Concordia University
Comp 249 - Winter 2017
Object-Oriented Programming II
Assignment 2 – Due 11:59 PM - Friday February 17, 2017

Purpose: The purpose of this assignment is to practice class inheritance, and other Object Oriented Programming concepts such as constructors, access rights, method overriding, etc. The assignment would also allow you to practice the notion of package creation. This assignment contains two parts. You need to complete part I to be able to do part II.

Part I

Various publications can be described as follows:

- A **PublicTransportation** class with the following: *ticket price* (double type) and *number of stops* (int type).
- A **CityBus** is a **PublicTransportation** that in addition has the following: an *route number* (long type), an *begin operation year* (int type), a *line name* (String type), and *driver(s)name* (String type).
- A **Tram** is a **CityBus** that in addition has the following: *maximum speed* (int type), which indicates the maximum speed that this Citybus could reach.
- A **Metro** is a **CityBus** that in addition it has the following: *number of vehicules* (int type) and *the name of the city* (String type), such as Montreal, Toronto, etc.
- A **Ferry** is a **PublicTransportation** that in addition has the following: *build year* (int type) and *Ship name* (String type).
- An **AirCraft** is a **PublicTransportation** that in addition has the following: *class type* (enumeration type that can be: *Helicopter*, *Airline*, *Glider* or Balloon), and *maintenance type* (enumeration type that can be: *Weekly*, *Monthly*, or *Yearly*).

Part I:

1. Draw a UML representation for the above mentioned classes hierarchy. Your representation must also be accurate in terms of UML representation of the different entities and the relation between them. You must use a software to draw your UML diagrams (no hand-writing/drawing is allowed).

2. Write the implementation of the above mentioned classes using inheritance and according to the specifications and requirements given below:

   - You must have 4 different Java packages for the classes. The first package will include the **PublicTransportation** class. The second package will include the **CityBus**, **Tram** and **Metro** classes. The third package will include the **Ferry** class and the last package will include the **AirCraft** class.

- For each of the classes, you must have at least three constructors, a default constructor, a parametrized constructor (which will accept enough parameters to initialize ALL the attributes of the created object from this class) and a copy constructor. For instance the parametrized constructor of the **Tram** class must accept 7 parameters to initialize the *price*, the *number of stops*, the *route number*, the *begin operation year*, the *line name*, the *driver(s)name*, and *maximum speed.*

- An object creation using the default constructor must trigger the default constructor of its ancestor classes, while creation using parametrized constructors must trigger the parametrized constructors of the ancestors.

- For each of the classes, you must include at least the following methods: accessors, mutators, ***toString()*** and ***equals()*** methods (notice that you are always overriding the last two methods).

  - The ***toString()*** method must display clear description and information of the object (i.e *"This Tram has 23 stops, and costs 17$. It's maximun speed is 70km/h….."*).

  - The ***equals()*** method must first check if the passed object (to compare to) is null and if it is of a different type than the calling object. The method would clearly return false if any of these conditions is true; otherwise the comparison of the attributes are conducted to see if the two objects are actually equal. You need to add a comment indicating how effective these null verifications inside the method will be in relation to protecting your program from crashing!

- For all classes, with the exception of the **CityBus** class, you are required to use the appropriate packages access rights. For the **CityBus** class for example, you are required to use protected access rights.

- When accessing attributes from a base class, you must take full advantage of the permitted rights. For instance, if you can directly access an attribute by name from a base class, then you must do so instead of using a public method from that base class to access the attribute.

3. Write a driver program (that contains the main() method) that would utilize all of your classes. The driver class can be in a separate package or in any of the already existing four packages. In the main() method you must:
   - Create various objects from the 6 classes, and display all their information using the ***toString()*** method.
   - Test the equality of some to the created objects using the ***equals()*** method.
   - Create an array of 10 **PublicTransportation** objects and fill that array with various objects from the 6 classes (each class must have at least one entry in that array).
   - Trace(search) that array to find the object that is least expensive (has lowest price) and the one that is most expensive. Display all information of these two objects along with their location (index) in the array.

Part II

For the requirements of this part, you need to modify your implementation from Part I as follows:

1. The classes from part I, must now have the most restrictive (secure/protective) access rights to their attributes. Adjust your implementation from Part I accordingly, and comment on the decision about using restricted access (i.e. what are the tradeoffs).

2. In the driver program of this part, you need to add to the one from part I, another static method (added it above the main()method), called ***copyCityBuss***. The method will take as imput an array of **PublicTransportation** (an array of any size) and returns an array of **PublicTransportation**. That is to say, the method needs to create an array of the same length as the passed paramter one, copy all CityBuss from the passed array to a new array then return it. Your copy of the objects must use the copy constructors of the different listed classes.

3. In the driver program, create an array of 12 objects (must have at least one from each of the classes), then call the *copyCityBuss()* method to create a copy of the that array.

4. Display the contents of both arrays, then add some comments indicating whether or not the coping is correct. If not; you need to explain why it has not been sucessful or as you might expected.

General Guidelines When Writing Programs

- Include the following comments at the top of each class you are writing.

```
// -------------------------------------------------
// Assignment #2
// Part: (include Part Number)
// Written by: (include your name(s) and student ID(s))
// -------------------------------------------------
```

- When commenting your code provide on the top a general and clear explanation of what the piece of code is doing; and within that piece of code if there is any method or any loop specify briefly what is doing. Include comments as needed.
- Display clear prompts for the user whenever you are expecting the user to enter data from the keyboard.
- All outputs should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

Submission & Demo

When you have finished the program, you must submit the assignment online to the EAS system.

1) Create **one** zip file, containing the two parts: I and II separately, where each part contains the files (.java and .html and test cases).

Please name your file following this convention:
If the work is done by 1 student: Your file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID number.
If the work is done by 2 students, ONLY ONE of the members can upload the file (do not upload twice): The zip file should be called *a#_studentID1_studentID2*, where # is the number of the assignment *studentID1* and *studentID2* are the student ID numbers of each student.

2) Assignments must be submitted in the right folder of the assignments. Upload your zip file at the URL: https://fis.encs.concordia.ca/eas/ as *Programming Assignment **2.*** **Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed.**

3) Finally, notice that a demo is required for the assignment. Failure to do your demo, or missing you reserved demo time, will result in a zero mark for the assignment regardeless of your submission. *There will be no substitution for a missed demo time.* Please see course outline for further details.

Evaluation Criteria

Part 1 (7 points)	
UML representation of class hierarchy	2 pt
Proper use of packages	1 pt
Correct implementation of the classes	1 pt
Constructors	1 pt
toString() & equals()	1 pt
Driver program & general correctness of code	1 pt
Part 2 (3 points)	
Access rights	1 pt
copyCityBuss() and its behaviour	2 pt
Total	**10 pts**