Northeastern University

# Lecture 3: Fundamentals of Programming - 3

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from Prof. Charlie Wiseman's materials.

# Outline

- Introduction to Computation and Programming
- Variables, I/O, Types and Strings
- Control Flow and Conditions
- Methods
- Arrays
- File I/O

# Outline

- Introduction to Computation and Programming
- Variables, I/O, Types and Strings
- Control Flow and Conditions
- Methods
- Arrays
- File I/O

# Methods

- Programs can be logically broken down into a set of tasks.
- Individual tasks can be separated out from the main program into *methods*.
- A method is simply a mini-program that completes a specific task.

# Types of Methods

- Predefined Methods
- Programmer-Defined Methods

# Predefined Methods

- Java includes many predefined methods for common programming tasks.
- Example of using the predefined square root method, `Math.sqrt()`:

```
double root, input_value;
System.out.print("Enter a number: ");
input_value = ...ouble();

root = Math.sqrt(input_value);

System.out.println("The square root is " + root);
```

returned value

method call

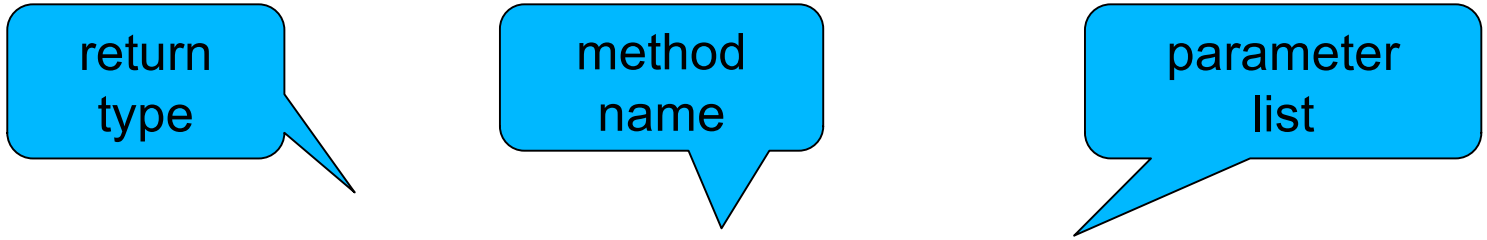
parameter

# Generic Form

```
RETURN_TYPE METHOD_NAME(PARAMETER_1, PARAMETER_2, …, PARAMETER_N)
```

- A method can have any number of parameters
  - » Each parameter has a specified type (**int**, **double**, String, etc.).
- A method has either zero or one return value(s).
  - » The return value is commonly the result of the method.
  - » The return value can be assigned to a variable of the same type.
  - » When the method call is placed directly in another Java expression, its return value will be used to replace the method call later.

# A Few Java Methods




- Square root: `double Math.sqrt(double a)`
- Power: `double Math.pow(double base, double exp)`
- Absolute value: `double Math.abs(double a)`
- Natural log: `double Math.log(double a)`
- Log base 10: `double Math.log10(double a)`

# More Examples

```
double number, cube, log2;
System.out.print("Enter a number: ");
number = input.nextDouble();

System.out.println(number + "'s square root is " + Math.sqrt(number));

cube = Math.pow(number, 3.0);
System.out.println(number + "^3.0=" + cube);

log2 = Math.log(number) / Math.log(2.0);
System.out.println("log2(" + number + ")=" + log2);
```

# Exercise

- Write a program that prints out the value of $2^x$ for x=1,2,3,…,32
- Use the `Math.pow()` method and a **while** loop

# Answer

```
double x = 1;
double pow2;

while (x <= 32) {
    pow2 = Math.pow(2, x);
    System.out.printf("2^%.0f=%.0f%n", x, pow2);
    x++;
}
```

System.out.printf() is just another method! It has a String parameter followed by one argument for each % place holder

# Programmer-Defined Methods

- Java allows you to define your own methods to meet the needs of your specific program.
- To define your own method, you need to write the method *signature* and the method *body.*
- The signature includes the return type, method name and parameter types.
- The body is the set of Java statements that will be executed when the method is invoked.

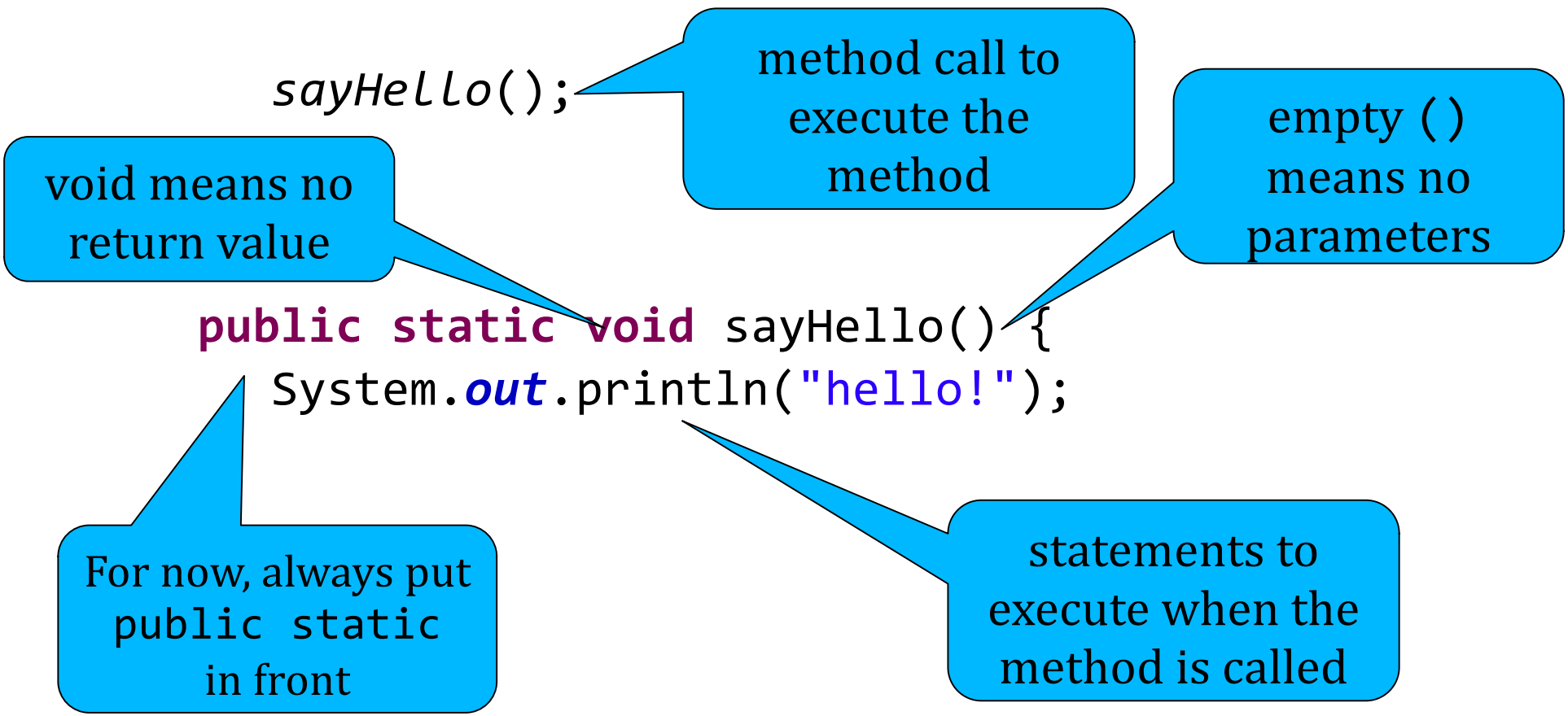
# No Parameters, No Return Value

```
public class ClassExamples {

    public static void main(String[] args) {

        sayHello();

    public static void sayHello() {
        System.out.println("hello!");
```



# The Process of Methods

- When a program executes a method, it temporarily stops where it is in main(), goes to the lines of code in the method, and executes those lines like normal.
- Then, when you get to the end of the method (or a **return** statement) it goes back to main() and resumes executing after the method call.

# No Parameters, One Return Value

```
import java.util.Scanner;

public class ClassExamples {

    public static void main(String[] args) {
        int i;
        i = getInteger();
        System.out.println("Got: " + i);
    }

    public static int getInteger() {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int input_value = input.nextInt();
        return input_value;
    }

}
```

`int` means the method returns an integer

Have to return an integer value

# Return Values

- Methods have zero or one return values.
- If a method has a return value, it is of a specific type (**int**, **double**, String, …).
  - » Type is defined as part of the method signature.
- Use the **return** statement to return a value of the specified type
  - » Can be a constant, variable, expression, method, or anything that is evaluated to the required type

# Another Example

```
import java.util.Scanner;

public class ClassExamples {

    public static void main(String[] args) {
        String s;
        s = getString();
        System.out.println("Got: " + s);
    }

    public static String getString() {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input_value = input.next();
        return input_value;
    }

}
```

# Exercise

- Write a method named getDouble() that reads a (`double`) value from the user and returns it to main(), then print the value in main().

# Answer

```
import java.util.Scanner;

public class ClassExamples {

    public static void main(String[] args) {
        double val;
        val = getDouble();
        System.out.println("Got: " + val);
    }

    public static double getDouble() {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        double input_value = input.nextDouble();
        return input_value;
    }

}
```

# Methods with Parameters

- Methods can take any number of parameters.
- Each parameter has a set type (**`int`**, **`double`**, String, …), defined as part of the method signature.
- When called, the *value* of the argument is passed to the method.

# Example with Two Parameters

```
import java.util.Scanner;

public class ClassExamples {

    public static void mai
        Scanner input = new
        double input1, inpu
        System.out.print("E
        input1 = input.nextDoub
        u
        a                                    ut2);
        r                                 .3f%

    public static double doCalculation(double a, double b) {
        return (a*a + b*b);
    }
}
```

the current value of input1 is passed to the method as a

the current value of input2 is passed to the method as b

double means the method returns a double value

double means the first parameter is a double value

double means the second parameter is double value

# Parameters

- Each time a method is called, you can pass in different arguments.

```
public class ClassExamples {
    public static void main(String[] args) {
        double result1, result2;
        result1 = doCalculation(3, 4);
        System.out.printf("result1 is %.3f%n", result1);
        result2 = doCalculation(2, 8);
        System.out.printf("result2 is %.3f%n", result2);
    }
    public static double doCalculation(double a, double b) {
        return (a*a + b*b);
    }
}
```

# Multiple `return` Statements

- Methods (including the `main()` method) can have multiple `return` statements in them.
- When a `return` statement is executed, the method stops and the stated value is returned to the caller immediately.

# Multiple `return` Statements Example

Methods that return a boolean are often used in boolean expressions

Methods can have multiple return statements

```
import java.util.Scanner;

public class ClassExamples {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int input_value;
        System.out.print("Enter an integer: ");
        input_value = input.nextInt();
        if(isEven(input_value)) {
            System.out.println(input_value + " is even!");
        }
        else {
            System.out.println(input_value + " is odd!");
        }
    }

    public static boolean isEven(int number) {
        if (number % 2 == 0) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

# Take Home Points

- Methods are mini-programs that are generally used to contain all of the code to complete some particular task.
- Methods can have either zero or one return value(s).
  - » If it has one, the value is of a specified type.
- Methods can have zero or more parameters.
  - » Each parameter (if any) has a specified type.
  - » When called, the current values of the arguments are plugged in and passed as values to the method.
- Methods can have multiple **return** statements.