Object-Orientated Programming

CMM024

RGU (2023)

Session 3 (Procedural Programming)

Introduction to:

- Using Standard Java methods
- Create your own methods
- How to use your own methods

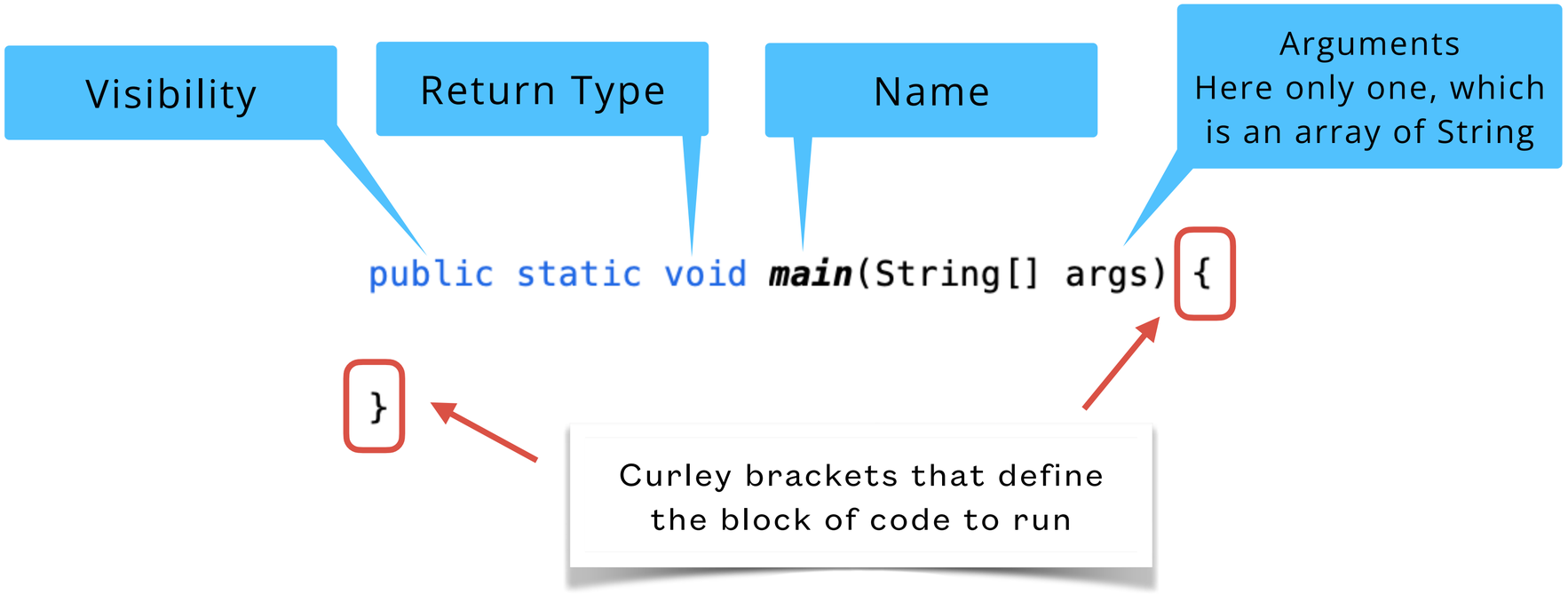
CMM024

Summary

- **A few facts about methods**
- **Why creating methods**
- **How to create procedural methods**
- **Coding Example**
- **How to invoke your own methods**
- **Next Session**

Methods, what are they?

- A method is a block of code that runs when it is invoked (i.e. called). It can have some inputs known as parameters (if more than one parameter, a coma separates them), and it returns a value of a certain type. In addition to this, there is the visibility which tells if that method can be seen by other objects or not.
- You have already used a method on many occasions



Methods: Examples seen already

- Let's start with an examples we have already used.

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter your name");

String name = sc.next();
```

```
Scanner input = new Scanner(System.in);
System.out.println(x:"Enter a number");

int times = input.nextInt();
```

Method is called next and its return type is a string

Method is called nextInt and its return type is and integer

THERE ARE IN THE SCANNER PACKAGES

Methods: Examples not yet seen already

- Let's start with an examples we have already used. Create Lect3Randoms.java file in the Lecture3 folder.

```
random = 0.11989367504708237
```

```
public static void main(String[] args) {
      double randValue = Math.random();
      System.out.println("\nrandom = " + randValue + "\n");
  }
```

Method is called random and its return type is a double. It returns a random value from 0 to 1, excluding 1

It is a method of the class called Math, which is part of the java.lang package

```
random = 0.8377685207975534
```

```
public static void main(String[] args) {
        Random randomiser = new Random();
        double randValue = randomiser.nextDouble();
        System.out.println("\nrandom = " + randValue + "\n");
    }
```

Method is called nextDouble and its return type is a double. It returns a double random value from 0 to 1, excluding 1

It is a method of the class called Random, which is part of the java.util package

Note: classes in the java.util package needs to be imported but class in the java.lang do not!

LET'S CODE

Methods: Examples not yet seen already

- We can get randoms within a wanted range. Let's say from 10 to 15 as an example. Create a new Lect3RandRange.java file. How we do this:
  - Determine the min value I.e. 10
  - Determine the max value I.e. 15
  - Determine the range of number to be generated = (max - min) = 5 The range become [0 , 5]
  - Shift the random value by the min. The range become [10+0 , 10+5] =. [10,15] (this is what we wanted!)

```
public static void main(String[] ags) {
    double min = 10.0; double max = 15.0;
    double range = max - min;
    Random randomiser = new Random();
    double randValue = randomiser.nextDouble(range) + min;
    System.out.println("Range = " + range + " Value = " + randValue);
}
```

```
Range = 5.0 Value = 10.737549802264638
```

Note: nextRandom method has two forms. One without parameter (returning 0 to 1 values) and one with which allows you to enter the range (returning value with the range)

LET'S CODE

Methods: creating your own method

- Lets says we want to repeat the process many times. This is a lot of code to enter each time we want to generate a random value. We can create our own method whose main job is to compute a random value and return it. Create a new Lect3Rand2.java file and copy the main you did earlier.

```
import java.util.Random;

public class Lect3Rand2 {

    public static double getRandValue(double min, double max){

        return 0.0;
    }

      public static void main(String[] args) {
        double min = 10.0; double max = 15.0;
        double range = max - min;
        Random randomiser = new Random();
        double randValue = randomiser.nextDouble(range) + min;
        System.out.println("Range = " + range + " Value = " + randValue);
      }


}
```

Note: To compute the random value we only need the minimum and maximum values

Note: the new method return 0.0 as we have not yet computed the random value to return

Note: Don't worry about the static right now!

LET'S CODE

Methods: creating your own method

- We can now code the method we just created

```
public static double getRandValue(double min, double max) {
   double range = max - min;
   Random randomiser = new Random();
   double randValue = randomiser.nextDouble(range) + min;
   return randValue;
}
```

- We can now re-code the main method to generate easily 5 random values and display them.

```
public static void main(String[] args) {
   int randNumber = 5;
   for (int i = 0; i < randNumber; i++) {
       double randValue = getRandValue(10, 15);
       System.out.println("Value " + i +": " +  randValue);
   }
}
```

LET'S CODE

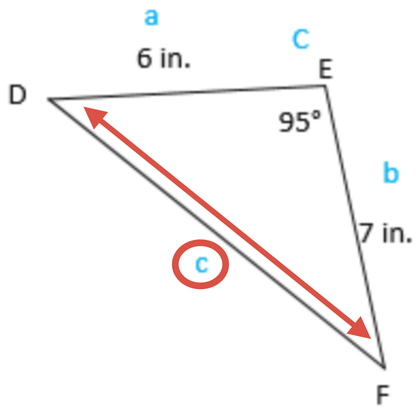
Creating methods: Pythagoras example

There is a purpose to create a method. I.e. a method accomplishes a task. Create a new FindSide.java file

You can create your own one. Here, given two side lengths and the angle between them, find the length of the third side.

If we want to compute two solutions, we need to duplicate the code below twice!

This method is using other methods from the Math class. I.e. sqrt & cos (PI is a constant in that class)




```
public class FindSide {

    public static void main(String[] args) {

        double angle = 95.3;
        double side1 = 6.2;
        double side2 = 7.5;

        // Note that Math.cos works with radians ( radians = degrees * pi / 180)
        double radians = angle * Math.PI / 180;
        double side3 = (side1 * side1) + (side2 * side2) - (2 * side1 * side2 * Math.cos(radians));
        side3 = Math.sqrt(side3);
        System.out.printf("Side1 = %.1f Side 2 = %.1f Angle = %.1f Side 3 = %.1f \n",
                side1, side2, angle, side3);
    }

}
```

Formula

$$c^2 = a^2 + b^2 - 2ab\cos C$$

LET'S CODE

Creating methods: Pythagoras example

A better way is to create your own method to compute the result

```
public class FindSide {


    public static double computeSide3(double angle, double side1, double side2){
        // Note that Math.cos works with radians ( radians = degrees * pi / 180)
        double radians = angle * Math.PI / 180;
        double side3 = (side1 * side1) + (side2 * side2) - (2 * side1 * side2 * Math.cos(radians));
        side3 = Math.sqrt(side3);
        return side3;
    }

    public static void main(String[] args) {
        double angle = 95.3;
        double side1 = 6.2;
        double side2 = 7.5;
        double side3 = computeSide3(angle, side1,side2);)
        System.out.printf("Side1 = %.1f Side 2 = %.1f Angle = %.1f Side 3 = %.1f \n",
                side1, side2, angle, side3);
    }
}
```

LET'S CODE

Notes on methods

You will often hear about a method signature. What is a method signature?

```
public static double computeSide3(double side1, double side2, double angle);
```

A method signature includes

- its visibility I.e. public (it can be private or protected as we will see later one)
- Its return type I.e. double (but it can return nothing i.e. void)
- Its name I.e. computeSide3
- Its parameters I.e. side1, side2 & angle, including their types

Notes:

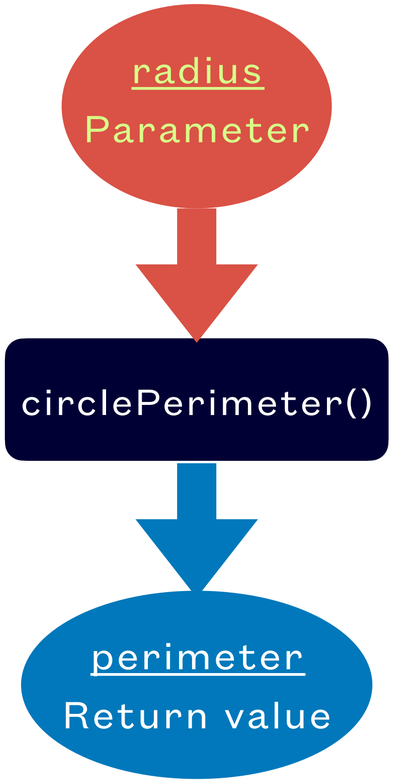
- You cannot have 2 methods with same signature in the same class
- Methods can have local variables I.e. other methods cannot see them!
- A method name MUST start with a lower case latter

Circle perimeter (practice)

$$C = 2\pi r$$

Let's suppose we want to create a method dedicated to compute the circumference of a circle.

To calculate the circumference of a circle C ($C = 2\pi r$) you must know the radius of that circle, and you must know the value of π.



What could be the inputs?

Once you know the radius you can compute the perimeter using the formula above. Therefore only one input for the radius as we can obtain π at any time using the Math.PI constant

What would be the type of this input?

The radius input can have many types depending what you want. If you want whole numbers (i.e. 5, 6,23) then the parameter type can be an integer, otherwise if you want decimal numbers (5.4, 6.78, 23.48) then you should have a double for the precision (or a float)

What would be the return type?

If the input is a double, then you want a precise number like a double, otherwise if the input was a integer, then the return type should be an integer

Example: Circle perimeter (practice)

(1) Create a file called CirclePerimeter.java

(2) Create a main function

(3) Type the method below

$$C = 2\pi r$$

Type the code below

```
public class CirclePerimeter {

    public static void main(String[] args){

    }
    // Using local variables
    public static double circlePerimeter(double radius) {
        double pi = Math.PI;
        double perimeter = 2 * pi * radius;
        return perimeter;
    }
}
```

```
Radius = 4.5 Perimeter = 28.274333882308138
```

Complete the code below

```
public static void main(String[] args){
   double radius = 4.50;
   double perimeter = circlePerimeter(radius);
   System.out.println("Radius = " + radius + "Perimeter = " + perimeter + "\n");
}
```

Change to code of the main so no variables are used

Format the output so that only 3 decimal digits are displayed

LET'S CODE

Next week

- **Arrays**
- **ArrayList**
- **How to use Arrays and ArrayLists**