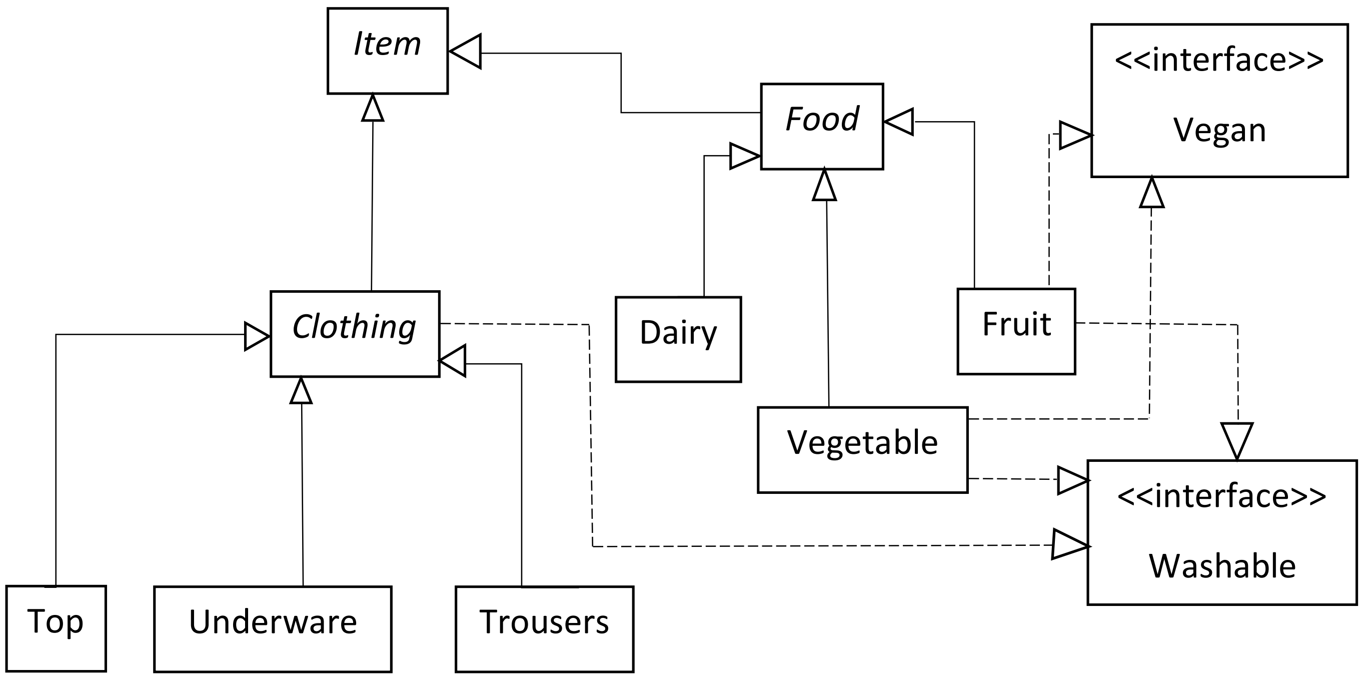
CSE 1141 - COMPUTER PROGRAMMING II
Programming Assignment # 2
DUE DATE: 25/03/2018 - 23:59 (No extension)

In this homework, you will implement a shopping mall simulator program with an object-oriented approach. The class hierarchy of the program will be as following:



1. Implement a `Vegan` interface with the following UML diagram.

<<interface>> *Vegan*	
+	*madeOf():void*

- The `Vegan` interface represents the property of not having any animal product.
- The `madeOf` abstract method will be implemented in the classes that implements *Vegan* interface to show what the item is made of.

2. Implement a `Washable` interface with the following UML diagram.

	<<interface>> *Washable*
+	*howToWash():void*

- The `Washable` interface represents the property of being washable.
- The `howToWash` abstract method will be implemented in the classes that implements `Washable` interface to show the instructions on how to wash an item.

3. Implement an `Item` abstract class with the following UML diagram.

	Item
-	vat:double
-	basePrice:double
+	getVat():double
+	setVat(vat:double):void
+	getBasePrice():double
+	setBasePrice(price:double):void
+	*calculatePrice():double*

- The `Item` class represents anything that is being sold at the shopping mall.
- The data field `vat` represents value added tax for each item.
- The data field `basePrice` represents the price of the item when there is no taxes or profit added.
- `getVat` and `setVat` methods are accessor methods for `vat` data field.
- `getBasePrice` and `setBasePrice` methods are accessor methods for `basePricedata` field.
- `calculatePrice` abstract method will be implemented in subclasses to calculate the final price of the item.

4. Implement a `Clothing` abstract class with the following UML diagram.

	Clothing
+	Clothing();

- The `Clothing` class represents any piece of clothing.
- You have to implement the constructor to set `vat` data field to 0.18.
- `Clothing` class should extend `Item` class and implement `Washable` interface.

5. Implement a `Food` abstract class with the following UML diagram.

	Food
+	Food();

- The `Food` class represents any kind of food.
- You have to implement the constructor to set `vat` data field to 0.08.
- `Food` class should extend `Item` class.

6. Implement a `Dairy` class with the following UML diagram.

	Dairy
+	Dairy()
+	Dairy(basePrice:double)

- A `Dairy` object represents a dairy product.
- It extends `Food` class.
- The no-arg constructor calls other constructor with the value 8.0.
- Argumented construtor sets basePrice to the given value.
- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the dairy product. You have to calculate retail price by adding 30% profit to the taxed price.

7. Implement a `Fruit` class with the following UML diagram.

	Fruit
+	Fruit()
+	Fruit(basePrice:double)

- A `Fruit` object represents a real-life fruit
- It extends `Food` class and implements `Vegan` and `Washable` interfaces.
- The no-arg constructor calls other constructor with the value 6.0.
- Argumented construtor sets basePrice to the given value.

- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the fruit. You have to calculate retail price by adding 20% profit to the taxed price.
- This class has to implement `howToWash` method as well. The method has to print “`Wash Fruit with cold water.`” on the screen.
- This class has to implement `madeOf` method as well. The method has to print “`It is made only of fruits.`” on the screen.

8. Implement a `Top` class with the following UML diagram.

Top	

+
+
	Top()
	Top(basePrice:double)

- A `Top` object represents a real-life top clothing.
- It extends `Clothing` class.
- The no-arg constructor calls other constructor with the value 20.0.
- Argumented construtor sets basePrice to the given value.
- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the top product. You have to calculate retail price by adding 20% profit to the taxed price.
- This class has to implement `howToWash` method as well. The method has to print “`Wash Top at 40 degrees.`” on the screen.

9. Implement a `Trousers` class with the following UML diagram.

Trousers	

+
+
	Trousers()
	Trousers(basePrice:double)

- A `Top` object represents a real-life trousers.
- It extends `Clothing` class.
- The no-arg constructor calls other constructor with the value 4   0.0.
- Argumented construtor sets basePrice to the given value.

- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the trousers. You have to calculate retail price by adding 20% profit to the taxed price.
- This class has to implement `howToWash` method as well. The method has to print "`Wash Trousers at 30 degrees.`" on the screen.

10. Implement an `Underware` class with the following UML diagram.

	Underware
+	Underware()
+	Underware(basePrice:double)

- A `Top` object represents a real-life underware.
- It extends `Clothing` class.
- The no-arg constructor calls other constructor with the value 30.0.
- Argumented construtor sets basePrice to the given value.
- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the underware. You have to calculate retail price by adding 45% profit to the taxed price.
- This class has to implement `howToWash` method as well. The method has to print "`Wash Underware at 60 degrees.`" on the screen.

11. Implement a `Fruit` class with the following UML diagram.

	Vegetable
+	Vegetable()
+	Vegetable(basePrice:double)

- A `Vegetable` object represents a real-life fruit
- It extends `Food` class and implements `Vegan` and `Washable` interfaces.
- The no-arg constructor calls other constructor with the value 10.0.
- Argumented construtor sets basePrice to the given value.
- This class has to implement `calculatePrice` method as well. The method has to return the retail price of the vegetable. You have to calculate retail price by adding 25% profit to the taxed price.
- This class has to implement `howToWash` method as well. The method has to print "`Wash Vegetable with warm water.`" on the screen.

- This class has to implement `madeOf` method as well. The method has to print `“It is made only of vegetables.”` on the screen.

12. Implement a `ShoppingMall` class with the following UML diagram.

ShoppingMall	
-	items:ArrayList<Item>
+	getItems():ArrayList<Item>
+	addDairy():void
+	addFruit():void
+	addTop():void
+	addTrousers():void
+	addUnderware():void
+	addVegetable():void
+	addArbitrary(item:Item):void
+	bill():double

- A `ShoppingMall` object represents a shopping malls.
- The data field `items` keeps all the items in the shopping mall.
- `getItems` method is getter for items ArrayList.
- `addDairy` method adds a default `Dairy` object to `items` ArrayList.
- `addFruit` method adds a default `Fruit` object to `items` ArrayList.
- `addTop` method adds a default `Top` object to `items` ArrayList.
- `addTrousers` method adds a default `Trousers` object to `items` ArrayList.
- `addUnderware` method adds a default `Underware` object to `items` ArrayList.
- `addVegetable` method adds a default `Vegetable` object to `items` ArrayList.
- `addArbitrary` method adds the given `Item` object to `items` ArrayList.
- `bill` method iterates through the `items` ArrayList and calculates the total price of items in the ArrayList.

13. Write a test program in which the following are performed in order.
    a. Create a new `ShoppingMall` object.
    b. Add a default `Dairy`, a default `Fruit`, a default `Top`, a default `Trousers`, a default `Underware`, a default `Vegetable` object to the `ShoppingMall` object.
    c. Add a `Top` object with `basePrice` = 100
    d. Create a `printContent` method that invokes `madeOf` method of the all the `Vegan` instances inside the `ShoppingMall’s` ArrayList.
    e. Create a `printWashingInstructions` method that invokes `howToWash` method of the all the `Washable` instances inside the `ShoppingMall’s` ArayList

f. Call these two methods inside main method.
g. Print the bill of the `ShoppingMall` object.

This is a simple scenario to test your class implementations. There might be other test cases too. Therefore, please pay attention to use the same class, method and variable names in your implementations. You are allowed to increase the number of methods in the classes; however, you cannot decrease the number of them.

Submission Instructions

Please zip and submit your files using filename YourNumberHW2.zip
(ex: 150713852HW2.zip) to Canvas system (under Assignments tab).
Your zip file should contain the following 26 files:

1. 13 Java source files: Clothing.java, Dairy.java, Food.java, Fruit.java, Item.java, ShoppingMall.java, Test.java, Top.java, Trousers.java, Underware.java, Vegetable.java, Vegan.java, Washable.java
2. 13 Java class files: Clothing.class, Dairy.class, Food.class, Fruit.class, Item.class, ShoppingMall.class, Test.class, Top.class, Trousers.class, Underware.class, Vegetable.class, Vegan.class, Washable.class

Notes:

1. Write a comment at the beginning of each program to explain the purpose of the program.
2. Write your name and student ID as a comment.
3. Include necessary comments to explain your actions.
4. Select meaningful names for your variables and class names.
5. You are allowed to use the materials that you have learned in lectures & labs.
6. Do not use things that you did not learn in the course.
7. In case of any form of **copying and cheating** on solutions, all parts will get **FF** grade. You should submit your own work. In case of any forms of cheating or copying, both giver and receiver are equally culpable and suffer equal penalties.
   All types of plagiarism will result in FF grade from the course.
8. No late submission will be accepted.

Grading:

- Clothing Class → 5 points
- Dairy Class → 5 points
- Food Class → 5 points
- Fruit Class → 7 points
- Item Class → 7 Points
- ShoppingMall Class → 11 points
- Test Class → 7 points
- Top Class → 7 points
- Trousers Class → 7 points
- Underware Class → 7 Points
- Vegeable Class → 7 points
- Vegan Class → 5 points
- Washable Class → 5 points
- The correctness of test cases → 15 points