Tutorial of Constructors 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
>
> Minor changes by Yida Tao, Oct.28 2022

Objectives

- Learn to declare constructors and use them to construct objects
- Learn to declare and use static data field and `toString( )` method.
- Learn to use various `String` methods
- Learn to read file and write file.

Part 1: Constructors

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 use `setter` methods in the `main` method like 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.

```java
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?

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 generate the `toString()` of the current class: in `Circle.java`, right click -> Generate -> `toString()`.

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 +
            '}';
       }
```

- Invoke `System.out.print(c)` and `System.out.print(c.toString())` again, what's the output?
- If we change the `toString( )` to the following code and invoke `System.out.print(c)`, what's the output and why?

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

Part 3: Static data fields

Add a static field `cnt` with an initial value 0.

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

In each constructor, add `cnt++`, i.e., whenever a new circle instance is constructed, `cnt` increases by 1.

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

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

Run the `main` method, what's the output and why?

```
public class CircleTest {
    public static void main(String[] args) {
        for(int i=0;i<5;i++) {
            Circle c1 = new Circle(i);
            Circle c2 = new Circle(i, Math.random()*5, Math.random()*5);
            System.out.println(Circle.cnt);
        }
    }
}
```

Basically, instance data field belongs to object while static data field belongs to class, meaning that static data field is **shared** with all the objects of the class.

Exercise 1

Add a new **instance** field `id` in class `Circle`. This field is the integer number according to the generated sequence of the `Circle` object. For example, the first created Circle object's id is 1,the second created Circle object's id is 2, and so on.

Next:

- Modify the constructors so that `id` could also be initialized when a new instance is created.
- Modify the instance method `toString()` to return a String which includes all the data field value as the desired format. That is, when calling `System.out.print(c)` on a `Circle` instance `c`, it should print out in a format like `Circle #2: radius = 2.91, x = 2.89, y = 2.92` (where `2` in `#2` is the id value).
- Add a public method `double distanceToOrigin()` to the `Circle` class. This method returns the distance between the circle's center point and the origin point `(0.0, 0.0)`.
- The incomplete `main` method below generates n circles, where n is in the range `[5, 10)`. Each circle has a random radius in the range `[1.0, 3.0)` and a random center position: x and y are both in the range `[2.0, 5.0)`.

```
public class Lab8E1 {
    public static void main(String[] args) {
        Random random = new Random();
        int n = random.nextInt(5) + 5;

        for (int i = 0; i < n; i++) {
            double r = random.nextDouble() * 2 + 1.0;
            double x = random.nextDouble() * 3 + 2.0;
            double y = random.nextDouble() * 3 + 2.0;
            Circle c = new Circle(i+1, r, x, y);
            System.out.println(c);

        }

    }
}
```

Your task is to update the `main` method, so that it finds the circle with the smallest area and the one whose center is the furthest from the origin point.

Sample output of executing `main`:

```
Circle #1: radius = 1.07, x = 2.56, y = 3.49
Circle #2: radius = 1.38, x = 4.80, y = 2.80
Circle #3: radius = 1.37, x = 3.89, y = 4.82
Circle #4: radius = 1.70, x = 4.56, y = 2.17
Circle #5: radius = 1.44, x = 2.21, y = 4.28
Circle #6: radius = 2.00, x = 2.51, y = 2.68
Circle #7: radius = 1.10, x = 3.09, y = 3.73
Circle #8: radius = 2.43, x = 3.19, y = 2.81
Smallest circle: Circle #1: radius = 1.07, x = 2.56, y = 3.49
Furthest circle: Circle #3: radius = 1.37, x = 3.89, y = 4.82
```

Part 4: String manipulations

Please use String methods to finish the tasks below. Methods in the `Character` class are also helpful.

Exercise 2

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”.

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 3

Write a program to remove all the repeated characters in a string provided by the user and return a new string without any repeating characters or white spaces. You may 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 4

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()
```