Tutorial of Class and Object

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech
>
> Designed by ZHU Yueming
>
> Improved by WANG Wei
>
> Modified (mainly change to markdown file) by ZHU Yueming in 2021. March. 29th
>
> Improved Exercise 1 and add part 2 by ZHU Yueming in 2023. Oct. 22th

Experimental Objective

- Learn how to define a Java class and create its object
- Learn how to define and use instance variables
- Learn how to define and use instance methods
- Learn how to use get and set methods
- Learn to declare constructors and use them to construct objects

Part1 Class and Object

Before Exercise

Attribute and Method

Step 1: How to define a circle on 2 dimensional plane?

A circle has three attributes including the **radius**, the **x** *coordinate* and the **y** *coordinate*.

We can define a class named `Circle`, in which there are three private attributes.

```
public class Circle {
  private double radius;
  private double x;
  private double y;
}
```

Step 2: Define the methods of a circle.

Define three public methods for computing the area, perimeter and print position of the circle.

```
public class Circle {
  private double radius;
  private double x;
  private double y;

  public double area() {
```

```
    return radius*radius*Math.PI;
  }
  public double perimeter () {
    return 2*Math.PI*radius;
  }
  public void position() {
    System.out.printf("Position of the cricle is (%.1f,%.1f)\n",x,y);
  }
}
```

Step 3: How to use the class Circle?

Create another class named `CircleTest` in the same package, in which there is a main method to be used.

In the main method, we can create an object of `Circle` by using the statement as follows:

```
Circle c1=new Circle();
```

After that, we want to know the perimeter, area and position about the `c1`, so we need to invoke the method of `c1`.

```
public class CircleTest {
  public static void main(String[] args) {
    Circle c1=new Circle();
    System.out.printf("The area of c1 is %.2f\n", c1.area());
    System.out.printf("The perimeter of c1 is %.2f\n", c1.perimeter());
    c1.position();
  }
}
```

When we run the program, the result would as follows:

```
The area of c1 is 0.00
The perimeter of c1 is 0.00
Position of the circle is (0.0,0.0)
```

Getter and Setter

Step 4: Set and get the values of the attributes

If we set or get the radius of a circle object in main method directly, it would lead to an error because of its private privilege.

In addition, the radius of a circle should not contain a negative number, how can we set the restriction?

```
public static void main(String[] args) {
    Circle c1=new Circle();
    System.out.printf("The area of c1 is %.2f\n", c1.area());
    System.out.printf("The perimeter of c1 is %.2f\n", c1.perimeter());
    c1.position();
      c1.radius=-1;
      System.out.println(c1.radius);
  }
```

We can define several public methods in class Circle for getting or setting the class variables, and we can check the validity of input value in the set method.

```
public class Circle {
  private double radius;
  private double x;
  private double y;

  public double area() {
    return radius*radius*Math.PI;
  }
  public double perimeter () {
    return 2*Math.PI*radius;
  }
  public void position() {
    System.out.printf("Position of the cricle is (%.1f,%.1f)\n",x,y);
  }
  public double getRadius() {
    return radius;
  }
  public void setRadius(double radius) {
    if (radius > 0) {
      this.radius = radius;
    }
  }
  public double getX() {
    return x;
  }
  public void setX(double x) {
    this.x = x;
  }
  public double getY() {
    return y;
  }
  public void setY(double y) {
    this.y = y;
  }
}
```

After that, we can access the attributes by the get and set methods.

```
public static void main(String[] args) {
    Circle c1=new Circle();

    c1.setRadius(5);
    System.out.println(c1.getRadius());

    System.out.printf("The area of c1 is %.2f\n", c1.area());
    System.out.printf("The perimeter of c1 is %.2f\n", c1.perimeter());
    c1.position();

  }
```

Sample output:

```
5.0
The area of c1 is 78.54
The perimeter of c1 is 31.42
Position of the circle is (0.0,0.0)
```

Exercise

Exercise 1 : User

Declare a class named **User**. The class contains:

- Private data fields:

  String **account;**

  String password;

  double money;

- Implement a public method named **introduce()** to print the user account and his account balance.

  Output:

  ```
  [account]'s account has a balance of [money] dollar
  ```

- Implement a public method **expense(double value,Scanner in)**.
  - **Check whether suffice the funds**.

    If money is not enough, output:

    ```
    Plan to expense [value] dollar but no sufficient funds
    ```

    If money is enough, output:

```
Plan to expense [value] dollar
Please input your password:
```

- **Input password**, and check password in three times, if the password is wrong, terminate the program.

  For example:

  ```
  password error,there are 2 times left to try
  ```

- **Expense**:

  output:

  ```
  Expense [value] dollar and balance [updated money] dollar
  ```

- Implement a public method **income(double value).** It deposits the money to the user account.

  Output:

  ```
  Got [value] as income,balance is [updated money] dollar
  ```

- Implement the **getter** and **setter** methods for each private field of the class User.

In the same package, we create a class named `UserTest`, which has a main method.

```
User user =new User();
Scanner in = new Scanner(System.in);
user.setUser("Lucy");
user.setPassword("123456");
user.setMoney(1000);
user.introduce();
user.expense(2000,in);
user.expense(500,in);
user.income(1000);
user.introduce();
in.close();
```

Sample Output 1:

```
Lucy's account has a balance of 1000.00 dollar
Plan to expense 2000.00 dollar but no sufficient funds
Plan to expense 500.00 dollar
Please input your password:
12345
password error,there are 2 times left to try
Please input your password:
```

```
12345
password error,there are 1 times left to try
Please input your password:
12345
password error,there are 0 times left to try
password error,expense failed
Got 1000.00 as income,balance is 2000.00 dollar
Lucy's account has a balance of 2000.00 dollar
```

Sample Output 2:

```
Lucy's account has a balance of 1000.00 dollar
Plan to expense 2000.00 dollar but no sufficient funds
Plan to expense 500.00 dollar
Please input your password:
123456
Expense 500.00 dollar and balance 500.00 dollar
Got 1000.00 as income,balance is 1500.00 dollar
Lucy's account has a balance of 1500.00 dollar
```

Part 2: Constructors

Before Exercise

The Circle class defined in the previous exercise does not contain any explicitly declared constructor. The Java compiler will provide a default constructor that would initialize all three fields (radius, x, y) to 0.0 when called. If we want to create a circle object, of which the three fields have the following values: radius = 2.0, x = 1.0, y = 1.0, we can write a main method like the one below.

```
public static void main(String[] args) {
   Circle c = new Circle();
   c.setRadius(2.0);
   c.setX(1.0);
   c.setY(1.0);
}
```

However, this is quite troublesome. A better solution is to declare constructors so that they can be called to construct objects with certain radius values and center positions. The following code declares two such constructors. The first constructor takes one argument to initialize the radius field (the fields x and y will be initialized to 0.0). The second constructor takes three arguments to initialize all three fields.

```
public Circle(double radius) {
    this.radius = radius;
}

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

Note that in the constructors, “this” keyword is needed to differentiate the field access from method argument access. Now we can simply create the circle (radius = 2.0, x = 1.0, y = 1.0) with a constructor call. Much easier, right?

```
public static void main(String[] args) {
   Circle c = new Circle(2.0, 1.0, 1.0);
}
```

Continue to type the following code in the main method and see what happens.

```
Circle c = new Circle();
```

The code would not compile. Do you know why?

Exercise:

Exercise1: Circle

Add a public method distanceToOrigin() to the Circle class. The method returns the distance between the circle’s center point and the origin point `(0.0, 0.0)`. Then write a Java program to perform the following tasks:

1. Generate a random number N in the range `[5, 10)`.
2. Create N circles. Each circle has a random radius in the range [1.0, 3.0) and a random center position: x and y are in the range `[2.0, 5.0)`.
3. Among the generated circles, find the one with the smallest area and the one whose center is the farthest from the origin point.

For random number generation, you may use the following two methods of the Random class:

- `public int nextInt(int bound)`
- `public double nextDouble()`

See https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Random.html for more details.

A sample run:

```
Circle #1: radius = 1.49, x = 4.01, y = 3.55
Circle #2: radius = 1.58, x = 3.92, y = 3.77
Circle #3: radius = 1.81, x = 2.89, y = 3.60
Circle #4: radius = 2.47, x = 3.90, y = 3.87
Circle #5: radius = 1.34, x = 2.94, y = 4.05
Circle #5 is the smallest circle, area = 5.62
Circle #4 is the farthest circle, distance to origin = 5.50
```

Exercise 2 : Food

Design a class named **Food**. The class contains:

- Private data fields:

  int **id;**

  String **name;**

  String **type**;

  int **size**;

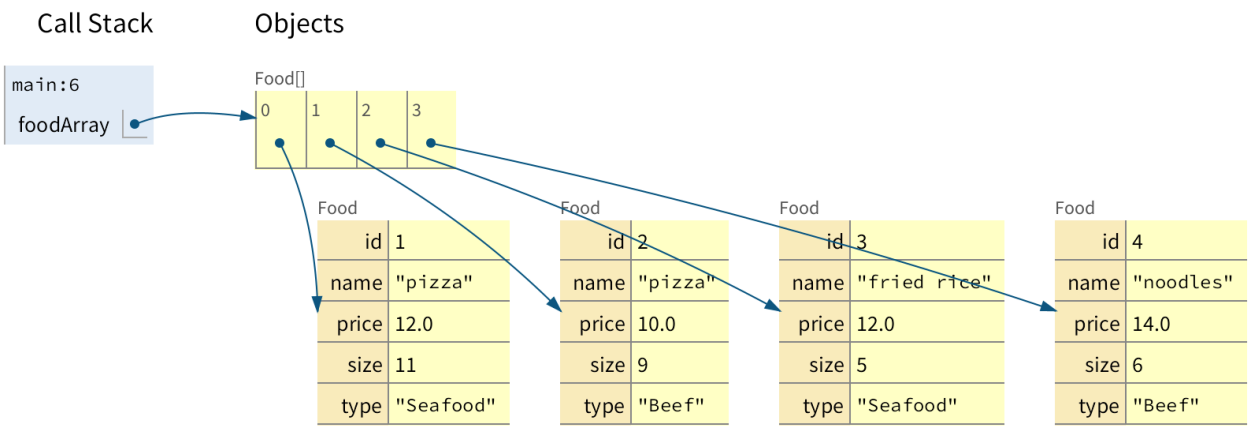
  double **price;**
- Design a 5 parameter **constructor** to build the object of Food.
- Implement a public method named `getMenu()` to print all the information of this food object.
- Implement the **getter** and **setter** method for each private field of Food.

In `FoodTest` class, create four objects of Food as follows:

Object Name	id	name	type	size	price
pizza1	1	pizza	Seafood	11	12
pizza2	2	pizza	Beef	9	10
Fried rice	3	fried rice	Seafood	5	12
Noodles	4	noodles	Beef	6	14

Create an `Food[]` to add those four Food objects, and then show the information of them as follows by iterating the `Food[]` we created.

```
Seafood pizza: (11 Inches) 12.00 $
Beef pizza: (9 Inches) 10.00 $
Seafood fried rice: (5 Inches) 12.00 $
Beef noodles: (6 Inches) 14.00 $
```


Call Stack
Objects
main:6
foodArray
Food[]
0
1
2
3
Food
id 1
name "pizza"
price 12.0
size 11
type "Seafood"
Food
id 2
name "pizza"
price 10.0
size 9
type "Beef"
Food
id 3
name "fried rice"
price 12.0
size 5
type "Seafood"
Food
id 4
name "noodles"
price 14.0
size 6
type "Beef"