Assignment 1

It is suggested that you attempt this assignment on your own. You may use any printed resources you would like (text, notes, Internet – not recommended!). You may ask your instructor and TA questions on a limited basis. This is not a group assignment; however, limited discussion with your classmates is permitted. You may not work with a tutor or receive assistance on this assignment from outside resources. Sharing your work with a classmate or receiving assistance from someone outside of the course is considered a violation of the Mason Honor Code.

LATE ASSIGNMENT IS NOT ACCEPTED after 11:30 p.m. on the due date. Please don't email your late assignments to the instructor or TA as they will be discarded. You need to submit the softcopy to the myMason Portal on or before 11:30 p.m. If not submitted on time, it is classified as LATE.

No global variables are allowed to be used in your application class.

Assignment 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 from a **DessertItem** *abstract* superclass.

The **Candy**, **Cookie**, and **IceCream** classes will be derived from the **DessertItem** class.

The **Sundae** class will be derived from the **IceCream** class.

You will also write a **Checkout** class which maintains an array of **DessertItem's**.

The DessertItem Class

The **DessertItem** class is an *abstract superclass* from which specific types of **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 except the **getCost()** method are defined in a generic way in the file, **DessertItem.java**, provided for you along with the other homework specific files in the directory.
The **getCost()** method is an *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.

The DessertShoppe Class

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

All of the classes which are derived from the **DessertItem** class must define a constructor. Please see the provided **TestCheckout** class,TestCheckout.java , 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**.

The **Candy** class should be derived from the **DessertItem** class. A **Candy** item has a *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.

The **Cookie** class should be derived from the **DessertItem** class. A **Cookie** item has a *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.

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*.

The Checkout Class

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 **Checkout** class must use an array to store the DessertItem's. The total tax should be rounded to the nearest cent. The complete specifications for the **Checkout** class are provided for you in JavaDoc format.

Take-Assignment #1 Rubric (Total Points = 100 points)

Grading Criteria

Submit and Grading

Your code must follow the “style_guidelines.pdf”. Every method and class must be documented using javadoc comments. The documentation for each class must contain the "@author" tag as well as the basic description of the class. The documentation of each method must contain a basic description, the "@param" tag for every parameter, the "@return" tag for any non-void returns, and the "@throws" tag if it throws an exception. For example,

```
 /**  * Inserts the integers in the other IntegerList in this Integer List beginning at  * the given index  * @param other other IntegerList  * @param n index in IntegerList where first integer in the other IntegerList should  *          be inserted  * @throws java.lang.ArrayIndexOutOfBoundsException  */   public void insert(IntegerList other, int n)                          throws ArrayIndexOutOfBoundsException {
```

Submit the **5** source files, **Candy.java**, **Cookie.java**, **IceCream.java**, **Sundae.java**, and **Checkout.java**. **Please see the gradesheet for the point distribution.**

Reference

Author of Assignment: Suzanne Marie Prem Balik , NCSU