COSC1337 Programming Fundamentals II /ITSE2357 Advanced Object-Oriented Programming

The submission opens two days before the due day, and closes right at the due time on due day. One submission is allowed per project. It is students' responsibilities to submit timely and correctly in order to get credits. Students MUST start each project when it is first discussed in class in order to complete it on time. All projects are individual projects. Project due days may be changed to better align with lectures.

If one of the following is true, not credits will be given:

- The project is late.
- The algorithm or class design is missing.
- The project has syntax errors.
- There are no comments (Javadoc format required) at all in your code.
- Wrong files are submitted.
- **A project is a copy and modification of another student's project. (Both will receive 0.)**

Files to be submitted through Canvas:

- The UML design (UML class diagram)
- All source codes and driver programs
- All supporting files if any

Software Development Project Rubric: (Design is worth 20%; the rest is worth 80 %.)
Analysis: There will be no credit if the software doesn't meet customer specification at all.

- Does the software meet the exact customer specification?
- Does the software read the exact input data and display the exact output data as they are shown in sample runs?
- Does each class include all corresponding data and functionalities?

Design: There will be no credit if the design is missing.

- Is the design (a UML class diagram) an efficient solution?
- Is the design created correctly?

Code: There will be no credit if there are syntactic errors.

- Are there errors in the software?
- Are code conventions and name conventions followed?
- Does the software use the minimum computer resource (computer memory and processing time)?
- Is the software reusable?

Debug:

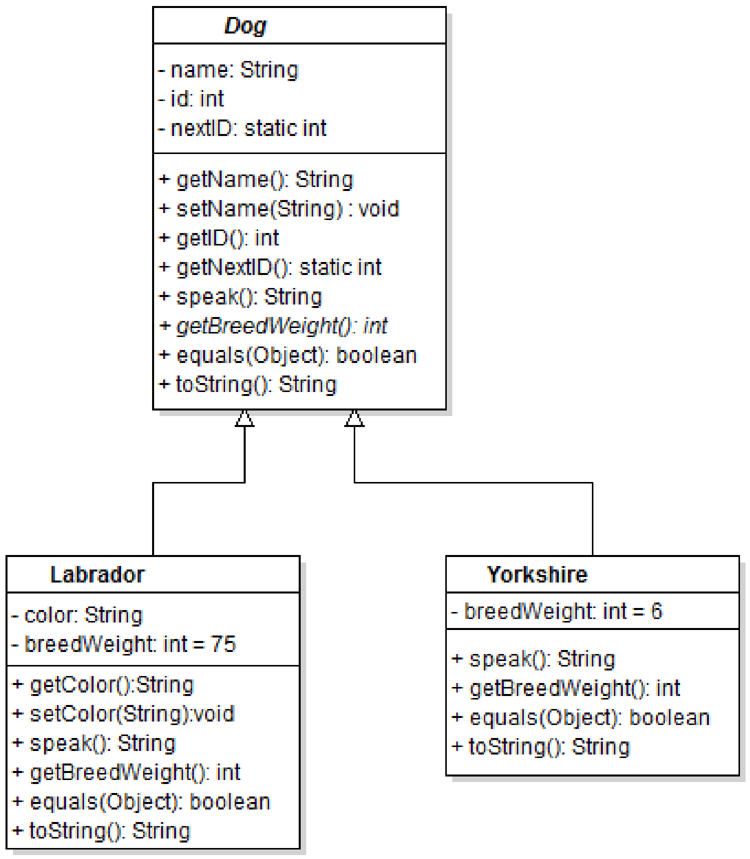
- Are there bugs in the software?

Documentation: There will be no credit if comments are not included.

- Are there enough comments included in the software?
- Class comments must be included before a class header.
- Method comments must be included before a method header.
- More comments must be included inside each method.
- All comments must be written in Javadoc format.

Project 3 Inheritance
A dog

Write classes to model two types of dogs: *Labrador* and *Yorkshire*. The UML class diagram shows a design for *Dog* class and its sub classes.



Class Dog is an abstract class with an abstract method *getBreedWeight*(). It overrides *equals* and *toString* method of Object class.

Class *Labrador.java* and class *Yorkshire.java* extend *Dog.java* class and define *getBreedWeight*() that is abstract in class Dog. They also override *equals, toString, and speak method of* class *Dog*. All Labradors are born with an average weight of 75. All Yorkshires are born with an average weight of 6. The weight is not changeable. A unique id is assigned to each new dog.

In constructors of sub class, always set up parent part by using super constructor, and then add child part.

In other overridden methods of sub classes, always set up parent part using methods of super version, and then add child part.

Dog, Labrador, and Yorkshire must have a default constructor and other constructors. Constructors in abstract class *Dog* are created for its sub classes to reuse ONLY. A dog can be created as a Labrador or a Yorkshire. A Labrador or a Yorkshire can be referred by a variable of same type or a variable of its super type, *Dog* type.

```
Labrador labrador1 = new Labrador ();
Labrador labrador2 = new Labrador (“Buddy”);
```

```
            Labrador labrador3 = new Labrador (“Buddy”, “yellow”);
            Dog labrador4 = new Labrador (“Candy”);

            Yorkshire yorkshire1 = new Yorkshire ();
            Yorkshire yorkshire2 = new Yorkshire (“Buddy”);
            Dog yorkshire3= new Yorkshire (“Sugar”);
```

Write *DogTest.java*, a simple driver program. Create an array list of Labradors and Yorkshires.