Lab Assignment 6, TCSS 143A Winter 2019

OBJECTIVE

The objective of this assignment is to give you practice with inheritance and *equals* method. The assignment consists of 4 exercise sets.

ASSIGNMENT SUBMISSION

Find the lab partner assigned to you for today's lab and start working on the exercise sets in the order they are listed. Once you are done with a set, check with one of the other pairs sitting next to you regarding their progress – help each other. Then, as a pair, present your solutions to the lab instructor. Each student is to have his version of the programs/answers and be capable of presenting them for the pair. The presenter will be chosen at random by the lab instructor. All the exercises other than the last set need to be shown to the lab instructor before leaving the lab for full credit. If your pair does not manage to finish the last set during the lab time, the last exercise set may be finished at home and submitted to the lab instructor per his instructions. Use jGrasp unless indicated otherwise.

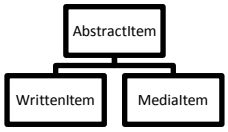
1. Inheritance implementation basics (25%)

Solve the following Practice-It problems:
Building Java Programs, 4th edition, Chapter 9:

- Self-Check 9.8: CarTruck
- Self-Check 9.9: CarTruck2
- Exercise 9.4: MonsterTruck

2. Equals and inheritance (25%)

A library contains two types of items: written materials and media items. Since much of the code for these two classes is identical (files provided in `Canvas`), it is your job to create a class hierarchy that reduces redundancy and abstracts all common functionality from these two classes into an `abstract` class called `AbstractItem`**.** This means that after restructuring the code, `AbstractItem` should be a parent of both written and media items and should contain the properties and methods that are identical in both of these subclasses (a simplified diagram is shown below). What follows is that subclasses should no longer contain identical properties or methods themselves. Make sure you provide a driver for your new hierarchy. Add an `equals` method to `AbstractItem` that compares items within the library for equality by their catalog ids and test if it is possible to compare a media item with a written item using your method.



3. Inheritance implementation, and different equals (30%)

Download the class *AbstractDateType.java* and its UML class diagram (you can open it in www.draw.io). Then, implement in Java a class hierarchy for the following:

- a child class of *AbstractDateType* called *BasicDateType* – this class adds one property to the *AbstractDateType* – the year – and its associated getter; the class should also include a parameterless constructor, *toString* and a new *setDate* method that allows the end-programmer to assign day, month, and year to the object
- a child class of *BasicDateType* called *EnhancedDateType* – this class adds one property to the *BasicDateType* – the name of the month (e.g. January) – and its associated getter; it should also include a parameterless constructor, *toString* and a new *setDate* method that allows the end-programmer to assign day, month, year, and the name of the month to the object
- a child class of *EnhancedDateType* called *DateTimeType* – this class adds 2 properties to the *EnhancedDateType* – integers that indicate hours and minutes – and their associated getters; it should also include a parameterless constructor, *toString* and a new *setDate* method that allows the end programmer to day, month, year, the name of the month, hours, and minutes to the object

- Add an *equals* method to each class in this hierarchy so that objects created on this template can only be equal if their contents are same and they are of the same exact type, e.g. an object of type *BasicDateType* can never be equal to an object of any other type, including its children instances.

4. Inheritance concepts and UML (20%)

a. Create a UML diagram for the code you created in step 3

b. Once you are done with the UML and the code from step 3, for each method that can be called on a *DateTimeType* object, (whether inherited or direct), fill out the following table (your table may have more rows than the one here). Do not include constructors, i.e. assume *DateTimeType* object called *myDate* was already created, as in *DateTimeType myDate = new DateTimeType();:*

Method name	In which class is it defined?	Is this method an example of override / overload / neither

----- THE END -----