Dessert Shoppe: Polymorphism & Inheritance
Created by Suzanne Balik @ NC State University

Description

For this homework assignment, you will be writing software in support of a Dessert Shoppe which sells candy by the pound, cookies by the dozen, ice cream, and sundaes (ice cream with a topping). Your software will be used for the checkout system.To do this, you will implement an inheritance hierarchy of classes derived froma "DessertItem" *abstract* superclass.

One important feature of this assignment is that it will be run through the use of a main() driver; that is, the program will carry out its execution by itself without any input from the user. You should not assume that the grader will use the exact same driver code. Feel free to make changes so that you can thoroughly test your classes.

You are expected to submit the following source files: Candy.java, Cookie.java, IceCream.java, Sundae.java, and Checkout.java.

Design and Implementation

Let's first take a look at the DessertItem class, which you will use to implement the DessertItem sub-classes.

DessertItem
The DessertItem class is an *abstract superclass* from which specific typesof DessertItems can be derived. It contains only one data member: a name. It also defines a number of methods. All of the DessertItem class methods exceptthe getCost() method are defined in a generic way in the file, DessertItem.java, provided for you along with the other homework specific files. The getCost() method isan *abstract method* that is not defined in the DessertItem class because the method of determining the costs varies based on the type of item. Tax amounts should be rounded to the nearest cent. For example, the calculating the tax on a food item with a cost of 199 cents with a tax rate of 2.0% should be 4 cents.

DO NOT change the DessertItem.java file! Your code must work with this class as it is provided.

DessertShoppe
The DessertShoppe class is also provided for you in the file, DessertShoppe.java. It contains constants such as the tax rate as well the name of the store, the maximum size of an item name and the width used to display the costs of the items on the receipt. *Your code should use these constants wherever necessary*. The DessertShoppe class also contains the cents2dollarsAndCents method which takes an integer number of cents and returns it as a String formatted in dollars and cents. For example, 105 cents would be returned as "1.05".

DO NOT change the DessertShoppe.java file! Your code must work with this class as it is provided.

The Derived Classes (Candy, Cookie, IceCream, and Sundae):
All of the classes which are derived from the DessertItem class must define a constructor. Please see the provided TestCheckout class to determine the parameters for the various constructors. Each derived class should be implemented by creating a file with the correct name, eg., Candy.java.

Candy
The Candy class should be derived from the DessertItem class. A Candy item hasa *weight* and a *price per pound* which are used to determine its *cost*. For example, 2.30 lbs.of fudge @ .89 /lb. = 205 cents. The cost should be rounded to the nearest cent.

Cookie

The Cookie class should be derived from the DessertItem class. A Cookie item hasa *number* and a *price per dozen* which are used to determine its *cost*. For example, 4 cookies @ 399 cents /dz. = 133 cents. The cost should be rounded to the nearest cent.

IceCream and Sundae

The IceCream class should be derived from the DessertItem class. An IceCream item simply has a *cost*. The Sundae class should be derived from the IceCream class. The *cost* of a Sundae is the *cost of the IceCream* plus the *cost of the topping*.

Checkout

The Checkout class provides methods to enter dessert items into the cash register, clear the cash register, get the number of items, get the total cost of the items (before tax), get the total tax for the items, and get a String representing a receipt for the dessert items. The total tax should be rounded to the nearest cent. The complete specifications for the Checkout class are provided for you in JavaDoc format. Do not forget to give Checkout a data member that keeps track of the DessertItems in the form of an array.

Note that while the exact implementation of Candy, Cookie, IceCream, and Sundae is up to you, you should very carefully follow the implementation of Checkout as outlined by Checkout's JavaDoc.

Other notes

Since we are not using dynamic arrays yet, you will have to maintain an instance variable 'numberOfItems,' which keeps tracks of how many DessertItems are in your array. You can hardcode the array size to 100.

Formatting and Style

The most important aspect of this assignment that concerns style is creating a clean, easy to read output. DessertItems and their corresponding prices should be arranged neatly, with a clear sense of alignment and spacing.

Grading Rubric

Points	Criteria
4	Provides complete implementations of Candy, Cookie, IceCream, and Sundae
3	Carefully follows the attached JavaDoc specification for implementation of Checkout
2	All prices are calculated and displayed correctly as per attached example output
1	Good style demonstrated in the code; sensible formatting