CSI 213 – Data Structures

Lab 2 Assignment

Posted on: 02/06/2017

A SIMPLE OBJECT ORIENTED PROGRAMMING EXERCISE

Program Description

In this lab you will create a program for the lab assignment 1, which shows the different attributes of flowers. Let's remember the design of the last lab assignment:

A *Flower* class that has the following members:

- A field for the name of the flower (a string)
- A constructor and appropriate accessors and mutators.
- A method(*isEdible*) that prints a string describing a Flower can or cannot be edible (a string)

A *Cauliflower* class that extends the *Flower* class, and it has the following members:

- A field for the color of the plant (a string)
- A constructor and appropriate accessors and mutators.
- A method(*isEdible*) that prints a string describing Cauliflower can or cannot be edible (a string)

An *Iris* class that extends the *Flower* class, and it has the following members:

- A field for the color of the plant (a string)
- A constructor and appropriate accessors and mutators.
- A method(*isEdible*) that prints a string describing Iris can or cannot be edible (a string)

Some important things to remember:

1. Every kind of flower we're working with has a different color and edibility.
2. All Cauliflowers are Flowers, but not all Flowers are Cauliflowers.
3. All Irises are Flowers, but not all Flowers are Irises.

Part 1: The Flower Class

Create a file called Flower.java. Inside the file, create a public class called Flower.

What kind of information is relevant to a Flower? All of the flowers have a name to distinguish one from another, so start by defining that attribute.

```
private String flowerName;
```

In most cases, all of the fields of a class should be *private*, because you don't want any external classes being able to modify that class' data directly (since it could cause unwanted behavior). This is called encapsulation, which is the idea that a classes internal implementation should be hidden from other classes, and that a class should only be interacted with by calling public methods on it.

To be able to find out the name of the flower, we need a public accessor method:

```
public String getName(){
    return flowerName;
}
```

This way, anyone can find out the name of the Flower (by simply calling the method), but no external classes can modify it.

To be able to change the name of the flower, we need a public mutator method:

```
public void setName(String name){
    flowerName = name;
}
```

Finally, our class needs a *constructor*. When we create instances of classes using the *new* keyword, the constructor is a special method that is called to initialize the fields. Our constructor should take in the name of the flower as a parameter. Constructors don't have return types:

```
public Flower(String name){
    flowerName = name;
}
```

Finally, define a default constructor that takes in no values:

```
public Flower(){

}
```

Before you move on, make sure you add **documentation** to the class via comments. Every class should have comments that describe the purpose of its fields, and the details of each method. For fields, you can use inline comments. So add a comment above your *flowerName* field as follows:

```
//the name of the Flower
private String flowerName;
```

For methods, use javadoc comments where you describe the method's behavior, and then add details for each parameter, as well as the return value. Parameter descriptions are prefixed with @param, and return descriptions are prefixed with @return. For this method, there are no parameters, so we only describe the return value:

```
/**
 * Accessor method to get the name of the Flower
 * @return the name of the Flower
 */
public String getName(){
    return flowerName;
}
```

Finally, we can write a similar javadoc comment for the constructor (with details about its parameter):

```
/**
 * Constructor to initialize the flowerName
 * @param name
 */
public Flower(String name){
    flowerName = name;
}
```

If you follow this convention, the Javadoc tool can generate an HTML file that outlines your program. While this may seem unnecessary for such a simple program, it can be extremely helpful for complex ones. So it's a good habit to get into.

If you're not familiar with Javadoc and want to learn more about it, you can read about it online after lab. For quick reference, wikipedia has some good information:

http://en.wikipedia.org/wiki/Javadoc. In this lab, your javadoc comments will be very simple and straightforward.

AT THIS POINT, YOUR PROGRAM SHOULD COMPILE.

Part 2: The Cauliflower Class

Don't start this part until your Flower.java file compiles successfully.

Create a file called Cauliflower.java and inside it, create a class called Cauliflower.

Since all Cauliflowers are Flowers, we want the Cauliflower class to *extend* the Flower class. This means that Cauliflower will be the *subclass* and Flower will be the *superclass*. The class definition for a subclass should appear as follows:

```
public class Cauliflower extends Flower{

}
```

By making Cauliflower *extend* Flower, it automatically *inherits* the flowerName field, as well as the getName() method.

Now, finish creating the Cauliflower class: Add private field "color" to the class. This should be a String. Create public accessor method for the field.

Also create mutator method ("set" methods) called *setColor()* so that this value can be changed. Here we are assuming that a Cauliflower can change the color once it's been created. These methods should each take in a String parameter and their return type should be void.

Create a constructor which takes in String for color, and does the following:

1. Call Flower's constructor with the statement: *super("Cauliflower")* to set *the name of the flower to* Cauliflower.
2. Set the *color* field equal to "White".

Create a public method called is*Edible()* which has no parameters and returns the Cauliflower's edibility information as a String.

Add inline comments for the *color* field. Also add Javadoc comments for the constructor, the accessor and mutator methods, and the *isEdible()* method.

MAKE SURE YOUR PROGRAM COMPILES BEFORE CONTINUING

Part 3: Create the Iris Class

Repeat the steps in the Cauliflower class for Iris and keep it in mind the differences between Iris and Cauliflower. (Such as the edibility information and color of the flower)

Compile your program.

BEFORE YOU LEAVE, SHOW YOUR PROGRESS TO YOUR TA TO GET CREDIT

Grading for this lab is based on completion level:

5.0 = Successfully finished the assignment at the lab, on time and helped other students
4.0 = Successfully finished the assignment at the lab, on time.
3.0 = Most of the assignment is finished with couple errors/missing parts
2.0 = Some of the assignment is finished and left the room early.
1.0 = Did not successfully finish, but attended the lab and showed effort.
0.0 = Did not attend the lab section, or did not attempt the lab.