Tutorial of Constructor and String

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech
>
> Modified (only change to markdown file) by ZHU Yueming in 2021. April. 6th
>
> Update by ZHU Yueming in 2021. Nov. 1st. Add File read and write demo

Objective

- Learn to declare constructors and use them to construct objects
- Learn to declare and use of static data field and `toString( )` method.
- Learn to use various String methods
- Learn to use StringBuilder
- Exercise Composition class

Part 1: Constructors and instance methods

The Circle class defined in the previous lab 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.

```java
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 1 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:

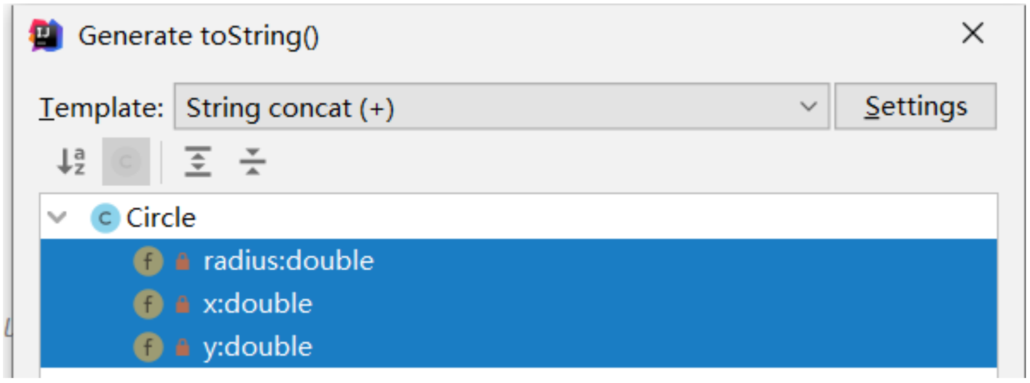
```
Circle #1: radius = 1.49, x = 4.01, y = 3.55
Circle #2: radius = 1.58, x = 3.92, y = 3.77
Circle #3: radius = 1.81, x = 2.89, y = 3.60
Circle #4: radius = 2.47, x = 3.90, y = 3.87
Circle #5: radius = 1.34, x = 2.94, y = 4.05
Circle #5 is the smallest circle, area = 5.62
Circle #4 is the farthest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

Part 2: toString( ) method

A special instance method **`toString( )`** which return a string related to the current object.

While the object is used as a string, the `toString( )` is invoked by default. For example:

- Create a Circle object c : `Circle c = new Circle(1.0, 1.0, 1.0);`
- Use `System.out.print(c)` to print the c, what's the output?
- Use `System.out.print(c.toString())` instead of `System.out.print(c)`, what's the output?
- Use IDEA to generated the `toString()` of the current class.


Generate toString()

Template: String concat (+)

Settings

Circle
- radius:double
- x:double
- y:double

The following instance method `toString()` is generated, all the instance data field is added into the String:

```java
public String toString() {
      return "Circle{" +
            "radius=" + radius +
            ", x=" + x +
            ", y=" + y +
            '}';
       }
```

- Invoke `System.out.print(c)` and `System.out.print(c.toString())` again, what's the output?
- While change the `toString( )` as follow code, invoke `System.out.print(c)`. what's the output, why?

```
public String toString(char z) {
      return "Circle{" +
            "radius=" + radius +
            ", x=" + x +
            ", y=" + y +
      ", z=" + z +
            '}';
       }
```

Exercise 1： Food

Continue to design te class Food.

The class contains:

- Private data fields:

  int **id**;

  String **name;**

  String **type**;

  int **size**;

  double **price;**

- Implement the **getter** and **setter** method for each private field of Food.
- Implement a five arguments **constructor** to initialize the five private data fields.
- Impement a **toString()** method to return a String, which contains all the information of this food object like:

```
Seafood pizza: (11 Inches) 120.00 $
```

In `FoodTest` class, create four objects of Food by using its **constructor** 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 print its **toString()** method of them as follows by iterating the `Food[]` we created.

```
Seafood pizza: (11 Inches) 120.00 $
Beef pizza: (9 Inches) 100.00 $
Seafood fried rice: (5 Inches) 40.00 $
Beef noodle: (6 Inches) 35.00 $
```

###

Part 3: String manipulations

Please use String methods

(https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) to finish the tasks below. Methods in the Character class are also helpful:

https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html.

Exercise 1

Write a program to check if a string provided by a user is a palindrome or not. A string is a palindrome is the reverse of the string is the same as the string (we do not differentiate upper-case and lower-case letters in this task). For example, “abba” is a palindrome. “#Aa#” and “0” are also palindromes.

Your program should continuously take user inputs for checking and stops when the user types “quit”.

String Library in this question:

```
public char charAt(int index)
public boolean equals(Object anObject)
```

Character Library in this question:

```
public static char toLowerCase(char ch)
```

A sample run:

```
Type a string ("quit" to finish): hello
hello is not a palindrome
Type a string ("quit" to finish): many
many is not a palindrome
Type a string ("quit" to finish): 0
0 is a palindrome
Type a string ("quit" to finish): 900
900 is not a palindrome
Type a string ("quit" to finish): #Aa#
#Aa# is a palindrome
Type a string ("quit" to finish): quit

Process finished with exit code 0
```

Exercise 2

Write a program to remove all repeated characters in a string provided by the user and return a new string without any repeating characters or white spaces. Please use `StringBuilder` to build the new string.

String Library in this question:

```
public char charAt(int index)
```

Character Library in this question:

```
public static boolean isWhitespace(char ch)
```

Sample runs:

```
Please type a string: hello
After removing repeating chars and spaces: helo
```

```
Please type a string:
Empty string, exit…
```

```
Please type a string: abcd  bcde cdef
After removing repeating chars and spaces: abcdef
```

Exercise 3

Write a program to count the occurrence of a substring in a string. The program should ask the user to input two strings s1 and s2, and output the number of occurrences of s2 in s1.

String Library in this question:

```
public int indexOf(String str)
public int indexOf(String str, int fromIndex)
```

Sample code:

```
String s1 = "JavaExamplesJavaCodeJavaProgram";
String s2 = "Java";
System.out.println(s1.indexOf(s2));
System.out.println(s1.indexOf(s2, 1));
System.out.println(s1.indexOf(s2, 13));
System.out.println(s1.indexOf(s2, 21));
```

Sample runs:

```
s1: JavaExamplesJavaCodeJavaProgram
s2: Java
Found at index: 0
Found at index: 12
Found at index: 20
Total occurrences: 3

s1: abcd  bcde cdef
s2: bc
Found at index: 1
Found at index: 6
Total occurrences: 2

s1: abcdefg
s2: xyz
Total occurrences: 0
```

Exercise 4

Input a number n, which means there are n lines, then write a program to print strings in lines as follows. And then improve your program to be more efficient.

```
please input number n:10
a
ab
abc
abcd
abcde
abcdef
abcdefg
abcdefgh
abcdefghi
abcdefghij
```

The following code can resolve the problem, but it is a time-consuming way. Using `StringBuilder` to improve it.

```
Scanner in = new Scanner(System.in);
System.out.print("please input number n:");
int n = in.nextInt();
String str = "";
for (int i = 0; i < n; i++) {
     str = str + (char) (97 + i);// str = new String("abc"+"d"); time-consuming
     System.out.println(str);
}
```

Hint:

```
StringBuilder sb = new StringBuilder();
sb.append('a');
```

API References:

String methods:

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

```
 public int length()

 public char charAt(int index)

 public boolean startsWith(String prefix)

 public boolean equals(Object anObject)

 public boolean equalsIgnoreCase(String anotherString)
```

```
  public String trim()

  public int indexOf(String str)

  public int indexOf(String str, int fromIndex)

  public String substring(int beginIndex)

  public String substring(int beginIndex, int endIndex)

  public String[] split(String regex)

  public char[] toCharArray()
```

Character methods:

https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html

```
public static char toLowerCase(char ch)

public static boolean isWhitespace(char ch)
```

StringBuilder methods:

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

```
public StringBuilder append(char c)

public String reverse()

public String toString()
```

Part 4: 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)
```