Garden Management System

Nikoloz Eriashvili

Description

Create a Garder Management System (GMS) in Java. GMS is widely used software. It can be any complexity. Our example is basic one, which have the following features:

1. storage for the flowers
2. ability to grow a flower
3. ability remove the flower from your possession
4. ability to print the collection of your flowers with their properties

GMS structure

We will need the following classes for the software:

1. Flower– the flower itself.

2. GMS – Garden management system.

3. GardenTester – the tester class. This class will be used to test our management system.

Class Flower

The class Flower should have several fields, including species, color and allergyProvoking. This class can be implemented in the following way:

```java
package Garden;

public class Flower {
   private String species, color;
   private Boolean allergyProvoking;



   public String getSpecies() {
       return species;
   }

   public void setSpecies(String species) {
       this.species = species;
   }

   public String getColor() {
       return color;
   }
   public void setColor(String color) {
       this.color = color;
   }
   public Boolean getAllergyProvoking() {
       return allergyProvoking;
   }
   public void setAllergyProvoking(Boolean allergyProvoking) {
       this.allergyProvoking = allergyProvoking;
   }
}
```

Pay attention to the setters and getters of the fields. In general, all the fields are private (unless the special requirements are stated) and the access functions are implemented such as setters and getters.

Read about the toString() function and implement it for Flower class.

Class LMS

The Garden management system should have an inner structure for storing flowers. The management system should have methods for adding the new flowers and removing the old ones. It should have the ability to print the entire garden content when needed. The class can be implemented in the following way:

```
package Garden;
import java.util.ArrayList;
import java.util.List;
public class LMS {
   // Mapping with Flower and the number of this flowers in the
library
   private List<Flower> garden = new ArrayList<Flower>();
   // adds the flower to the garden
   public void addFlower(Flower flower) {
       garden.add(flower);
   }
   // removes the flower from the garden
   public boolean removeFlower(Flower flower) {
       boolean removed = false;
       for (int i = 0; i < garden.size(); i++) {
           Flower f = garden.get(i);
           if (f.getSpecies().equals(flower.getSpecies()) &&
f.getColor().equals(flower.getColor())) {
               garden.remove(i);
               removed = true;
               break;
           }
       }
       return removed;
   }
   public void printGarden() {
       if (garden.isEmpty()) {
           System.out.println("The garden is empty");
       } else {
           for (Flower f: garden) {
               System.out.println(f.getSpecies() + ", " +
f.getColor());
               System.out.println();
           }
```

```
        }
    }
}
```

Pay attention to the usage of the ArrayList class, for loops for the lists, break clause and the String object comparison. It is a good point to understand how Interface works. Usage of the boolean variables can also be observed in this example.

GMS Tester class

Now let's test our management system. First, create some flowers. Then create GLM and add those flowers to the garden using the GLM. Then try to remove some of the flowers.

```
package Garden;

public class GardenTester {
    public static void main(String[] args) {

        Flower f1 = new Flower();
        f1.setSpecies("Rose");
        f1.setColor("Red");
        f1.setAllergyProvoking(false);

        Flower f2 = new Flower();
        f2.setSpecies("Tulip");
        f2.setColor("Orange");
        f2.setAllergyProvoking(true);

        LMS lms = new LMS();
        lms.addFlower(f1);
        lms.addFlower(f1);
        lms.addFlower(f2);
        lms.removeFlower(f1);
        lms.printGarden();
    }
}
```

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