Tutorial of Composition and static field

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech
>
> Refer to the question and design idea provided by TAO Yida in 2023. Nov. 4th
>
> Designed by ZHU Yueming in 2023. Nov. 4th

Objective

- Learn to static class members
- Exercise Composition class
- Learn how to use package by intellijIDEA

Part 1: Static class member

Before Exercise:

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: Composition

Exercise 1 Circle and Rectangle:

Continue to the exercise, create a `Position` class, which has fields `double x` and `double y`, and an instance method `double distanceToOrigin()`
that returns the distance between the position and the origin point `(0.0, 0.0)`.

Modify the `Circle` class, replace `double x` and `double y` with `Position position`,
then make necessary changes.

Create a `Rectangle` class, which has the following fields:

```
private double width;
private double length;
private Position position;
```

Then create necessary constructors, getters, setters, and `toString()` methods.

Finally, create a `ShapeTest` class with a `main` method, which
creates `N` circles and `N` rectangles (`N` is a randomly generated integer).
The positions of these circles and rectangles should also be randomly generated.
Meanwhile, each shape should have a unique ID.

The `main` method should identify and print the shape (could be either a circle or a rectangle) that is furthest from the origin point (0,0).

Sample output1

```
Circle #1: radius = 2.62, position = (4.9,2.3)
Circle #2: radius = 1.89, position = (4.0,4.0)
Circle #3: radius = 1.55, position = (4.4,3.3)
Rectangle #1: width = 1.78 height = 1.67 position=(5.0,2.1)
Rectangle #2: width = 2.96 height = 2.74 position=(2.4,4.2)
Rectangle #3: width = 1.84 height = 1.72 position=(3.3,2.1)

Furthest Circle #2: radius = 1.89, position = (4.0,4.0)
```

Sample output2

```
Circle #1: radius = 2.72, position = (2.2,3.0)
Circle #2: radius = 2.75, position = (4.7,3.4)
Circle #3: radius = 1.31, position = (3.9,2.2)
Rectangle #1: width = 1.87 height = 1.63 position=(4.0,4.1)
Rectangle #2: width = 2.49 height = 1.01 position=(2.3,2.8)
Rectangle #3: width = 2.78 height = 2.60 position=(4.5,2.7)

Furthest Circle #2: radius = 2.75, position = (4.7,3.4)
```

Exercise 2 Person

Design a class named **Direction**, the objects of which represents directions:

- Private data fields:

  int **row**; // represents the direction of row

  int **col**; // represents the direction of col
- Design a **constructor** to initialize two private data fields:
- You can add other fields or methods that you think are necessary.

Design a class named **Preson**

- Private data fields:

  Direction[] **directions**;// represents the directions the preson can go

  int **i**;// represents the vertical position

  int **j**;// represents the horizontal position

  int **index**;// represents the index of current directions
- Design a **constructor** with three parameters to initialize three private data fields includes i, j and index. In constructor, initialize the `directions` fields with 8 directions objects as the table below:

Direction Name	row	col
Right	0	1
Right up	-1	1
Up	-1	0
Left up	-1	-1
Left	0	-1
Left down	1	-1
Down	1	0
Right down	1	1

- Design the **getter** and **setter** methods of those three fields: i, j and index.
- Design a method **changeDirection()** to increase the index of current direction to the next direction.

```
Person p = new Person(0,0,6);
System.out.println(p.getIndex());
p.changeDirection();
System.out.println(p.getIndex());
p.changeDirection();
System.out.println(p.getIndex());
```

result:

```
6
7
0
```

- Design a method **walk(int step)** with an int parameter step, which indicates the preson talk steps in current direction, after that the i and j would be changed accordingly.
- Design a **toString()** method, which returns String describe the current location of the preson, like:

```
(1,1)
```

Sample Main method:

```
public static void main(String[] args) {
        Person p = new Person(0,-1,0);
        p.walk(3);
        p.changeDirection();
        System.out.println(p);
        p.walk(2);
        p.changeDirection();
        System.out.println(p);
        p.walk(5);
        p.changeDirection();
        System.out.println(p);
    }
```

Sample output:

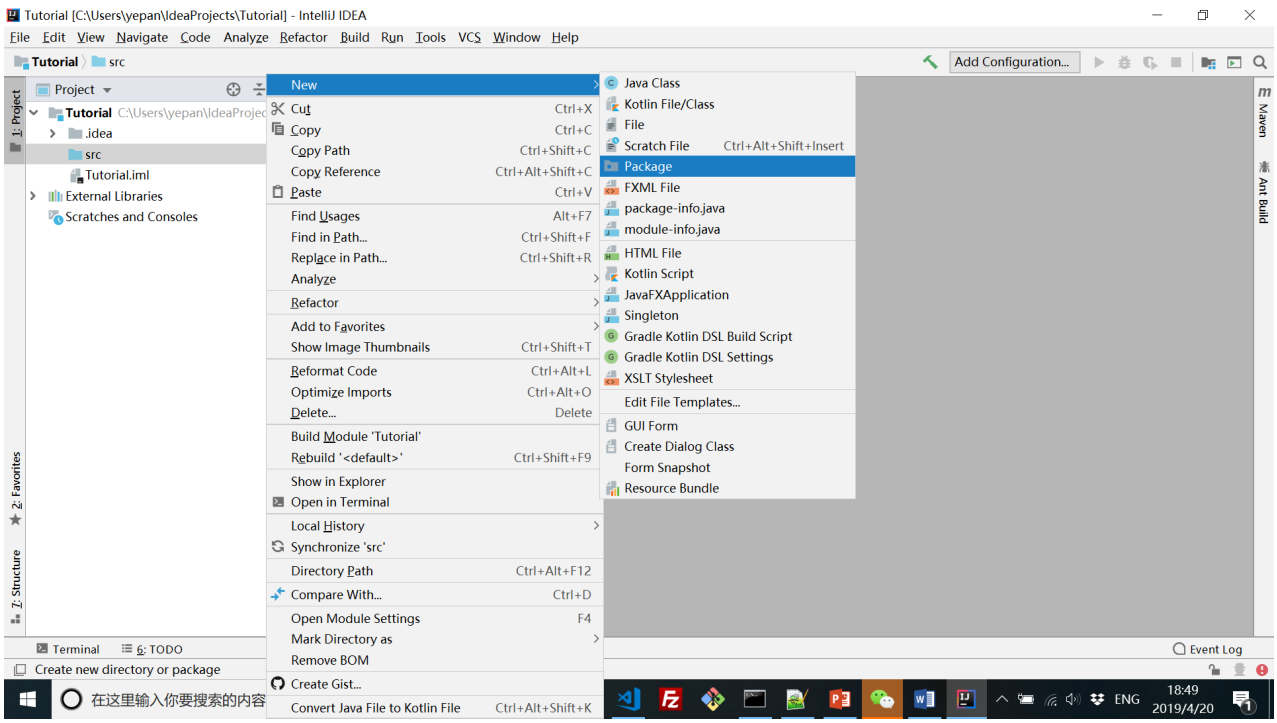
```
(0,2)
(-2,4)
(-7,4)
```

Part 3: Package

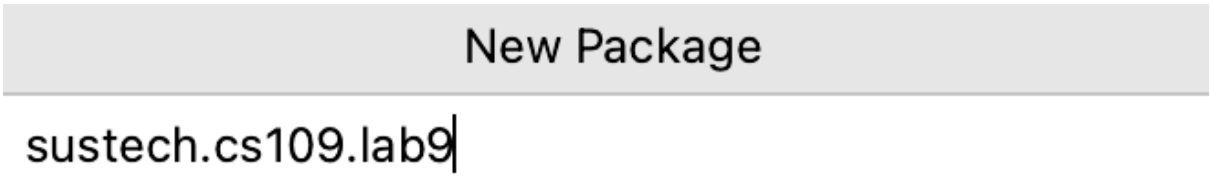
1. By Intellij IDEA

The above tutorial explains package and classpath at a low level. In an IDE, creating packages and declaring classes in them is as easy as pie. We provide the steps below for IntelliJ IDEA.

Step 1: In a project, right click on the src, then click New->Package

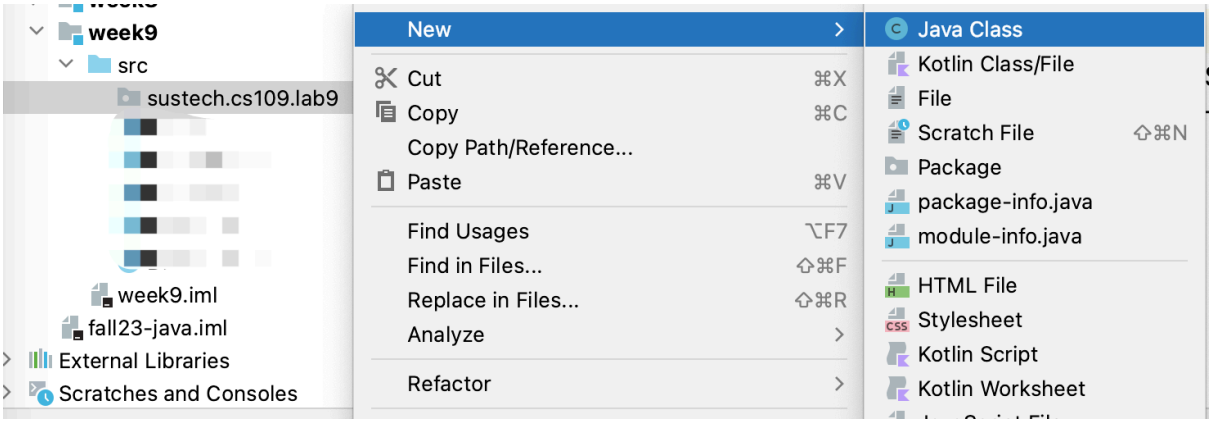


- Enter the package name in the dialog box, click ok and you will see the package created.

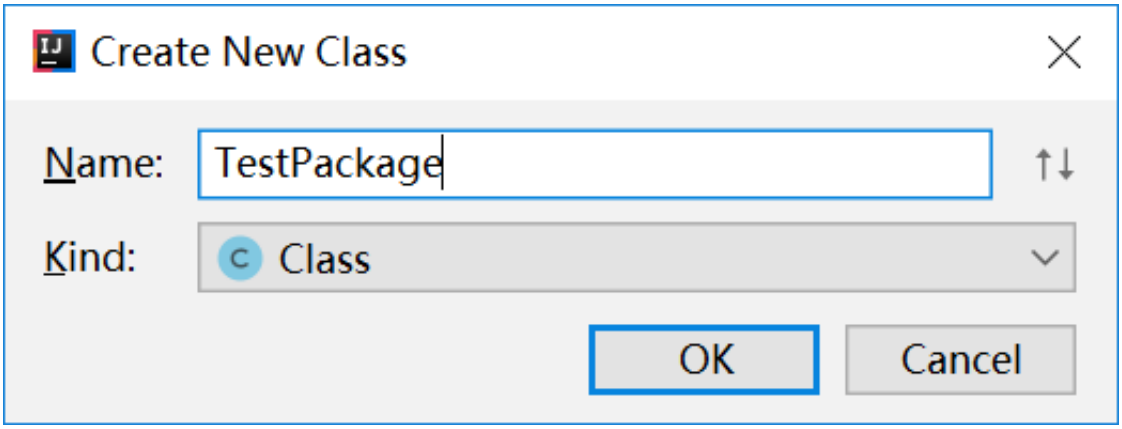


(correction: please type “sustech.cs109.lab9” here)

- Right click on the package, then click New->Java Class



- Enter the class name in the dialog box, click ok and you will see the skeleton code of the newly created class. You will find that the package declaration statement is added automatically.

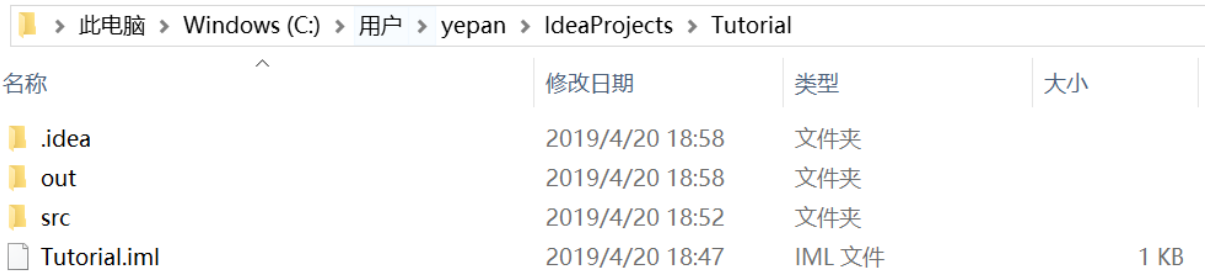


The generate class would be:

```
package sustech.cs109.lab9;

public class TestPackage {
}
```

Now compile the class and go to the directory where the project is stored. You will see that project directory contains two sub-directories: `src` stores the .java source files and “out” stores the .class files after compilation.



If you check the `src` and out directories, you will find that the `TestPackage.java` and `TestPackage.class` files reside in the following directories, respectively:

```
src/sustech/cs109/lab9
out/production/[Project Name]/sustech/cs109/lab9
```

IDEA helps manage all the stuff automatically. The way how source files and class files are organized may be different for other IDEs (for example, Eclipse), but the `TestPackage.class` file is always put under `sustech/cs109/lab9` after compilation.