CS 46A Homework 3

Overview

In this assignment, you'll demonstrate your ability to write classes that have object constructors and callable methods.

Learning Outcomes

By the end of this assignment, you should be able to …

- … implement a class with several methods.
- … use a tester program to test a class that your write.
- … generalize a class using arguments and static constants.

Guidelines

1. Use BlueJ to create your code.
2. You must name your classes exactly as specified. Otherwise Codecheck will not be able to process your submission and you will get no credit.
3. When you are finished with your code, submit it to Codecheck one final time then download the `.signed.zip` file.
4. You must upload all three `signed.zip` files together to Canvas and you should double check the files in Canvas to make sure all three zip files are uploaded.
5. Do not open the downloaded zip files. The files are digitally signed, and the grader program will check that they have not been opened.

Problem 3A

Codecheck Link: HERE

Goal:

In this problem, envision that you need to write a class called **`Climber`** that describes a child is climbing a pole. The child can either move up (climb) one unit at a time or move down (slide) all the way to the bottom. Perhaps this class is for is a mini-game in a role playing-type video game (an "RPG"). Your class should have one constructor and 5 methods. A **`ClimberTester`** class is provided for you to verify your **`Climber`** class works as expected.

Instructions:

Start a new BlueJ project called hw3a in the cs46a/homework/hw03 folder. In the BlueJ project, create a class called **`ClimberTester`** and copy the code from Codecheck. Do not change this class in any way.

Next, create another class called **`Climber`** and copy over the contents from CodeCheck. Fill in this class with the following:

1. A constructor:
   `public Climber (String theName, int thePosition)`
   Constructs a new climber with the specified name and position. Remember that the job of the constructor is to initialize the instance variables.
2. Five methods:
   `public String getName()`
   Gets the name of the Climber object
   `public int getPosition()`
   Gets the position of this Climber object on the pole
   `public void setName(String newName)`
   Set a new name for this Climber object
   `public void climb()`
   Climbs one unit on the pole (add one to this climber position to simulate climbing one position on the pole_
   `public void slide()`
   Slide this Climber object back to the bottom of the pole

Other tips and guidelines:

- Javadoc is provided for this class. Study how it is done so you can write the required Javadoc in the next problem.
- The position of the climber can be anywhere from the bottom to the top of the pole.
- The pole is 10 units high.

Problem 3B

Codecheck Link: HERE

Goal:

In this problem, you will write a class to mimic the behavior of HAL 9000 – a fictional "character" in Arther C. Clarke's 2001: A Space Odyssey. HAL 9000 (Heuristically programmed ALgorithmic computer) is an AI (artificial intelligence) computer that controls the system of the spacecraft and interacts with the ship's crew. Near the end of the film, HAL 9000 stops taking commands from the humans.

In the **`Hal9000`** class you design, HAL 9000 will respond with cordial greetings to the humans but will not complete the commands they request. A **`Hal9000Tester`** class is provided for you to verify your **`Hal9000`** class works as expected.

Instructions:

Start a new BlueJ project called hw3b in the cs46a/homework/hw03 folder. In the BlueJ project, create a class called **`Hal9000Tester`** and copy the code from Codecheck. Do not change this class in any way.

Next, create another class called **`Hal9000`** (there is no starter code provided for this example). Fill in this class with the following:

1. A constructor:
   `public Hal9000 (String crewName)`
   Constructs a new object with the specified name of the crew member. HAL will communicate with this crew member.
2. Five methods:
   `public String getName()`
   Gets the name of the crew member.
   `public String greet()`
   Returns a string "Greetings, [crew member].". For example, if the crew member's name is "Dave", then return "Greetings, Dave."
   `public void setName(String newName)`
   Set a new name for the crew member.
   `public String giveStatus()`
   Returns a string "Everything is a go, [crew member].". For example, if the crew member's name is "Dave", then return "Everything is a go, Dave."
   `public String executeCommand(String whatToDo)`
   Returns a string "I am sorry, [crew member]. I can't [task].". For example, if the crew member's name is "Dave" and the task is "engage drive", then return "I am sorry, Dave. I can't engage drive."

Other tips and guidelines:

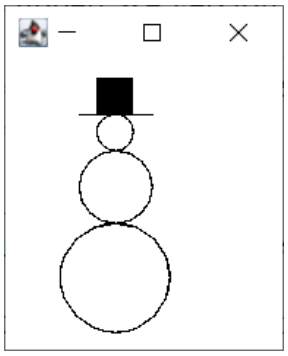
- In this code, half of the points will be assigned for correct Javadocs

Problem 3C

Codecheck Link: HERE

Goal:

Revisit the **SnowmanViewer** code from Homework 2 to write a more generalized **Snowman** class. While the class your write here will be more generalized, the output of your class will be identical to problem 2C:



You will make two main updates to your code from Homework 2:

1. Draw the location of the snowman relative to a given position (x, y) rather than relative to the origin (0,0)
2. Remove all "magic numbers" from the code

Instructions:

In Homework 2 Problem 2C, you wrote a **SnowmanViewer** with a main() method and did all the drawing in it. But there is a problem with that design: If you wanted to draw a second snowman (or a third or a fourth), you would have to recalculate all the coordinates and repeat all the lines of code. Not good! The solution to this problem is to create a **SnowMan** class that draws a snowman at a given location. Then that class can be used in different applications to draw **SnowMan** objects at any locations.

Create a BlueJ project and import the graphics package. Then create a new class **SnowMan**. There is no starter code for the class (but you may want to take a look at your solution for Homework 2 Problem 2C for inspiration).

Start a new BlueJ project called hw3c in the cs46a/homework/hw03 folder. Following the same steps as in Homework 1 Problem 1C, import the Horstmann graphics package. In the BlueJ project, create a class called **SnowManProg** and copy the code from Codecheck. Do not change this class in any way – this is the tester script.

Next, create a new **c**lass **Snowman**. Fill in this class with the following:

1. A constructor:
   ```
   public Snowman (int x, int y)
   ```
   Constructs a Snowman object at location (x, y) – the upper left corner of the hat.
2. A method:
   ```
   public void draw()
   ```
   Draw the Snowman object.

Next, follow the steps below to draw the picture of the snowman:

1. Draw a rectangle of size 20 x 20 as the hat at position (x, y).
2. Fill the hat with the default black color.
3. Draw a the brim of the hat and the three circles relative to the hat as you did in Homework 2 Problem 2C. You'll have to figure out the coordinates for each object – they should have the same size as in the previous problem, but they should be drawn relative to the point (x,y) instead of (0, 0)
4. Draw a circle of diameter 40 at position (40, 50).
5. Draw a circle of diameter 60. You need to figure out the position of the final circle.

Tips and Guidelines:

- There should be no "magic numbers" in your code (2 is not a magic number in this example). Use the following values in your class and use them in your references:

```
public static final int HAT_SIZE = 20;
public static final int HAT_BRIM_SIDE_SIZE = 10;
public static final int SMALL_DIAMETER = 20;
public static final int DIAMETER_INCREMENT = 20;
public static final int RADIUS_INCREMENT = 10;
```

- Do not draw anything in the Snowman constructor.
- Be sure to add Javadoc comments if you did not do so in Homework 2 Problem 2C
- Hint: start with your code from Homework 2C. First alter the code to be referenced to (x,y) with some magic numbers. Then, slowly update all of the magic numbers to the static numbers above.