Defining Class

- Define a **class Product** with the following data members :
  - name (String)
  - price (int)

Info : Classes which are defined just for the sake of saving data are called Model Classes or POJO (Plain Old Java Object). Here **class Product** is an example. From now on, define every such model class in separate files. **DO NOT** define these classes within a class with the main() method!

Defining Constructors

- Define an **empty/default constructor** for it
- Define a **parameterized constructor** passing name & price to it (Use Generator (Alt + Insert) & '**this**' keyword)

Overriding toString() method

- Override **toString()** and using **String.format()** return string in format *"Name : %s @ Rs. %d"*

  Tip : Google for String.format() method if you have confusion

Overriding equals() method

- Learn about the **equals()** method of Object class. Observe it's default behaviour.
- In Java, Strings are compared using **.equals()** method and not using '==' . We can also compare objects using **.equals()**. Observe the following example :

```
Product apple = new Product( name: "Apple",  price: 100);
Product apple1 = new Product( name: "Apple",  price: 100);
Product orange = new Product( name: "Orange",  price: 80);

System.out.println(apple.equals(apple1)); //Default behaviour : false, required : true
System.out.println(apple.equals(orange)); //returns false
```

Override the .equals() method of **Product class** for this required behaviour.

Tip : Google about .equals() to learn more

Inheritance

- Define another **class SpecialProduct** extending **Product class** with following data members :

    - regularPrice (int)
    - percentageOff (int)
- Define a parameterized Constructor for **class SpecialProduct** passing name & price. Invoke **super(name, price)** in it.
- Override **toString()** for **class SpecialProduct** and return the required formatted string.

Static Method

- Define a static method -

  `SpecialProduct applyOffOnProduct(Product product, int percentageOff)`

  in **class SpecialProduct** as follows :
    - Calculate **discountedPrice** of product by applying percentageOff to product.price
    - Create a new object of **class SpecialProduct,** call its constructor and pass **(product.name, discountedPrice)** to it
    - Set **regularPrice = product.price**
    - Set **this.percentageOff = percentageOff**
    - Return the object

Driver Code

- In the **Main class**,
    - Create 3 products of your choice but give same name & price to 2 of the products
    - Print the products
    - Print equality of the 2 similar products using .equals()
    - Create a new object of **class SpecialProduct** by calling the static method **applyOffOnProduct** and pass any product along with some offPercentage of your choice
    - Print this specialProduct object

Submission

- Create a new repository & an IntelliJ project dedicated just for such tasks we will provide
- Create a new package inside src folder named “Task1” and place all of the files for this task in it
- Push the code to GitHub
- Fill this form