Software Development (CS2514) Assignment 2

Inheritance (Due: February 17th. Marks: 5)

General Comments

Carefully read the submission guidelines before you submit the assignment.

Like all other exercises, this is an exercise about implementing *maintainable* classes. You should always assume the specifications may change (slightly) and make sure your implementation can implement these changes with the minimum amount of effort.

Before you start implementing your classes, please make sure you understand the API. If you don't you'll make your life much more difficult.

Learning Objectives

For this assignment you will learn about inheritance. You will learn how to design a class hierarchy for `Bikes`. You will start by studying the dependencies between the different `Bike` classes, by working out which are the more general `Bike` types, and by determining which `Bike` types are more specific than other `Bike` types. As part of this exercise you may have to introduce intermediate `Bike` types that aren't mentioned in the specification and other auxiliary classes to represent `Bike` components. You will then translate your class hierarchy to a `Java` inheritance-based hierarchy, with the more general classes at the top of the hierarchy and the more specific classes at the botom of the hierarchy.

Your aim should be to maximise code re-use in the hierarchy, so *don't use copy-and-paste* to "implement" shared behaviour.

Main Details

In this assignment you will implement a `Bike` class hierarchy that has the class `Bike` sitting at the top of the hierarchy.

There are three kinds of `Bikes`: `MountainBike`, `CityBike`, and `Hybrid`. All `Bikes` have `Brakes`, `Wheels`, a `Saddle`, and a `Handlebar`. *For the moment* we shall assume these are the only components they have in common. You are supposed to represent these components in a robust way. For example, when you compile your classes the `javac` compiler should inform you when you are trying to use a non-existent component.[1] *Hint: introduce auxiliary classes.*

All your classes should respect encapsulation, so all class and instance attributes should be `private`. Using `private` attributes is a major part of the challenge. If you use attributes that are not `private` you can lose up to 50% of the marks.

The following are the details of the components of the remaining `Bikes`.

[1] Errors like this always happen, so you should exploit the fact that `Java` is strongly typed and that you can only use existing classes.

MountainBike: `LowFrame`.

CityBike: `FrontLight` and `RearLight`, `Carrier`, and `HighFrame`.

Hybrid: `FrontLight`, `RearLight`, and `MediumFrame`.

Each `Bike` can print its components with a method called `void printComponents( )`. You should implement the method yourself and the components should be printed *properly,* i.e. they should print as proper English. For exmple, a `MediumFrame` instance should not print `MediumFrame` but it should print `medium frame` or equivalent.

To make this assignment a bit more challenging and to make sure you understand the advantages of reusing superclass behaviour, you are *not* allowed to represent `Bike` components with arrays, string builders, with `ArrayLists` or with other collection-based classes.

Every component should be an instance of a dedicated `Component` subclass. The `FrontLight` and `RearLight` can be on or off.

Submission Details

- Remember that each class should be in its own file. You will lose marks if you violate this rule.
- Remember that all classes, attributes, and public methods should be commented using a proper `JavaDoc` comment.
- The `JavaDoc` class comment should explain the purpose of the class; *not the purpose of the assignment.*
- Please provide your name and student ID as part of your class `JavaDoc`. You can use the `@author` tag for this:

```Java
@author Java Joe (ID 12345678)
```

- Use the CS2514 `Canvas` section to upload your classes as a single *.tgz* archive called *Lab-2.tgz* before 23.55pm, February 17th, 2020. To create the *.tgz* archive, do the following:
  - Create a directory `Lab-2` in your working directory.
  - Copy `Bike.java` and your other user-defined Java files into the directory. Do not copy any other files into the directory.
  - Run the command '`tar cvfz Lab-2.tgz Lab-2`' from your working directory. The option '`v`' makes `tar` very chatty: it should tell you exactly what is going into the `.tgz` archive. Make sure you check the `tar` output before submitting your archive.
  - Note that file names in `Unix` are case sensitive and should not contain spaces.
- Note that the format of your submission should be `.tgz`: do *not* submit `zip` files, do *not* submit `tar` files, do *not* submit `bzip` files, and do *not* submit `rar` files. If you do, it may not be possible to unzip your assignment.
- No marks shall be awarded for classes that do not compile.