CSC 143 – Weekly Exercise: Satisfying Slurps

For this assignment, imagine you have been hired to create a program to manage drink orders at a local coffee shop. The coffee shop is known for changing their menu frequently, and people come to the shop in anticipation of getting unusual, but delicious, drinks. Since the drinks change so frequently, your program will need to have the flexibility to handle this. Accordingly, you decide to create classes that can be used to price drinks based on their common features, rather than creating a static look-up table of prices.

To demonstrate how your program will work to the coffee shop owner and employees, you want to design and demo a prototype of your idea. To do this, you must implement a *Drinks.java* Class and associated subclasses that will be utilized by a *CoffeeShop.java* Class (the driver, or client, class) to calculate the cost of an order of drinks. The output will be displayed in a java console. This assignment is intended to give you practice using inheritance.

The Drinks Class will contain fields for the base price/oz. and the drink size (small = 6 oz., medium = 12 oz., large = 16 oz.). In essence, this covers the minimal costs (labor, cup, etc.). It will include also methods to print the drink information and to change the base drink prices (ideally all the prices should be changed simultaneously).

A number of subclasses will extend the Drinks class. You are being asked to implement a *CaffeinatedDrinks.java* class and a *NonCaffeinatedDrinks.java*. Both of these classes extend the Drinks class. In addition, you are being asked to implement a *Tea.java* class and a *Coffee.java* class that extends the *CaffeinatedDrinks* class.

Data for Sample Program:

Drinks Class:

- **Base Drink price: 10 cents/oz.**
- **Default drink size = small (6 oz.)**

CaffeinatedDrinks Class:

- **Base Drink price: 3 times the base price for the Drinks class**
- **Default drink size = small (6 oz.)**

NonCaffeinatedDrinks Class:

- **Base Drink price: 2 times the base price for the Drinks class**
- **Default drink size = small (6 oz.)**

Coffee Class:

- **Base Drink price: Same as CaffeinatedDrinks class for small size, but a surcharge of 50 cents for medium and $1 for large to cover the cost of extra shots needed**
- **Default drink size = small (6 oz.)**
- **Default drink type = drip**

Tea Class:

- **Base Drink price: same as for CaffeinatedDrinks class, except no surcharge for larger sizes, as it is just adding more water**
- **Default drink size = small (6 oz.)**
- **Default tea flavor = English Breakfast**

Here are the requirements:

- **Use the attached drink class hierarchy to organize your code.**
- **Implement the classes to minimize code redundancy by extending superclasses.**
- **For this prototype, you can have the tea flavors and coffee drink types entered in as parameters. Future versions of the program might have a different way of handling this. Create a no-parameter constructor and a constructor that takes an appropriate number of parameters for each class.**
- **Write methods to return basic drink information (size and price for all drinks).**
- **Also write the client Class *CoffeeShop.java* that demonstrates your classes and their methods. This class should contain a main method that utilizes an array of Drinks of different types. I should be able to run your code without creating a new client!**

Description of the client Class:

1. **Should print a sample drink order – The information included will contain: Size of drink, Type of drink, Price of Drink**
2. **Should print the TOTAL price for the order**

Sample Output (you may choose to format your output differently):

```
Your order consists of:

        water, size small, cost: $0.60
        coffee, type mocha, size medium, cost: $2.30
        tea, flavor earl grey, size large, cost: $1.20

The total cost of your order is: $4.10
```

Submit the following using Canvas online assignment submission:

1. **A zipped folder containing all source-code files (*.java), including a client Class that includes a main method to demonstrate your classes.**
2. **Attach the above folder when submitting this weekly exercise.**

Class Hierarchy (constructors omitted):