Laboratory 3: Ice Cream!

Download Lab03.zip from ilearn and extract it into your working lab 3 directory

In this lab, you will be writing a **complete class from scratch**, the IceCreamCone class. You will also write a driver to interact with a user and to test all of the IceCreamCone methods.

Lab:

- Cone
- Ice Cream
  - Flavor
  - Topping
- Ice Cream Cone
- Driver

Part I: Cone

An ice cream cone has two main components, the flavor of ice cream and the type of cone. Write a Cone class that takes an integer in its constructor. Let a 1 represent a sugar cone, let a 2 represent a waffle cone, and let a 3 represent a cup. Make sure to set an invalid entry to a default cone (the choice of default is up to you). Write a method to return the price of the cone. The sugar cone is $0.59, the waffle cone is $0.79, and the cup is free. Write a toString() method to return a String indicating the type of cone that was selected.

Note: If you need a carriage return in a String, use "\r\n"

Part II: Ice Cream

Download the following files:

- Topping.java
  - public double price() //toppings are $0.15 ("no topping" is free)
- Toppings.java //this file has the following public interface:
  - public static Toppings getToppings() //singleton design pattern
  - public int numToppings() //the number of different toppings
  - public String listToppings() //list (and number) available toppings
  - public Topping getTopping(int index) //1-based
- Flavor.java
- Flavors.java //this file has the following public interface:
  - public static Flavors getFlavors() //singleton design pattern
  - public int numFlavors() //the number of different flavors
  - public String listFlavors() //list (and number) available flavors
  - public Flavor getFlavor(int index) //1-based

Ice cream has a number of scoops, a flavor, and a topping. Write an IceCream constructor that accepts the number of scoops (up to 3), a Flavor object, and a Topping object. Set default values (up to you) if the input is invalid (make sure to protect against null). Write a method to return the price (each scoop over 1 adds $0.75 to the cost of the ice cream) and a method to return a String listing the ice cream parameters selected.

Part III: Ice Cream Cone

Download the following file:

- Currency.class //this file has the following public interface:
    - public static String formatCurrency(double val) //formats the double as a String representing US currency

The IceCreamCone is composed of the Cone and the IceCream. Write a constructor that accepts an IceCream and a Cone (protect against null), a method to return the price (the base price is $1.99 plus any costs associated with the components), and a method to return a String display of the ice cream cone ordered by the user. Use Currency.class to format the total price of the Ice Cream Cone. Does this class use proper encapsulation techniques?

Note: Without using the Currency class, it is possible to get final prices that are not quite correct, such as 3.5300000000000002.

Part IV: Ice Cream Driver

Download the following file:

- Keyboard.class //this file has the following public interface:
    - public static Keyboard getKeyboard() //singleton design pattern
    - public int readInt(String prompt)
    - public double readDouble(String prompt)
    - public String readString(String prompt)

Write a driver to obtain the ice cream cone order from the user. Allow multiple ice cream cones to be ordered, and keep a **running total** of the price of the ice cream cones. Of course, wait (loop) for the user to input valid entries. Thus, you are checking for valid entries in the driver and in the constructor of your objects. **This is defensive programming**, although it results in some code duplication.

In addition to main, include the following methods in your driver:

1. public static Flavor getFlavorChoice() //look at the methods available in the Flavors class
2. public static Topping getToppingChoice() //look at the methods available in the Toppings class
3. public static int getScoopsChoice()
4. public static Cone getConeChoice()

All of these methods take input from the user (using Keyb

Example output included as a jpg in Lab04 files.

To compile: javac *.java

To run: java IceCreamDriver

You can also run your driver with my input file via: java IceCreamDriver < ice_cream_test.txt (or by running make.bat)

Only one submission per team is necessary, but please make sure to include both names at the top of your source code, as well as in the comments section when submitting the lab, so both people can get credit.