German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Computer Programming Lab,
Lab Assignment1

This lab assignment aims at modifying the work of last week's home assignment, "Spaghetti", by introducing the concepts of Inheritance and Encapsulation into the project.

Supermarkets offer various Grocery products. In our model, we are only concerned with two types of Grocery products; `DairyProduct` and `Beverage`. These two types of Grocery products share certain traits and characteristics while they differ in others. Your task is to develop a hierarchy of classes and apply the concepts of Inheritance and Encapsulation to produce an organized code.

Exercise 1-1

Packages

Programmers typically use **packages** to organise classes belonging to the same category or providing similar functionality. Thus for this project, you must organise your work into the following packages:
`guc.supermarket`
`guc.supermarket.products`

Inheritance

Java Tips

- The attributes that are shared by all subclasses should be declared in the superclass.

Exercise 1-2

GroceryProduct Class

The class `GroceryProduct` is a **generic** class. It should have all variables and methods that apply generically to all grocery products. There are 2 types of Grocery Products: Dairy Products and Beverages.

Exercise 1-3

DairyProduct Class

Every supermarket offers some dairy products like Milk, Cheese, Jogurt, etc. In our supermarket project we will represent the dairy products using the `DairyProduct` class obeying the following set of characteristics:

- It should be placed in the `guc.supermarket.products` package.
- Each dairy product should have a `name`, `price`, and `discount`(0%-100%).
- Each dairy product should have a measure of the contained fat. The contained `Fat` of a dairy product is usually classified as `FULLCREAM`, `HALFCREAM`, or `SKIMMED`. This should be represented using Enumeration.
- Appropriate constructors should be implemented, in order to allow creating dairy products while specifying their name, price, discount and fat. Note that the parameters of the constructor should be passed in the previously given order.

Exercise 1-4

Beverage class

Supermarket offers some beverages like fizzy drinks, juices, etc. In our supermarket project we will represent the beverages using the `Beverage` class obeying the following set of characteristics:

- It should be placed in the `guc.supermarket.products` package.
- Each beverage should have a `name`, `price`, and `discount`(0%-100%).
- Each beverage should have a measure of the contained sugar level. The `Sugar level` could be `LIGHT`, `ZERO`, `ADDED_SUGAR`, or `NO_ADDED_SUGAR`. This should be represented using Enumeration.
- Appropriate constructors should be implemented, in order to allow creating beverages while specifying their name, price, discount and sugar level. Note that the parameters of the constructor should be passed in the previously given order.

-Inheritance - Overriding

Java Tips

- The method that has the same behaviour for all subclasses should be implemented once in the superclass, and thus inherited by all the subclasses.
- The method that is common between all subclasses but has a different behaviour for each and every subclasses should be declared in the superclass and overridden in each subclass.
- A method is declared as FINAL if it is not to be overridden by any subclass.

Exercise 1-5

DairyProduct Class

The `DairyProduct` class has the following methods:

- A method getActualPrice() should be implemented to get the actual price of the dairy product, i.e. original price - discount amount. ($DiscountAmount = price \times (discount/100)$)
- The `String toString()` method should be overridden to produce a string representing all information about the product. For example:
  Name: Juhayna Milk
  Price: 10.0 L.E.
  Discount: 25.0 %
  Fat Level: Half-Cream

Exercise 1-6

Beverage class

The `Beverage` class has the following methods:

- A method `getActualPrice()` should be implemented to get the actual price of the beverage, i.e. original price - discount amount. ($DiscountAmount = price \times (discount/100)$)
- The `String toString()` method should be overridden to produce a string representing all information about the product. For example:
  Name: Coke Zero
  Price: 5.0 L.E.
  Discount: 12.5 %
  Sugar Level: ZERO

Exercise 1-7

Do you get overriding?

Given the following piece of code:

```
DairyProduct milk= new DairyProduct("Juhayna Milk", 10, 5, Fat.FULLCREAM);
System.out.println(milk.toString());
```

Which `toString()` will be invoked?
The `toString()` method in the class: `DairyProduct`, `GroceryProduct` or `Object`?

Exercise 1-8

Testing our code

In order to make sure that everything is working correctly, we will test using JUnit tests.

Download the test file `Lab2PublicTest` from the MET website.

Test files should be placed in a separate package for a better organization.

You should place the test file in the package `guc.supermarket.tests`.

Encapsulation

Java Tips

- You cannot override a method in a subclass giving it a more restricted access modifer than that of the superclass.
- Use the protected access modifer when a superclass should provide a method only to its subclass and other classes in the same package, but not to other classes.

Instance variables should be declared as private and appropriate methods should be implemented to manipulate them.

It is highly recommended that instance variables of superclasses are declared as private, rather than protected or public, and non-private methods are implemented to access such variables. Thus ensuring that all variables can be accessed from subclasses.

Exercise 1-9

Encapsulation

The objective of this exercise is to re-declare the instance variables of the classes you have previously implemented to be private variables with READ and WRITE access given to the subclasses.

Make sure you update the variables in all the classes: `GroceryProduct`, `DairyProduct` and `Beverage`

For example, the instance variable `name` of the `GroceryProduct` class should be declared as follows:

```
private String name;
```

And the appropriate methods should be implemented as follows:

```
String getName(){
    return this.name;
}

void setName(String name){
    this.name = name;
}
```