



# Coding-bootcamps.com

## Introduction to Java Programming Language

By Jim Sullivan  
from [Coding Bootcamps](https://coding-bootcamps.com)



# Coding Bootcamps

# Application Development Fundamentals

Session 2

# Brief Recap

- Re-hash what we've covered in our first class

# Objective

- Learn about the structure of a Java Program
- Learn about memory concepts
- Learn about Fundamental Data Type Declarations
- Learn about Fundamental I/O Concepts
- Learn about Fundamental Operators
- Make our first program

# Materials

- These Powerpoint Slides

# Structure Of A Java Program

- A package
- A class
- Data members
- User defined methods
- Block Statements

# Structure Of A Java Program (Package)

- A package
  - A package is a collection of classes, interfaces and sub-packages.
  - A sub package contains collection of classes, interfaces and sub-sub packages.
- `java.lang.*`; package is imported by default and this package is known as default package.

# Structure Of A Java Program (Class)

- A class
  - A class is keyword used for developing user defined data type and every java program must start with a concept of class.
  - A class has a class name and that's what defines the actual data type
- Example:
  - `public class HelloWorld`



# Structure Of A Java Program (Data Member)

- Data members
  - Represent either instance or static they will be selected based on the name of the class.
- Example:
  - `int age = 17;`
  - `String name = "Bryan";`
  - `double gpa = 3.2`

# Structure Of A Java Program (User defined methods)

- User-defined methods are meant for performing the operations either once or each and every time.
- Example:
  - `print_age();`
  - `calulate_gpa();`
  - `get_course_schedule();`

# Structure Of A Java Program (Block Statements)

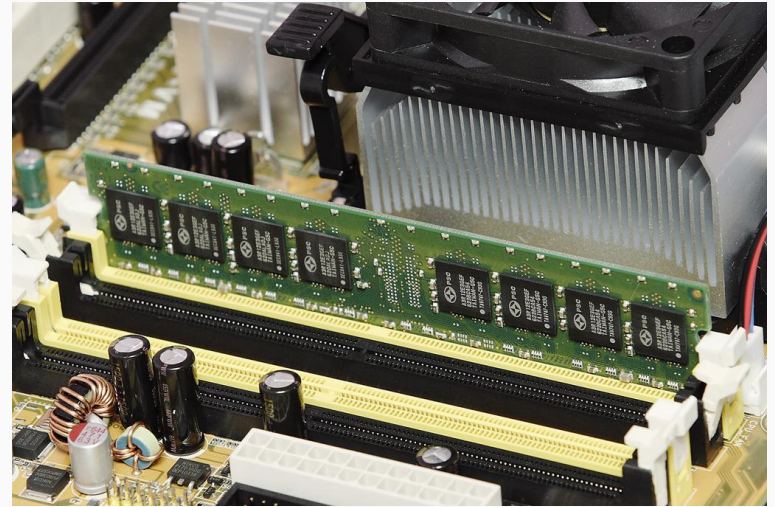
- Block Statements represent sets of executable statements which are in term calling user-defined methods
- Example:
  - `System.out.println("Hello Class!");`

# Structure Of A Java Program (Putting it all together)

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        // Prints "Hello, World" to the terminal window.  
        System.out.println("Hello, World");  
    }  
  
}
```

# Understanding Memory Management

- Variable names such as age, name, and gpa actually correspond to locations in the computer's memory.
- Every variable has a name, a type, a size (in bytes) and a value.



# Fundamental Data Type Declarations

- In JAVA there are primitive (basic) data types
  - float (1.234)
  - Int (17)
  - double (3.14)
  - String ("Bryan")
  - boolean (false)
  - char ('t')
  - byte (0)

# Fundamental Data Type Declarations

- Java is a strongly typed language
  - Every variable must have an explicit type
  - Expressions assigned to the variable must be a compatible type
- Local variables (declared within methods) must be declared and initialized before they are used
- Class and instance variables can be declared anywhere and are initialized to defaults:
- Multiple variables can be declared on a single line: `int i, j, k;`

# Fundamental I/O Concepts

- The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java.
- All these streams represent an input source and an output destination.
- A stream can be defined as a sequence of data. There are two kinds of Streams
  - InputStream – The InputStream is used to read data from a source.
  - OutputStream – The OutputStream is used for writing data to a destination.





# Fundamental I/O Concepts

- Java byte streams are used to perform input and output of 8-bit bytes.
- The most frequently used classes are, `FileInputStream` and `FileOutputStream`.

# Fundamental I/O Concepts

```
import java.io.*;
public class ReadConsole {

    public static void main(String args[]) throws IOException {
        InputStreamReader cin = null;

        try {
            cin = new InputStreamReader(System.in);
            System.out.println("Enter characters, 'q' to quit.");
            char c;
            do {
                c = (char) cin.read();
                System.out.print(c);
            } while(c != 'q');
        } finally {
            if (cin != null) {
                cin.close();
            }
        }
    }
}
```

# Fundamental I/O Concepts

```
$javac ReadConsole.java
```

```
$java ReadConsole
```

```
Enter characters, 'q' to quit.
```

```
1
```

```
1
```

```
e
```

```
e
```

```
q
```

```
q
```

# Fundamental Operators (Arithmetic)

- Let A = 10 and B = 20

Operator	Description	Example
+ (Addition)	Adds the values on either side of the operator	A + B = 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B = -10
* (Multiplication)	Divides left-hand operand by right-hand operand.	A * B will give 200
/ (Division)	Adds the values on either side of the operator	B / A will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	B % A will give 0
++ (Increment)	Increases the value of operand by 1.	B++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	B-- gives 19

# Fundamental Operators (Logical Operators)

- Let A = 10 and B = 20

Operator	Description	Example
== (equal to)	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
> (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.

# Building and Deploying a Java Program

- Create the program by typing it into a text editor and saving it to a file named, say, `MyProgram.java`.
- Compile it by typing `"javac MyProgram.java"` in the terminal window.
- Execute (or run) it by typing `"java MyProgram"` in the terminal window.
- The first step creates the program; the second translates it into a language more suitable for machine execution (and puts the result in a file named `MyProgram.class`); the third actually runs the program.

# Building and Deploying a Java Program

- Creating a Java program.
  - A program is nothing more than a sequence of characters, like a sentence, a paragraph, or a poem. To create one, we need only define that sequence characters using a text editor in the same way as we do for email.
  - HelloWorld.java is an example program. Type these character into your text editor and save it into a file named HelloWorld.java.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // Prints "Hello, World" in the terminal window.  
        System.out.println("Hello, World");  
    }  
}
```

# Building and Deploying a Java Program

- Compiling a Java program.
  - A compiler is an application that translates programs from the Java language to a language more suitable for executing on the computer.
  - It takes a text file with the .java extension as input (your program) and produces a file with a .class extension (the computer-language version).
- To compile HelloWorld.java type the boldfaced text below at the terminal.

```
javac HelloWorld.java
```



# Building and Deploying a Java Program

- Compiling a Java program.
  - A compiler is an application that translates programs from the Java language to a language more suitable for executing on the computer.
  - It takes a text file with the .java extension as input (your program) and produces a file with a .class extension (the computer-language version).
- To compile HelloWorld.java type the text below at the terminal.

```
javac HelloWorld.java
```

- To run the application type the text below at the terminal

```
java HelloWorld
```

# Assignment

- Write a program that prints Hello World 10 times
- Write a program that shows understanding of all the operator types

# Summary

# Live private coaching sessions for Java

- [Private tutoring sessions for software design and engineering- Weekly and monthly plans](#)
- [Java programming language- Private tutoring sessions](#)



Coding-bootcamps.com

Thank You



**Coding**  
Bootcamps