

Introduction to Computer Programming (Java A)

Lab 8

[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

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.

```
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? If not, please go check our lecture notes.

[Exercises]

Exercise 1: 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:

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

See <https://docs.oracle.com/javase/8/docs/api/java/util/Random.html> for more details.

A sample run:

```
Circle #1: radius = 2.14, x = 2.00, y = 4.04
Circle #2: radius = 1.45, x = 3.33, y = 4.16
Circle #3: radius = 1.89, x = 4.68, y = 4.87
Circle #4: radius = 1.59, x = 4.97, y = 4.47
Circle #5: radius = 2.71, x = 2.66, y = 4.83
Circle #6: radius = 2.42, x = 2.18, y = 3.34
Circle #7: radius = 2.72, x = 4.60, y = 2.46
Circle #2 is the smallest circle, area = 6.62
Circle #3 is the farthest circle, distance to origin = 6.75
```

Part 2: static data field and instance method `toString()`

1. Static data field belongs to class instead of object.

```
public class Circle {
    private double radius;
    private double x;
    private double y;
    private static int cnt = 0;
}
```

`cnt` is the static data field, it got a initial value 0.

In the class static method could access a static data field.

```
public static int getCnt(){
    return cnt;
}
```

How about constructor ?

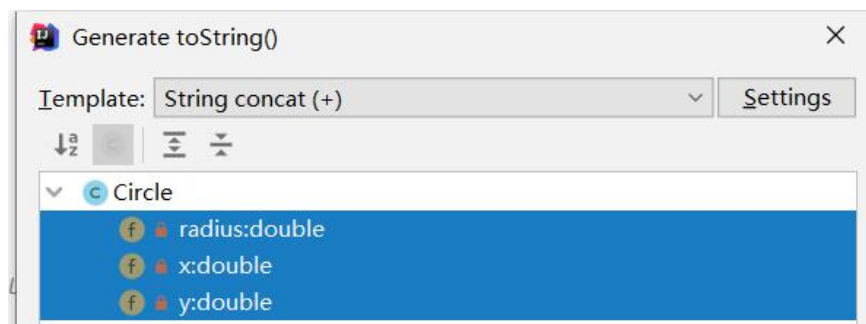
```
public Circle(double radius) {
    this.radius = radius;
    cnt++;
}
```

Instance data field belongs to object while static data field belongs to class, static data field is shared with all the objects of the class.

2. 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:

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



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

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

- 5) Invoke `"System.out.print(c)"` and `"System.out.print(c.toString())"` again, what's the output?
- 6) 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 2: Modify the program in Exercise 1, add a new instance field `id` and a public method `toString()` in class `Circle`. Modify Exercise 1 by using `id` and `toString()` of `Circle` while keep the same function.

1. The instance data field “`id`” is the integer number according to the generated sequence of the `Circle` object. For example, the 1st created `Circle` object’s `id` is 1, the 2nd created `Circle` object’s `id` is 2, and so on.
2. The instance method “`toString()`” returns a `String` which include all the data field value as desired format.

```
Circle #idValue: radius = radiuValue, x = xValue, y = yValue
```

Notes: `idVaule` using `%d`, `radiuValue`, `xVaulue` and `yValue` using `%.2f` as format description.

For example:

```
Circle #1: radius = 1.71, x = 4.84, y = 4.46
```

A sample run:

```
Circle #1: radius = 2.15, x = 2.48, y = 4.71
Circle #2: radius = 2.52, x = 4.40, y = 2.14
Circle #3: radius = 2.24, x = 2.72, y = 2.61
Circle #4: radius = 2.87, x = 3.53, y = 4.68
Circle #5: radius = 1.45, x = 2.43, y = 2.56
Circle #6: radius = 1.15, x = 2.60, y = 2.90
Circle #6 is the smallest circle, area = 4.13
Circle #4 is the farthest circle, distance to origin = 5.86
```

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 3: Write a program to check if a string provided by a user is a palindrome or not. A string is a palindrome if 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`”.

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 4: 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.

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 5: 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`.

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
--

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 toString()