CS 251 Intermediate Programming
Lab 3: Inheritance

Brooke Chenoweth

Spring 2024

This is a little lab to get to you more familiar with inheritance.

Problem description

Consider a dessert shop that sells candy by the pound, cookies by the dozen, ice cream, and sundaes (ice cream with a topping). We would like to be able to calculate the price of each of these items.

Provided Classes

I am providing you with two classes, `Dessert` and `DessertTest`. Do not change these files. Your code must work with these classes as they are provided.

- The `Dessert` class is an abstract superclass from which specific types of `Dessert`s can be derived. It contains only one field, a name. The `getPrice()` method is an abstract method that is not implemented in the `Dessert` class because the method of determining the prices varies based on the type of item.
- The `DessertTest` class tests your various dessert implementations. Your code must compile with the given code and produce the expected output.

Derived Classes

All of the classes which are derived from the `Dessert` class must define a constructor. Please see the provided `DessertTest` class to determine the parameters for the various constructors and to see the names of the expected methods.

- The `Candy` class should be derived from the `Dessert` class. A `Candy` item has a weight in pounds and a price per pound which are used to determine its price. For example, 2.30 lbs of fudge @ \$0.89 /lb. = \$2.05

The `Candy` constructor takes a String for the name and doubles for weight in pounds and price per pound.

In addition to the methods inherited from `Dessert`, this class provides methods to get the weight in pounds and price per pound.

  - `public double getWeightInPounds()`
  - `public double getPricePerPound()`

- The `Cookie` class should be derived from the `Dessert` class. A `Cookie` item has a number and a price per dozen which are used to determine its price. For example, 4 cookies @ $3.99/dz. = $1.33

  The `Cookie` constructor takes a String for the name, an int for the number of cookies, and a double for the price per dozen.

  In addition to the methods inherited from `Dessert`, this class provides methods to get the number of cookies and price per dozen.

  - `public int getItemCount()`
  - `public double getPricePerDozen()`

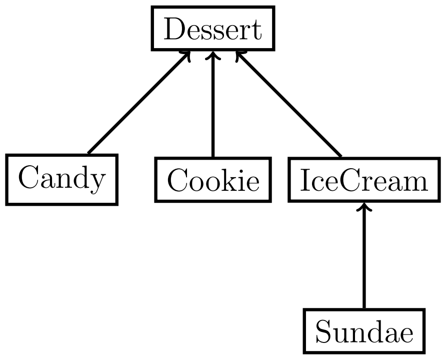
- The `IceCream` class should be derived from the `Dessert` class. An `IceCream` item simply has a price.

  The `IceCream` constructor takes a String for the name and a double for the price.

- The `Sundae` class should be derived from the `IceCream` class. A `Sundae` item is constructed with an `IceCream` item and a topping, which can be any `Dessert` item. The price of a `Sundae` is the price of the `IceCream` plus the price of the topping.

  The `Sundae` constructor takes an `IceCream` object for the base ice cream and a `Dessert` object for the topping. The name of a `Sundae` is the name of the `IceCream` concatenated with the String " topped with " and the name of the topping.

  Do note that since `Sundae` extends `IceCream`, a `Sundae` can be used as the base of another `Sundae`. Likewise, since the topping can be any `Dessert` type, a sundae can be topped with candy, cookie, ice cream, or even another sundae.



Turning in your assignment

Once you are done with your assignment, use Canvas to turn in the java files that you have created. You should turn in a total of four files (code for `Candy`, `Cookie`, `IceCream`, and `Sundae` classes). Do *not* turn in the `Dessert` or `DessertTest` files, since you should not have changed them.