Introduction to Java programming

By STEMPowering Girls
stempoweringgirls.org

What is Java?

- It is a programming language
- You can write commands to do different tasks
- There are set rules on how you give these commands

Object oriented programming

- Java is OOP language
- Meaning it is based on objects that are made in the program
- These objects can contain data and have different tasks or procedures called methods

Step 1:
Have JGRasp and Java downloaded

Classes

- A class is a user defined blueprint from which objects are created. It represents the set of methods that are common to all objects of one type.

```
public class Dog {
    String breed;
    int age;
    String color;

    void barking() {
    }

    void hungry() {
    }

    void sleeping() {
    }
}
```

Dog Class

- This class is called the dog class
- There are different variables in this class that are defined
  - Breed
  - Age
  - color
- There are different methods in this class
  - barking
  - hungry
  - sleeping

```
public class Dog {
    String breed;
    int age;
    String color;

    void barking() {
    }

    void hungry() {
    }

    void sleeping() {
    }
}
```

Some variable types are String and Int

STEMPowering Girls
https://stempoweringgirls.org

Main method

- When someone runs an application , it starts by calling the class's main method
- The main method then calls all the other methods required to run your application

```
public class Test {

public void main(String[] args){


}
}
```

Print

System.out.println(); is the syntax used to print something

```
System.out.println("STEMPowering Girls");
```

```
Output: STEMPowering Girls
```

Concatenating strings

Concatenation refers to joining 2 or more data types together in a String using the '+' operator.

Ex:

```
String name = "Sam";
int grade = 8
String school = "Longfellow Middle School";
System.out.println(name + " is in grade " + grade + " and goes to " + school);
```

```
Output: Sam is in grade 8 and goes to Longfellow Middle School
```

Try to print the words "Hello world"

Primitive Data Types

Integer

```
    int number = 5;
```

Double

```
    double decimal = 4.3;
```

String

```
    String word = “Hello!”;
```

Math Operators

Assume integer variable A holds 10 and variable B holds 20, then −

Operator	**Description**	**Example**
+ (Addition)	Adds values on either side of the operator.	A + B will give 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B will give -10
* (Multiplication)	Multiplies values on either side of the operator.	A * B will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	B / A will give 2
% (Modulus)	Divides left-hand operand by right hand operand and returns the remainder as an integer.	B%A will give 0

Relation Operators

Assume integer variable A holds 10 and variable B holds 20, then –

> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Methods

Methods in Java are segments of code that perform a certain task. They exist outside of the main method and can be called in the main method by using their name. Methods are like subprograms that take in data inputted by the programmer or user and usually return data. There is no limit to the amount of methods a program can have.

Methods are often used to perform repeated tasks throughout a program. This way, all the code to perform that task only appears once and does not have to be rewritten multiple times in the main method.

Examples of tasks that could be completed using methods:

- Adding a percentage of tax to a subtotal
- Calculating a person’s BMI given weight and height
- The largest value given two integers or doubles

Writing a method

Basic parts of a method:
-header
-body

Ex: method that prints out the statement “This is a method”

```
public static void example(){
     System.out.println(“This is a method”);
}
```

To call the method in the main method:

```
public static void main(String[]args){
     example();
}
```

Output: This is a method

Calling a method in the main method

```java
public class MyClass {

  public static void myMethod() {

    System.out.println("I just got executed!");

  }

  public static void main(String[] args) {

    myMethod();

  }

}

// Output: "I just got executed!"
```

Methods with Return types

The return type of a method specifies what the method will return, or output to the user. The return type of a method is established in the method's header like so.

void
-means that the method does not return anything
-to call in the main method, type the method name with any parameters
- ex header: public static void print()
int
-means that the method returns an integer value, needs the keyword **return**
-needs to be within a printing statement when called in the main method
-ex header: public static int average()
double
-means that the method returns a double value, needs the keyword **return**
-needs to be within a printing statement when called in the main method
-ex header: public static double maxValue()
String
-means that the method returns a double value, needs the keyword **return**
-needs to be within a printing statement when called in the main method
-ex header: public static String uppercase()

Methods with Parameters

A parameter is a value passed to a method. Methods can take one or more parameters. If methods work with data, we must pass the data to the methods. This is done by specifying them inside the parentheses. In the method definition, we must provide a name and type for each parameter.

Examples of methods with parameters:

```
public static int maxValue(int x, int y){}

public static double average(double a, double b){}

public static String student(String name, int idNumber){}
```

Method with a return type and parameters

Ex: a method that has two parameters (String name and int age), and returns a String with the individual’s name and age.

```
public static String sentence(String name, int age){
     return “This person’s name is ” + name + “ and they are ” + age + “ years old.”);
}
```

To call this method in the main method:

```
System.out.println(sentence(Sarah,16)); //Output: This person’s name is Sarah and they are 16 years old
```

OR

```
String example = sentence(Sarah,16);
System.out.println(example); //Output: This person’s name is Sarah and they are 16 years old
```

both ways of calling the method have the same parameters and produce the same output

Write a method that takes 2 integers as parameters and returns their average

Write a method that takes two String parameters and then returns a new string that is a concatenation of the 2 parameters.

Reading a users input

- First you have to import the Scanner
- Then create an instance of the Scanner
- Tell your user what to type
- Take what the typed and put it into a variable
- Close the Scanner

```
import java.util.Scanner;

Scanner reader = new Scanner(System.in);
//Reading from System.in

System.out.println("Enter a number: ");

int n = reader.nextInt(); // Scans the
next token of the input as an int.

reader.close();
```

Scanner methods

nextInt() - scans the user input as an integer

nextDouble() - scans the user input as a double

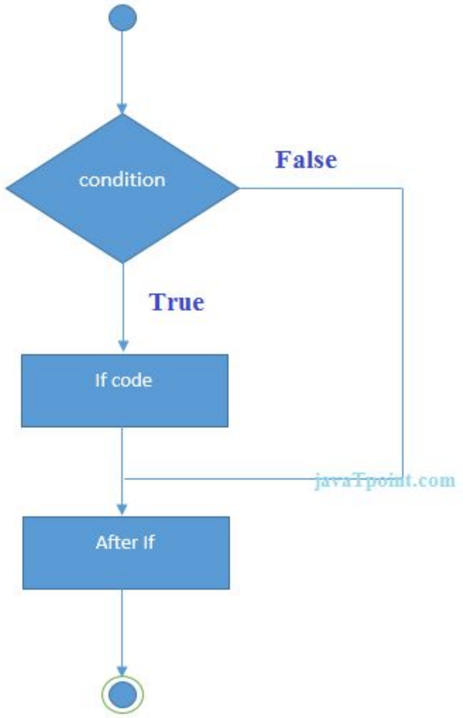
nextBoolean() - scans the user input as a boolean

nextLine() - scans the user input as a String with all whitespace

next() - scans the user input until the first instance of whitespace

Write a method that prints out a String inputted by the user through a scanner

If... else Statements



```
public class IfExample {
public static void main(String[] args) {
    int age=20;
    if(age>18){
        System.out.print("Age is greater than 18");
    }
}
}
```

Output: Age is greater than 18

Write a method that takes 2 integer parameters, a and b, and returns true if a>b

Using if statements, Write a method that takes 2 doubles as parameters and returns the greater double value

Now Try and ask the user for a number and return whether the number is even or not

Sample Program

```
import java.util.Scanner;

public class EvenOdd {

    public static void main(String[] args) {

        Scanner reader = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = reader.nextInt();

        if(num % 2 == 0)
            System.out.println(num + " is even");
        else
            System.out.println(num + " is odd");
    }
}
```