CS 203 HW #8 — Enough Rope to Hang Yourself

Due Dates: **Part A –** Wednesday, April 9th at 10pm
Part B – Wednesday, April 16th at 10pm

This assignment will give you practice using inheritance in Java and also practice in software design.

Grading

Your assignment grade for both parts of the assignment will be based on:

- Program functionality (90%)
- Code quality (10%) - this will be evaluated using the same metrics as previous assignments.

Overall Specification

For this assignment, you will create a Java program of your choice that takes advantage of inheritance. This assignment is subject to the following limitations and requirements:

1. Your program must contain at least four classes you have written. At least three of these classes will be used for creating objects, and one class will have a main method for running your program.

2. Each of your classes must have a clear reason for existing, that is, a class should model one "thing." Its attributes should describe or capture some aspect of the object being modeled, and the methods should perform actions that pertain to the object and make use of the object's instance variables in some way.

3. Unless it makes no sense, your classes should have `equals` and `toString` methods that behave properly, i.e., `equals` correctly reports when two objects of that class are equal, and `toString` produces a String representation of the object that is more helpful than the default `toString` method.

4. Two classes you have written must inherit from some other class. For example, in the Balls demo the `Balloon` class inherits from `Ball` and the `Chameleon` class inherits from `Balloon`. Another example: In the Library example, the `Book` and `Periodical` classes both inherit from `CirculationItem`.

5. The classes you define that inherit from others must *clearly benefit* from that inheritance. Your use of inheritance should not be contrived to meet the requirements of the assignment. Meeting this requirement effectively constitutes a significant portion of your grade for part A of this assignment.

6. In at least one place, a class that inherits from a parent class should override one of its parent's methods. The constructor, `equals`, and `toString` methods don't count for the purposes of this requirement.

7. Your program should have an obvious purpose or point. It should not be just a random collection of classes. Each class you write must contribute significantly to the program's overall functionality.

8. If you wish, you may build upon code you've written from previous assignments. You can even use code written by others. However, code written by others must be clearly identified. If you start with existing code, you still must write additional code specifically for this assignment that meets the requirements above. For example, adding one more face to the Faces lab for CS273 would not be sufficient.

9. You are allowed to work in a partnership with one other student on this assignment. However, the scope of your program must be larger. Ask permission before starting work with a partner. A partnership will not be permitted if you ask less than 72 hours before part A of the assignment is due.

Part A: Design Your Program

Overview (5%)
Create a clear and concise description in English of what you intend to create that is about one or two paragraphs in length. Your description should include (but not be limited to) what classes you propose to implement and what your class with a `main()` method in it will do with those classes.

UML Diagram (70%)
Draw a UML class diagram for the classes you will implement. Each class specification should show the attributes (instance variables) and actions (methods) that each class will have, including their public/private/protected status. The diagram should also show the "is-a" and "has-a" associations between the classes. You can draw this diagram by hand or use a tool[1] for the purpose.

Create Starter Code (15%)
Create a "starter code" version of your program. The starter code should contain a partial implementation of each class:

- Each class should contain all of its attributes.

1 Dia is the recommended tool for this. If you complete your specification in Dia, it can generate starter code for you though you'll have to edit it to meet the homework specification. Dia is included in the engineering build that is installed on the computers in the Shiley computer labs and classrooms. You can also download a copy for your personal computer: http://dia- installer.de

- Methods should be present as stubs: they must be declared but need not contain any code except a return statement. The return statement is optional for void methods.
- Each class and method should have appropriate comment headers using the Javadoc comment conventions. You can check this by loading your starter code into BlueJ and selecting Tools->Project Documentation to view your API.
- The code should compile.

Hints for Part A

- **Important:** Be careful not to get carried away! Design a program that you know you can build. It's okay to have big ideas, but make sure you have a plan to create something of small scope first that meets the specifications. Then you can expand it as time allows.
- If you are uncertain about what meets the requirements, *ask for clarification*. Don't assume.
- Consider using inheritance to enhance classes that already exist. As a simple example, did you ever wish that the String class had an `indexOfSecond` method? You can create your own string class that does this!
- Proposed exceptions or adjustments to the specifications will be considered on a case-by-case basis. Ask first! Do not assume a change is okay.
- If you are paralyzed for ideas about what to do, feel free to implement the Library example we've used in class. Alternatively, consider enhancing a previous assignment to add functionality to it.

Logistitics

There is no starter code for this assignment.

Follow these steps to turn in your assignment:

- Make sure your name is in all of your files, including your UML diagram and each .java file.
- Create a folder that contains both your program description and your UML design.
- Add a sub-folder that contains your BlueJ project.
- Compress the entire folder into a .zip
- Submit your zip file via the associated "Turn it in Here" link on the course website.

Part B: Write your Program

Create a complete and working program based on the design you created in Part A. It is okay if your final program does not match your design exactly. In fact, some deviation is normal. However, the code you turn in should clearly be derived from your original design.

Your program should make clear use of the Java's built-in support for inheritance where it make sense rather than working around it. For example:

- If extending the functionality of a parent method, you should use the `super` reserved word.
- If you have a class with multiple constructors that have content in common, use the `this` reserved word to call a fundamental constructor.
- If you need to identify the type of an object, use the `instanceof` operator rather than an id variable.

Logistics

- Make sure that exactly one class in your program contains a properly defined main method: `public static void main(String[] args)`. This is the method that the grader will use to run your program.
- In the class that contains the main method, add the following information to the comment header at the top of the file:
    - A short description of what your program does
    - Major changes, if any, you made to your design since part A.
    - Known errors or shortcomings in your program.
    - Any information the grader will need to know to run and use your program
    - Where in the code that the grader can see that you are overriding one or more methods
- Compress your entire BlueJ project into a single .zip archive. Be sure your archive includes the directory itself so that, when unzipped, it creates that directory.
    - On Windows operating systems, you can do this by right-clicking on the directory in Windows Explorer and selecting “Send to” and “Compressed (zipped) Folder.” This will create a .zip file in the project’s parent directory that you can submit.
    - On OS/X you should right click on the folder in Finder and select “Compress” or “Create Archive”. This will create a .zip file in the project’s parent directory that you can submit.
- Submit your .zip file via the associated “Turn it in Here” link on the course website.