

FRESHER ACADEMY

BASIC JAVA

Ngattt (FHO.FA)

04/20/2022

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

Contents


Objective2

Business needs2

Working requirements3

Technologies.....3

Project Descriptions3

	Assignment topic : Java Basic Lab Assignment duration : 60 minutes	FRESHER ACADEMY
---	---	----------------------------

Objective

- Fresher can apply knowledge about Class and Object to code a simple application.
- Coffee Shop with Different Types of Beverages: Suppose you are building a coffee shop that offers different types of beverages such as coffee, tea, and smoothies. Create a class called Beverage with instance variables for name and price, ingredients. Then create subclasses called Coffee, Tea, and Smoothie that add the appropriate instance variables for each type of beverage.
 - o Implement appropriate constructors, getters, and setters for all classes.
 - o Implement a method called makeBeverage() in the Beverage class that prints out the name and price of the beverage.
 - o The Coffee and Tea classes should override the makeBeverage() method to include the type of coffee or tea being made.
 - o The Smoothie class should override the makeBeverage() method to include the fruits used in the smoothie.
- Fresher can create abstract class, subclasses
- Fresher can write methods and override methods
- Fresher can create some instances of class and test all the methods of the class.

Business needs

- Users can create as many instances of Employee, ITEmployee, HREmployee, MarketingEmployee as they want. After tests all the methods of the instances, the program will print like this:

```

1. Coffee: Latte costs $3.5
- Making Latte...
  + Brewing coffee...
  + Adding milk.
  -> Done!
2. Tea: Green Tea costs $2.0
- Making Green Tea...
  + Steeping tea leaves in hot water.
  + Adding sugar.
  -> Done!
3. Smoothie: Berry Blast costs $5.0
- No Ingredient: []
- Ingredients: [Banana, Milk]
- Preparing Berry Blast (Strawberry, Blueberry, Raspberry) for $5.0

```

- Do not verify information user input

Working requirements

- Working environment: Eclipse/IntelliJ/NetBeans.

Technologies

- The product implements Java program language with: Abstract class, Subclasses

Project Descriptions

1. **Step 1: Create a new Java project** (skip this step if you already have **YourFullName_JavaSE** project)
 - Open your preferred Java IDE (such as Eclipse, IntelliJ, or NetBeans).
 - Create a new Java project by selecting File > New > Java Project.
 - Name the project **YourFullName_JavaSE**, for example **"NguyenVanA_JavaSE"** and click Finish.
2. **Step 2: Create a package** (skip this step if you already have **lab4** package)
 - In the project explorer, right-click on the src folder and select New > Package.
 - Name the package "lab4" and click Finish.
3. **Step 3: Create an **abstract** class**
 - In the Package Explorer panel on the left side of the screen, right-click on the 'lab4' package to create the class.
 - Select "New" from the context menu, then "Class" from the submenu.
 - In the "New Java Class" dialog box, enter "**Beverage**" for the class in the "Name" field.
 - Click checkbox **abstract**.
 - Click "Finish" to create the class.
4. **Step 4: Create a Coffee class**
 - In the Package Explorer panel on the left side of the screen, right-click on the 'lab4' package to create the class.
 - Select "New" from the context menu, then "Class" from the submenu.
 - In the "New Java Class" dialog box, enter "**Coffee**" for the class in the "Name" field.
 - Click "Finish" to create the class.
5. **Step 5: Create a Tea class**
 - In the Package Explorer panel on the left side of the screen, right-click on the 'lab4' package to create the class.
 - Select "New" from the context menu, then "Class" from the submenu.
 - In the "New Java Class" dialog box, enter "**Tea**" for the class in the "Name" field.
 - Click "Finish" to create the class.

6. Step 6: Create a Smoothie class

- In the Package Explorer panel on the left side of the screen, right-click on the 'lab4' package to create the class.
- Select "New" from the context menu, then "Class" from the submenu.
- In the "New Java Class" dialog box, enter "Smoothie" for the class in the "Name" field.
- Click "Finish" to create the class.

7. Step 7: Create a BeverageTest class for testing

- In the Package Explorer panel on the left side of the screen, right-click on the 'lab4' package to create the class.
- Select "New" from the context menu, then "Class" from the submenu.
- In the "New Java Class" dialog box, enter "BeverageTest" for the class in the "Name" field.
- Click "Finish" to create the class.

8. Step 8: Coding attributes and methods for Beverage (**abstract class**)

- Attributes:

```
protected String name;  
protected double price;  
protected ArrayList<String> ingredients;
```

- Generate getter and setter for attributes:
 - Right click on the class
 - Select "Source" from the context menu. Click "Generate Getters and Setters".
 - In the "Generate Getters and Setters" dialog box, select the fields for which you want to generate getters and setters.
 - Choose whether you want to generate getters, setters, or both, and click the "OK" button. Eclipse will generate the getters and setters for you in the appropriate format, and add them to your Java file.
 - Note: You can also use keyboard shortcut "Alt + Shift + S" and then select "Generate Getters and Setters" from the options.
- Methods:

```
public String[] getIngredients() {  
    return ingredients.toArray(new String[0]);  
}  
  
public void addIngredient(String ingredient) {  
    ingredients.add(ingredient);  
}  
public void makeBeverage() {  
    System.out.println("Preparing " + name + " for $" + price);  
}
```

9. Step 9: Coding attributes and methods for Coffee

- Attributes: **private boolean hasMilk;**
- Methods: write **makeBeverage()** method

```
@Override
public void makeBeverage() {
    System.out.println("- Making " + getName() + "... ");
    System.out.println("\t+ Brewing coffee...");
    if (hasMilk()) {
        System.out.println("\t+ Adding milk.");
    }
    System.out.println("\t-> Done!");
}
```

10. Step 10: Coding attributes and methods for Tea class

- Attributes: **private boolean hasSugar;**
- Methods: write **makeBeverage()** method

```
@Override
public void makeBeverage() {
    System.out.println("- Making " + getName() + "... ");
    System.out.println("\t+ Steeping tea leaves in hot water.");
    if (hasSugar()) {
        System.out.println("\t+ Adding sugar.");
    }
    System.out.println("\t-> Done!");
}
```

11. Step 11: Coding attributes and methods for Smoothie

- Attributes: **private String[] fruits;**
- Methods: write **makeBeverage()** method

```
@Override
public void makeBeverage() {
    System.out.print("- Preparing " + name + " (");
    for (int i = 0; i < fruits.length; i++) {
        System.out.print(fruits[i]);
        if (i != fruits.length - 1) {
            System.out.print(", ");
        }
    }
    System.out.println(") for $" + price);
}
```

12. Step 12: Coding main() method for BeverageTest class

Write **main()** method to create some instances of Coffee, Tea, Smoothie and then test all the methods:

```

// Coffee
Beverage coffee = new Coffee("Latte", 3.50, true);
System.out.println("1. Coffee: " + coffee.getName() + " costs $" + coffee.getPrice());
coffee.makeBeverage();

// Tea
Beverage tea = new Tea("Green Tea", 2.00, true);
System.out.println("2. Tea: " + tea.getName() + " costs $" + tea.getPrice());
tea.makeBeverage();

// Smoothie
String[] fruits = { "Strawberry", "Blueberry", "Raspberry" };
Beverage smoothie = new Smoothie("Berry Blast", 5.00, fruits);
System.out.println("3. Smoothie: " + smoothie.getName() + " costs $"
    + smoothie.getPrice());

// test getIngredients() and addIngredient() methods
System.out.println("- No Ingredient: " + Arrays.toString(smoothie.getIngredients()));
smoothie.addIngredient("Banana");
smoothie.addIngredient("Milk");
System.out.println("- Ingredients: " + Arrays.toString(smoothie.getIngredients()));
smoothie.makeBeverage();

```

Note: add more instances in the main method!

The End!