Concordia University
Comp 249 – Fall 2020
Object-Oriented Programming II - Assignment 2
Due 11:59 PM - Friday October 23, 2020

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. You should however handle each part as if it was a separate assignment (i.e. you should not modify Part I after finishing with it; rather create a copy of that part and use it to create Part II. This is needed so that you can finally submit both parts!

Part I

Various flying objects can be described as follows:

- An **Airplane** class, which has the following attriobutes: a *brand* (String type), *price* (double type) and *horse power* (int type).

**

- A **Helicopter** is an **Airplane** that additionally has the following: a *number of cylinders* (int type), a *creation year* (int type), and a *passenger capacity* (int type).

- A **Quadcopter**, **i**s a **Helicopter** that additionally has the following: *max flying speed* (int type), which indicates its maximum moving speed.

- A **Multirotor i**s a **Helicopter** that additionally has the following: *number of rotors* (int type), which indicates its number of rotors/blades that it has.

- A **UAV (Unmanned aerial vehicle / Drone)** class has the following attributes: *weight* (double type), and *price* (double type).

** All images have been obtained from, and © wikipedia.org

- An **AgriculturalDrone** (which is used for crop production) is **UAV** that additionally has the following: *brand* (Sting type), and *carry capacity* (int type).

- A **MAV (Micro Air Vehicle)**, is a miniature **UAV** that has a size restriction (and can be as small as few centimeters. It has the following: *model* (String type) and *size* (double type).

Phase I:

1. Draw a UML representation for the hierarchy of the above-mentioned classes. 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 5 different Java packages for the classes. The first package will include the **Airplane** class. The second package will include the **Helicopter**, and the **Quadcopter** classes. The third package will include the **Multirotor** class. The fourth package will include **UAV** class, and the last package will include the **AgriculturalDrone** and **MAV** classes.

   - 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 **Quadcopter** class must accept 7 parameters to initialize the *brand*, the *price*, the *horse power*, the *number of cylinders*, the *creation year*, the *passenger capacity*, and the *maximum flying 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 return clear description and information of the object (i.e *"This Agricultural Drone is manufactured by Agridrones. It weights 340 pounds, and costs 98000$. It can carry up to 25 Kg….*).

     - The ***equals()*** method must first verify 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 is conducted to see if the two objects are actually equal. Two objects are equal if the values of all their attributes are equal.

   - For all classes you **must** use the appropriate access rights, which allow most ease of use/access **without compromising security**. Do not use most restrictive rights unless they make sense!

- 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 calling 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 packages. In the main() method you must:
    - Create various objects from the 7 classes, and display all their information (you must take advantage of the ***toString()*** method).
    - Test the equality of some of the created objects using the ***equals()*** method.
    - Create an array of 15 to 20 these flying objects (**HINT**: Do you need to add something else to the classes described above? If so; go ahead with that!) and fill that array with various objects from these classes (each class must have at least one entry in that array).
    - Trace(search) that array to find the two objects that have the least expensive price (that is least expensive and second least expensive). Display all information of these two objects along with their location (index) in the array.

Phase II

In that phase, you need to modify/expand the implementation from Phase I as follows:

1. In the driver program, you need to add another static method (add it above the main() method), called ***copyFlyingObjects***. The method will take as input an array of these objects (the array can be of any size) and returns a copy of that array. That is to say, the method needs to create an array of the same length as the passed array, copies all objects from the passed array to a new array, then returns the new array. Your copy of the objects **will automatically** depend on the copy constructors of the different listed classes. You **must** consider ther following restrictoins: **Do NOT attempt to explicitly find the exact type of the objects being copied, do NOT attempt to find the object type inside the copy constructors and Do NOT use clone().**

2. In the driver program, create an array of 15 to 20 objects (must have at least one from each of the classes), then call the *copyFlyingObjects()* method to create a copy of the that array.

3. Display the contents of both arrays, then add some comments indicating **whether or not the copying is correct. If not; you need to explain why it has not been successful or as you might have expected.**

General Guidelines When Writing Programs

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

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

- Use **JavaDoc** to create the documentations of your program. Use appropriate comments when 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.

Submitting Assignment 2

- For this assignment, you are allowed to work individually, or in a group of a maximum of 2 students (i.e. you and one other student). You and your teammate must however be in the same section of the course. Groups of more than 2 students = zero mark for all members!

- Only electronic submissions will be accepted. Zip together the source codes.

- Students will have to submit their assignments (one copy per group) using the EAS system (https://fis.encs.concordia.ca/eas/). Assignments must be submitted in the right folder of the assignments. **Assignments uploaded to an incorrect folder will not be marked and result in a zero mark. No resubmissions will be allowed.**

- Naming convention for zip file: Create one zip file, containing all source files and produced documentations for your assignment using the following naming convention:

  The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number respectively. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a1_Mike-Simon_123456.zip.* if working in a group, the name should look like: *a1_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*

- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded and all others will be disregarded.

- If working in a team, only one of the members can upload the assignment. Do **NOT** upload the file for each of the members!

⇨ Important: Following your submission, a demo is required (please refer to the courser outline for full details). The marker will inform you about the demo times. Please notice that failing to demo your assignment will result in zero mark regardless of your submission.

Evaluation Criteria

Phase I	
UML representation of class hierarchy	2 pts
Proper use of packages	1 pt
Correct implementation of the classes	1 pt
Constructors	1 pt
toString() & equals()	1 pt
Phase II	
copyFlyingObjects() and its behaviour	2 pts
Driver program & general correctness of code	2 pts
Total	**10 pts**