry seeks to assess various models.

Your calculation for the getEstimate method will be determined by the last digit of your UWI student ID number. Please note the calculations in the table below and implement the appropriate one for your solution.

Last ID digit	Calculation for PartyBus Estimated value
0	π x barArea x superclass getEstimate(type, numPersons, level)/foodArea
1	7 x superclass getEstimate(type, numPersons, level)/foodArea
2	foodArea x superclass getEstimate(type, numPersons, level)/(3 x barArea)
3	π x $barArea^2$ x superclass getEstimate(type, numPersons, level)/(foodArea x numPersons)
4	barArea x superclass getEstimate(type, numPersons, level)/(π x foodArea));
5	(comfortLevel+1) x barArea x superclass getEstimate(type, numPersons, level)/foodArea
6	11 x square root of(barArea) x superclass getEstimate(type, numPersons, level)/foodArea
7	Superclass getEstimate(type, numPersons, level)/(π x $barArea^3$)
8	5 x square root of(foodArea) x superclass getEstimate(type, numPersons, level)/barArea
9	π x numPersons x superclass getEstimate(type, numPersons, level)/(foodArea*barArea)

Please be very careful to submit an appropriate solution. Failure to do so will produce an automatic zero for this section.

COMPLETING THE PLANNER CLASS*(25 marks)*

The **int planTrip(int numPassengers, String tripType, Date date, int comfortLevel)** method of Planner plans an trip of a given type with the specified number of passengers on the specified date. Steps include:

a. Determine the set of buses that the promoter can afford
b. Find the lowest cost Bus from the set of affordable buses that
    i. can hold the number of passengers and
    ii. is available
c. If there is at least one Bus that meets the requirements
    i. Create an instance of a Trip
    ii. Call the reserve method of the Bus and capture the return value in an integer
    iii. If the return value is greater than or equal to 0,
        1. Call the setBus method of the trip to the selected bus
        2. Pay for the selected bus
        3. Return the return value as the result of planTrip
d. If plans for the trip did not work out, return -1