Description

Create a Bicycle Store in Java. Our example is basic one, which have the following features:
1. storage for the bikes
2. ability to add the bike in the store
3. ability remove the bike from the store
4. ability to print the library bike information on the console

BSS Structure:

We will need the following classes for the software:
1. Bicycle– the bike itself.
2. BicycleStore – Bicycle store management system.
3. BicycleStoreTester – the tester class. This class will be used to test our management system.

Class Bicycle	
String	**brand**
String	**model**

Class BicycleStore	
List<Bicycle>	**inventory**
Void addBicycle(Bicycle bicycle)	
Boolean sellBicycle(Bicycle bicycle)	
Void printInventory()	

Class Bicycle

The class Bicycle should have several fields, including brand and model. This class can be implemented in the following way:

```java
package dimitri_durmishian_1.task3;

public class Bicycle {
    private String brand, model;

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    @Override
    public String toString() {
        return "Bicycle{" +
                "brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                '}';
    }
}
```

Class BicycleStore

The Bicycle Store management system should have an inner structure for storing bikes. The management system should have methods for adding the new bikes and selling them. It should have the ability to print the entire store content when needed. The class can be implemented in the following way:

```java
package dimitri_durmishian_1.task3;

import java.util.ArrayList;
import java.util.List;

public class BicycleStore {
    // Inventory of bicycles in the store
    private final List<Bicycle> inventory = new ArrayList<>();

    // Add a bicycle to the store's inventory
```

```
    public void addBicycle(Bicycle bicycle) {
        if (!inventory.contains(bicycle)) {
            inventory.add(bicycle);
        } else {
            System.out.println("The bicycle is already in the inventory.");
        }
    }

    // Remove a bicycle from the store's inventory
    public boolean sellBicycle(Bicycle bicycle) {
        return inventory.remove(bicycle);
    }

    // Print the current inventory of bicycles in the store
    public void printInventory() {
        if (inventory.isEmpty()) {
            System.out.println("The inventory is empty");
        } else {
            for (Bicycle bike : inventory) {
                System.out.println("Brand: " + bike.getBrand() + ", Model: " +
bike.getModel());
            }
        }
    }
}
```

BicycleStoreTester Class

Now let's test our management system. First, create some bikes. Then add those bikes to the store. Add the bike which is already in the store. Then try to sell some of the bikes.

```
package dimitri_durmishian_1.task3;

public class BicycleStoreTester {
    public static void main(String[] args) {
        BicycleStore store = new BicycleStore();

        // Creating sample bicycles
        Bicycle bike1 = new Bicycle();
        bike1.setBrand("Trek");
        bike1.setModel("FX");

        Bicycle bike2 = new Bicycle();
        bike2.setBrand("Giant");
        bike2.setModel("Defy");

        // Adding bicycles to the store's inventory
        store.addBicycle(bike1);
```

```
        store.addBicycle(bike1); // Attempting to add the same bicycle again
        store.addBicycle(bike2);

        // Printing the current inventory of bicycles in the store
        store.printInventory();
    }
}
```

We print the state of the store to check if all the methods are working properly.