Tutorial of Composition and static field

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech
>
> Designed by ZHU Yueming in 2024. April. 8th

Objective

- Learn to static class members
- Learn to use ArrayList

Part 1: Static class member

Question Description:

Continue to use the `Circle` class we used before.

(1) Adding a new field id, which represents the created order of the current Circle object. For example, if one circle is the first created, its id is 1, and if one circle is the second created, its id is 2.

(2) The instance method `toString()` returns a String which include all the data field value as desired format.

```
Circle #[id]: radius = [radius], x = [x], y = [y]
```

The sample main method would be:

```
public static void main(String[] args) {
        Random random = new Random();
        Circle[] circles = new Circle[random.nextInt(3) + 3];
        for (int i = 0; i < circles.length; i++) {
            double radius = random.nextDouble() * 2 + 1;
            double x = random.nextDouble() * 10 - 5;
            double y = random.nextDouble() * 10 - 5;
            circles[i] = new Circle(radius, x, y);
        }
        for (Circle c: circles) {
            System.out.println(c);
        }
    }
```

The output would be:

```
Circle #1: radius = 2.33, x = 3.77, y = 0.31
Circle #2: radius = 2.62, x = 0.43, y = 2.41
Circle #3: radius = 2.94, x = -0.04, y = -2.07
Circle #4: radius = 2.38, x = 3.13, y = 2.42
Circle #5: radius = 2.49, x = 1.70, y = 3.22
```

1. Add static class number

Add a static class number `count`, which records how many circle has been created.

```
private static int count = 0;
```

Add a privated data field `id`, which represents the id of current object.

```
private int id;
```

2. Modify Constructor:

When creating a circle object, increasing the count, and giving its value to id.

```
public Circle() {
        this.id = ++count;
    }

public Circle(double radius, double x, double y) {
        this.id = ++count;
        this.radius = radius;
        this.x = x;
        this.y = y;
    }
```

3. Modify toString Method:

```
public String toString() {
    return String.format("Circle #%d: radius = %.2f, x = %.2f, y = %.2f", id,
radius, x, y);
}
```

4. How to visit the value of static field?

We know if we want to visit a private member field, we can use getter and setter method. If the field is a static, how can we design the getter and setter method?

Design:

```
 public static int getCount() {
        return count;
 }

public static void setCount(int count) {
        Circle.count = count;
}
```

Invoke:

```
System.out.printf("There are %d Circles:\n",Circle.getCount());
```

Part 2: ArrayList

How to use ArrayList

An ArrayList servers as a resizable-array that stores data based on an array structure. Each ArrayList instance has a capacity. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayList, its capacity grows automatically.

The elements in an ArrayList must be objects of a class, and they cannot be primitive data types. However, we can use wrapper classes for primitive data types to store data.

Create an Integer List

```
ArrayList<Integer> numbers = new ArrayList<>();
```

You should write `Integer` in `<>` instead of `<int>`, becaue `Integer` is the wrapper class of `int`.

isEmpty() and size()

```
System.out.println(numbers.isEmpty());
System.out.println(numbers.size());
```

add value into ArrayList

```
numbers.add(1);
numbers.add(3);
numbers.add(5);
numbers.add(7);
```

Traverse ArrayList

```
for (int i = 0; i < numbers.size(); i++) {
     System.out.println(numbers.get(i));
}

for (int e:numbers) {
     System.out.println(e);
}
```

Exercise 1: Manage multiple circle objects by ArrayList.

Continue to the exercise above, we can use an array or an `ArrayList` to store circles.

In the main method, create an arrayList with a Circle type, to store many objects of Circle. Add the following code in main method.

```
ArrayList<Circle> circleList=new ArrayList<>();
Circle c1 = new Circle(1,2,3);
circleList.add(c1);
System.out.println(circleList.get(0));
```

Sample output:

```
Circle #1: radius = 1.00, x = 2.00, y = 3.00
```

Add the following code at the end of main method.

```
for (int i = 1; i < 5; i++) {
      Circle c = new Circle(i, Math.random() * 5, Math.random() * 5);
      circleList.add(c);
}
System.out.println("---Begin to print the circle list---");
for (Circle c: circleList) {
     System.out.println(c);
}
```

Sample output:

```
Circle #1: radius = 1.00, x = 2.00, y = 3.00
---Begin to print the circle list---
Circle #1: radius = 1.00, x = 2.00, y = 3.00
Circle #2: radius = 1.00, x = 0.54, y = 2.66
Circle #3: radius = 2.00, x = 1.29, y = 2.42
Circle #4: radius = 3.00, x = 4.46, y = 3.92
Circle #5: radius = 4.00, x = 4.00, y = 1.93
```