Due Saturday January 16th, 2016 at 11:55PM

This assignment is primarily designed to be a review of the following object-oriented concepts: interfaces, abstract classes, and subclasses. Indirect goals include practicing standard code organization, getting a feel for lab sessions for this course, and simply warming up programming skills!

Overview

Your goal is to create an **abstract class** implementing the minimal interface given below, and then create **at least two subclasses** extending your abstract class. You have a lot of freedom over what your classes represent, so as long as you adhere to all the specific requirements in the 'Tasks' segment of this assignment, you can be as creative as you like.

Once you've got your abstract class and 2+ subclasses set up, you will write a driver to instantiate your subclasses a number of times and store them in an array.

Read the following section carefully to ensure you address every requirement. See rubric for details.

Tasks

- Create a new project/package in the IDE of your choice and name it **program0.0**

- Copy the 'Thing' interface given below into a new file in your project. Interfaces are a common way to describe the actions or functionality for certain objects without worrying about implementation details; this is called *abstraction.*

- Create a new abstract class implementing the 'Thing' interface. This class must represent a general object of your choice. 'Vehicle,' 'Food,' 'SportsTeam,' and 'Shape' are all good examples, because each represents a class of 'Things' with variations on the same properties and interactions. For example, Bike, Car, and Truck are all Vehicles, and each could have a different value for a 'wheels' field and could all implement a 'drive' method differently.

Your abstract class must contain the following:

- A **name field**
- At least **2 other fields** for properties or attributes shared by all instances
- Getter/Setter methods for each of the above fields
- **1 static class variable 'count'** representing the number of instances
- At least **1 abstract method** for a type of interaction shared by all subclasses
    - ex: SportsTeam might have an abstract 'playSport' method, which might say "kicking ball" for a SoccerTeam and "swinging bat" for a BaseballTeam
- **Implement the 'describe' method** from the 'Thing' interface to print a brief description to the console
- **Implement the 'setName' method** from the 'Thing' interface to set the name field to the given string

- Create at least 2 subclasses extending your abstract class. Each class must represent a more specific type of your chosen object.

Your subclasses must each contain the following:

- At least **1 field or method** unique to the particular subclass
- **1 static class variable 'count'** representing the number of instances
- Getter/Setter methods for each added field
- **A constructor method** which at minimum increments the count variables of the superclass and subclass. You can also instantiate fields here if you like.
- **Implement the abstract method** from the superclass. The easiest thing is to simply print something to the console.
- **Implement the 'describe' method** from the 'Thing' interface to print a brief description to the console. This method should begin by calling the 'describe' method of the superclass, and could then print a sentence involving some or all of the class' fields.

- Create a new class called **Driver** with a main method.

In your driver's main method, declare an array to hold instances of the classes you created, and then populate the array with objects from both subclasses. The array must contain **at least three of each type** of subclass. Each object you add to the array must have values given for all its fields, either in the constructor in the class itself or through the setter methods.

Finally, use a loop to iterate over your array of objects, call the 'describe' method for each object, and call the abstract method you created if it also prints a brief message to the console.

'Thing' Interface

```
/*
 * Interface 'Thing' describes a very general template for interactions a 'Thing' should support.
 *      Methods described are 'describe' and 'setName'
 */
public interface Thing {

        /*
         * This method should print a brief description of the thing.
         */
        void describe();

        /*
         * This method assigns the given name to the appropriate field within thing's subclass.
         */
        void setName(String name);

}
```

Scoring Rubric

For each item in the following lists, points are awarded as follows unless otherwise specified:

1 point for the item existing (the field or method is in the class, etc)
1 point for the item being used (the field is given a value or the method is called at some point during program execution)
1 point for appropriate comments/documentation

Abstract class - 21 points available
- A name field
- New field 1
- New field 2
- 1 static class variable 'count'
- At least 1 abstract method
- Implement the 'describe' method
- Implement the 'setName' method

Subclasses – 15 points available for each subclass (up to 30 points)
- 1 unique field or method
- 1 static class variable 'count'
- A constructor method
- Implement the abstract method
- Implement the 'describe' method

Driver – 15 points available
- Array of objects
- Objects instantiated and placed into array – *1 point for each object (up to 6)*
- Loop over array, printing descriptions – *1 point for each object (up to 6)*

Total points available: 66

The final score will be calculated out of 60, so it is possible to miss a few points without penalty.