JAVA

Class One

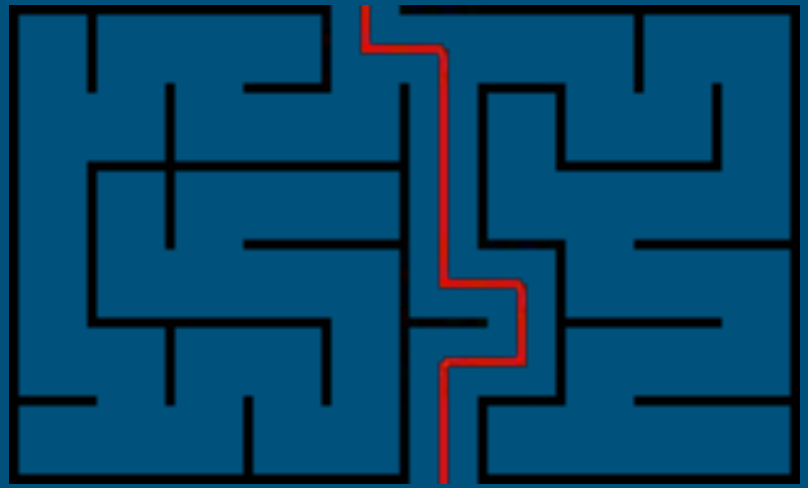
Program

A sequence of coded instructions that can be inserted into a mechanism (such as a computer)

End

Start

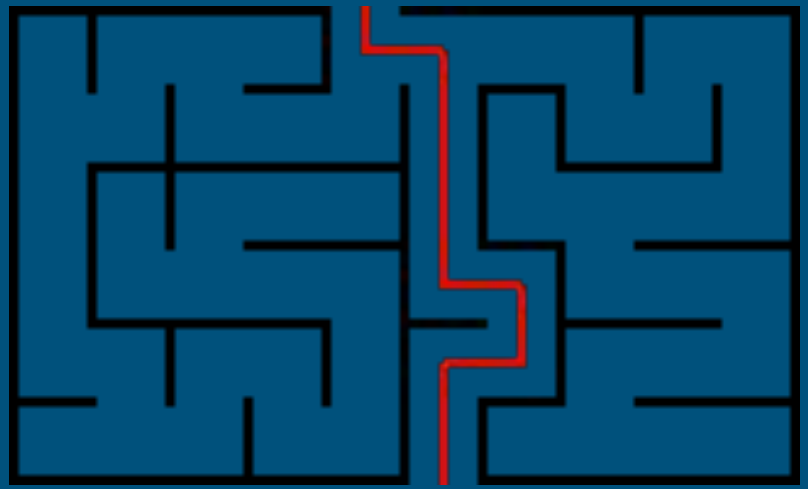
End



Start

Right
Left
Left
Right
Left
Right

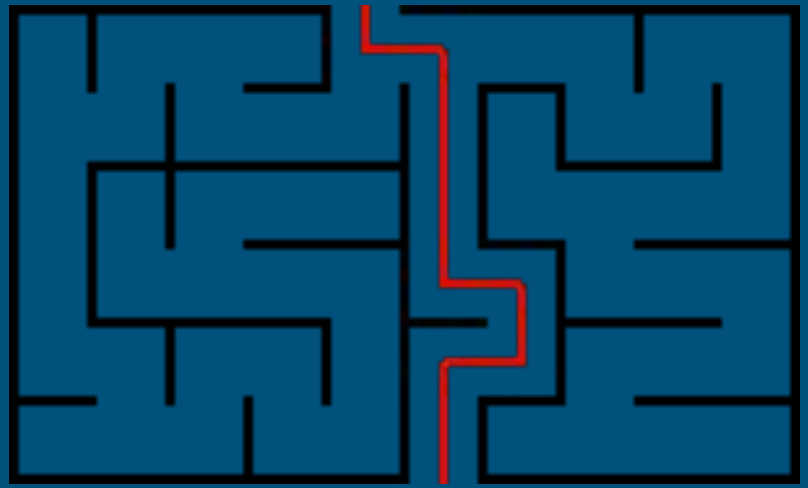
End



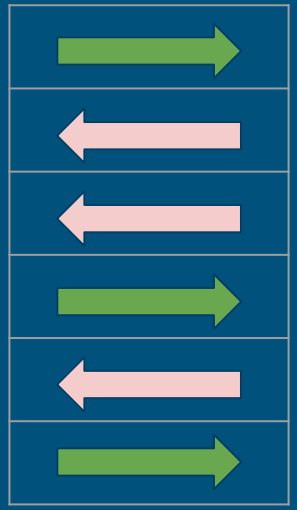
Start

Derecho
Izquierda
Izquierda
Derecho
Izquierda
Derecho

End



Start



Binary Code

A binary code represents text, computer processor instructions, or any other data using "0" and "1"

Binary

Something made of two things or parts.

Computers use the binary number system, which includes only the digits 0 and 1

Number System

Using ten digits(0-9) we can represent unlimited numbers

Alphabet System

Using 26 letters(A-Z) we can tell unlimited stories

Binary Number System

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Advantages of Java

Java is easy to learn – *Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.*

Java is object-oriented – *This allows you to create modular programs and reusable code.*

Java is platform-independent – *One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.*

JVM

JRE

JDK

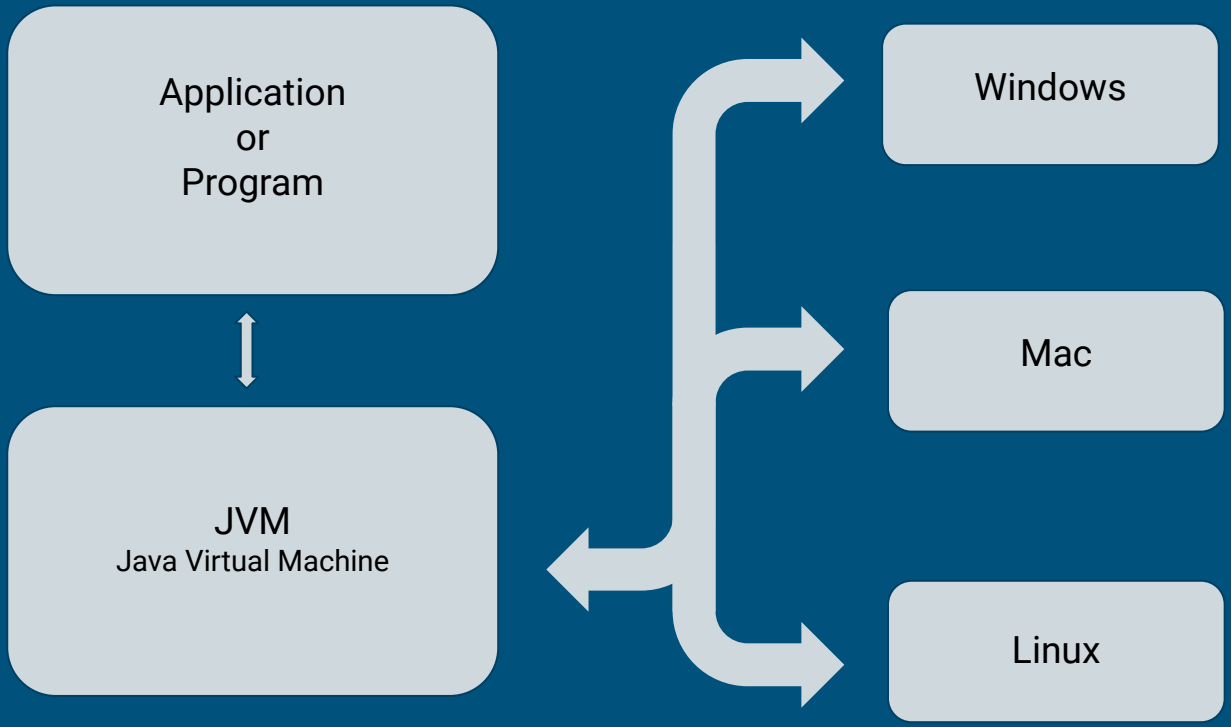
JVM (Java Virtual Machine)

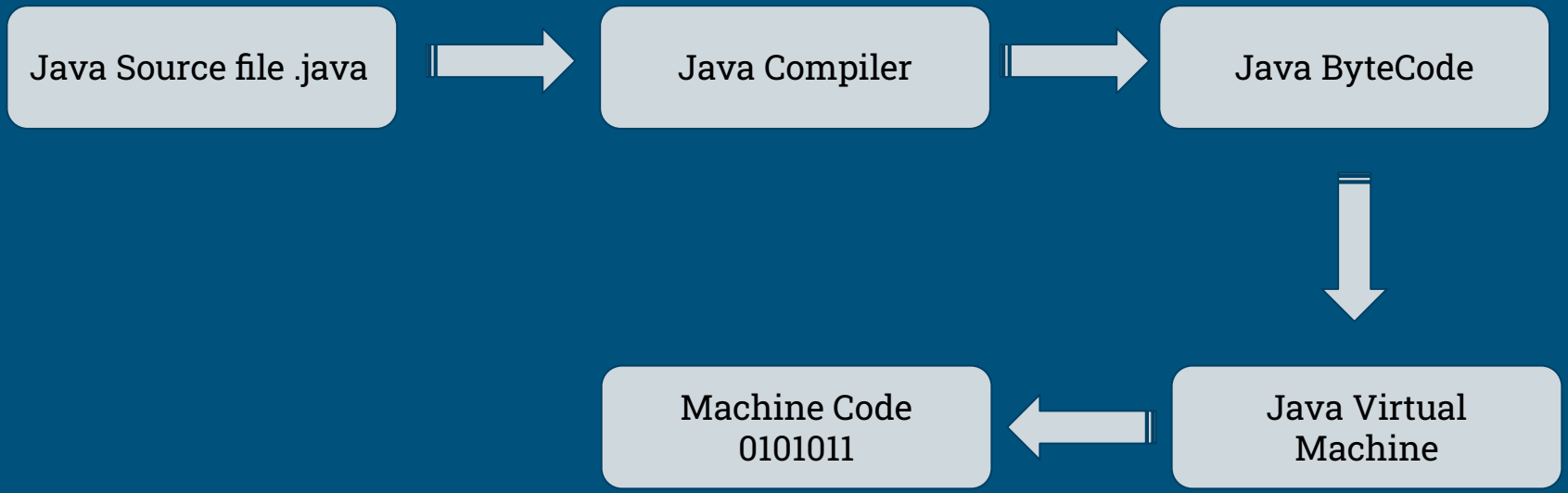
The JVM manages system memory and provides a portable execution environment for Java-based applications. The JVM is how we run our Java programs.

What the JVM is used for

The JVM has two primary functions: to allow Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle), and to manage and optimize program memory.

Write once, run anywhere




Java Source file .java
Java Compiler
Java ByteCode
Java Virtual Machine
Machine Code
0101011

Java ByteCode

Java bytecode is the bytecode-structured instruction set of the Java virtual machine (JVM).

Java Code

```
outer:
for (int i = 2; i < 1000; i++)
{
    for (int j = 2; j < i; j++)
{
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
  }
```

Java Bytecode

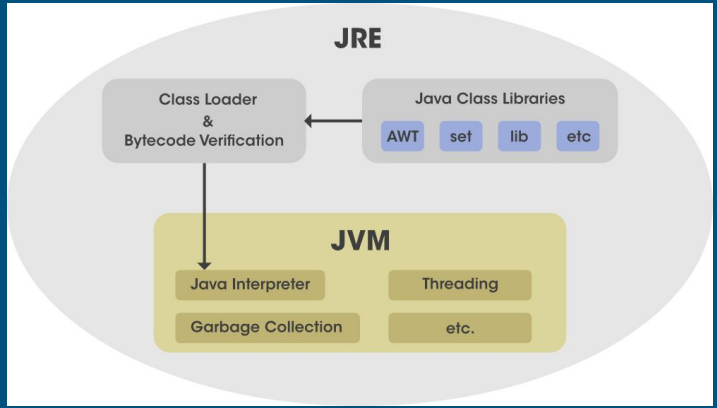
```
0:   iconst 2
1:   istore 1
2:   iload 1
3:   sipush  1000
6:   if icmpge       44
9:   iconst 2
10:  istore 2
11:  iload 2
12:  iload 1
13:  if icmpge       31
16:  iload 1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic       #84; // Field
java/lang/System.out:Ljava/io/PrintStre
am;
34:  iload 1
35:  invokevirtual   #85; // Method
java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
   44:  return
```

JRE (Java Runtime Environment)

A Java™ runtime environment (JRE) is a set of components to create and run a Java application. A JRE is part of a Java development kit (JDK).

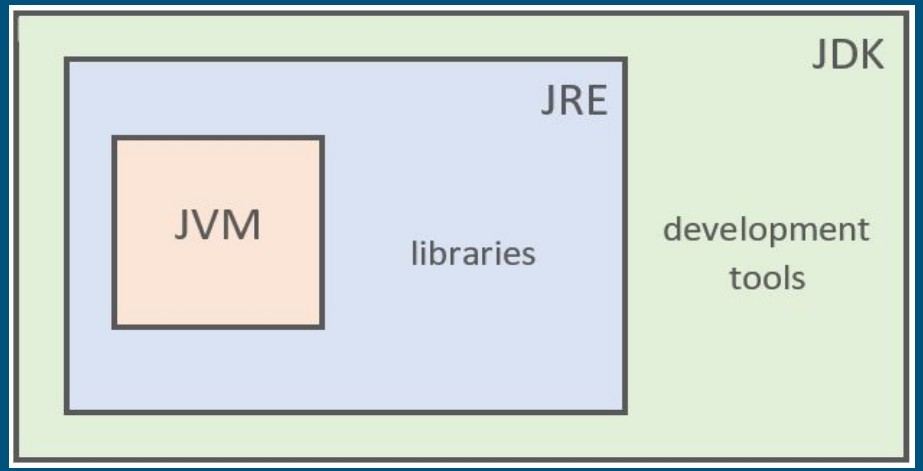
A JRE is made up of a Java virtual machine (JVM), Java class libraries, and the Java class loader. JDKs are used to develop Java software; JREs provide programming tools and deployment technologies; and JVMs execute Java programs.

https://www.ibm.com/cloud/learn/jre



JDK (Java Development Kit)

The Java Development Kit (JDK) is a cross-platformed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets. It is a core package used in Java, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment).
https://www.geeksforgeeks.org/jdk-in-java/



JDK - Java Development Kit

Developer

Development Tools

java, javac,JShell, javadoc, jar, jdeps,JConsole, VisualVM,
JFR, JPDA, JVM TI, IDL, Java DB, Debugging Tools, Deployment Tools, Monitoring Tools

JRE - Java Runtime Environment

Java App Users

Java API, Runtime Libraries, Byte Code Verifier,
Libraries: User Interface (Swing, JavaFx, Sound, Image I/O .....),
Integration Libraries (JDBC, RMI, . .), Serialization, Netowrking,
lang and util libraries (Math, Collections, Concurrency, Logging, ...).....

JVM - Java Virtual Machine

Java Interpreter, JIT, Garbage Collector, Thread Synchronization,
Class Loader Subsystem

JDK = JRE + Development Tools
JRE= JVM + Libraries

Interpreted vs Compiled Programming

Compiled

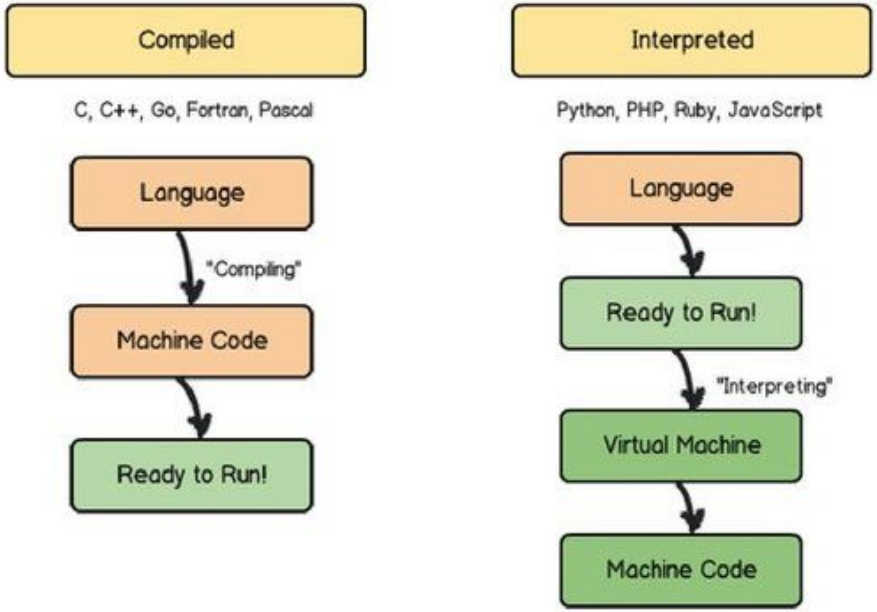
- Compiled languages are converted directly into machine code that the processor can execute.
- Compiled languages are faster
- Compiled languages need a “build” step
- You need to “rebuild” the program every time you need to make a change
- Compiled languages are Platform dependence of the generated binary code
- Pure compiled languages are C, C++, Erlang, Haskell, Rust, and Go.

Interpreted

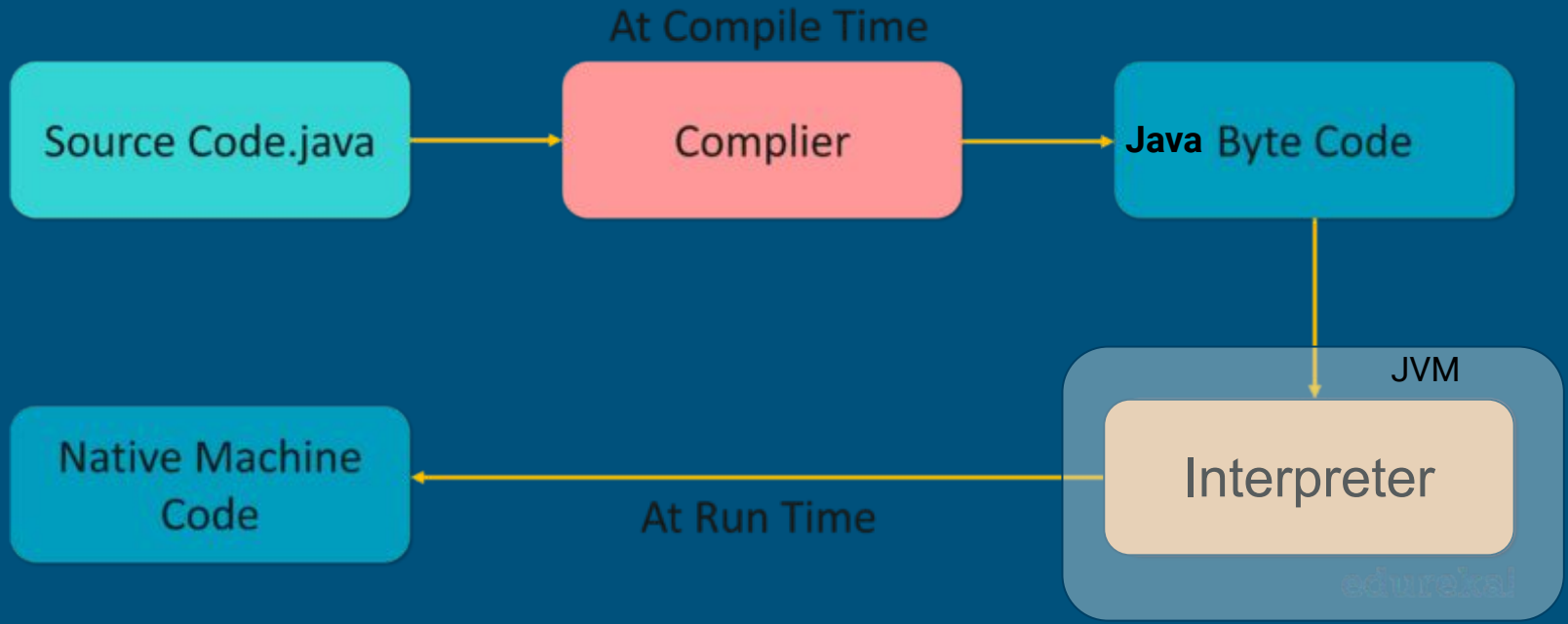
- Interpreters run through a program line by line and execute each command.
- Interpreted languages were once significantly slower than compiled languages.
- Common interpreted languages are PHP, Ruby, Python, and JavaScript.

https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/

Compiler vs. Interpreted



Java is Both Compiled and Interpreted

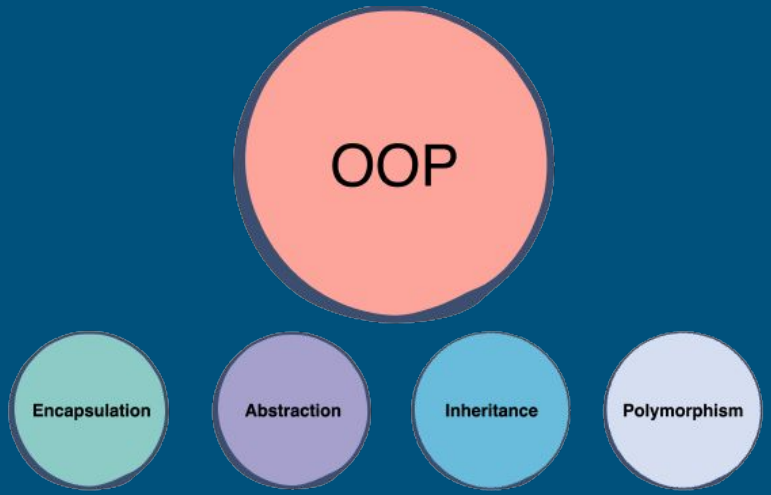

At Compile Time
Source Code.java
Complier
Java Byte Code
JVM
Native Machine Code
At Run Time
Interpreter

Questions?

Object-oriented programming (OOP)

Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects. It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects. There are many object-oriented programming languages including JavaScript, C++, Java, and Python.

Four Principles of OOP



- Inheritance: *child classes inherit data and behaviors from parent class*
- Encapsulation: *containing information in an object, exposing only selected information*
- Abstraction: *only exposing high level public methods for accessing an object*
- Polymorphism: *many methods can do the same task*

Building blocks of OOP

- Classes
- Objects
- Methods
- Attributes

Benefits of OOP

- OOP models complex things as reproducible, simple structures
- Reusable, OOP objects can be used across programs
- Allows for class-specific behavior through polymorphism
- Easier to debug, classes often contain all applicable information to them
- Secure, protects information through encapsulation

Classes and Objects

Classes

Classes are essentially user defined data types. Classes are where we create a blueprint for the structure of methods and attributes. Individual objects are instantiated, or created from this blueprint.

Attributes

Attributes are the information that is stored. Attributes are defined in the Class template. When objects are instantiated individual objects contain data stored in the Attributes field.

Objects

Objects are instances of classes created with specific data.

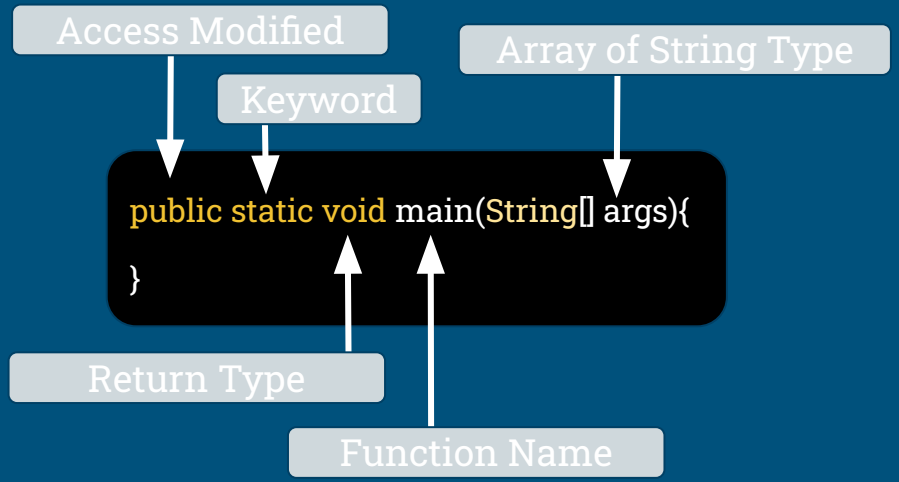
Objects are created from classes.

Methods

Methods represent behaviors. Methods perform actions; methods might return information about an object, or update an object's data. The method's code is defined in the class definition.

Java main() method

The main() is the starting point of a Java program. The JVM calls the main method to start execution of a Java program. Without the main() method, JVM will not execute the program



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

}
```

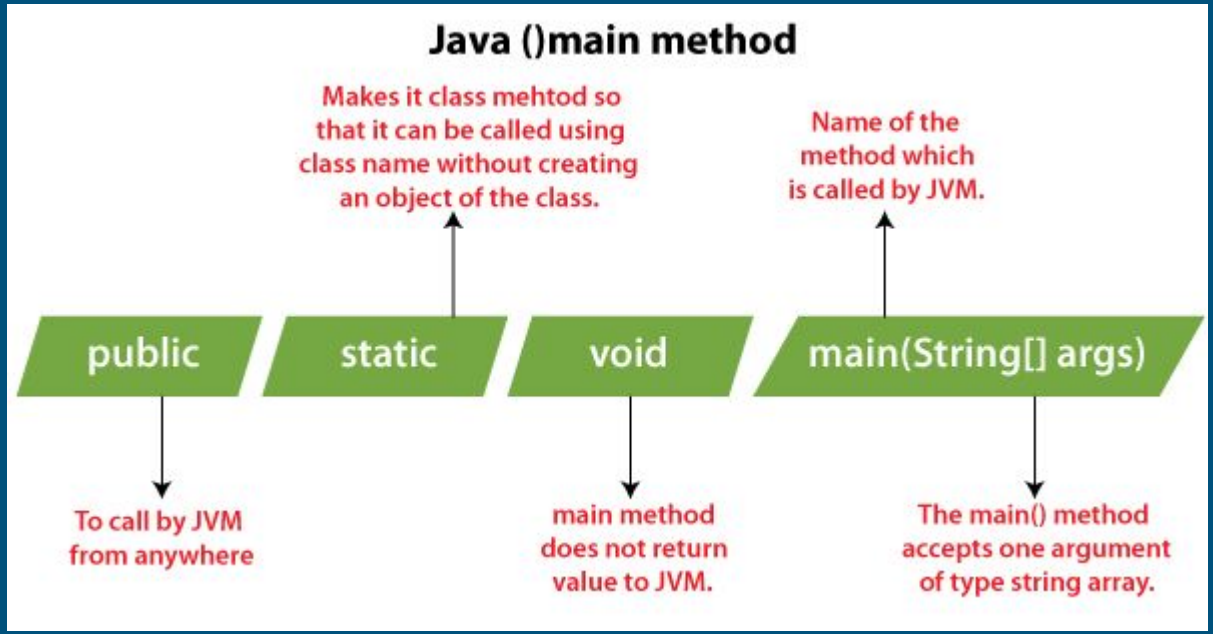
public: It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected, and default before the main() method, it will not be visible to JVM.

static: You can make a method static by using the keyword ***static***. We should call the main() method without creating an object. Static methods are the method which invokes without creating the objects, so we do not need any object to call the main() method.

void: In Java, every method has the return type. ***Void*** keyword acknowledges the compiler that main() method does not return any value.

main(): It is a default signature which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the main() method.

String args[]: The main() method also accepts some data from the user. It accepts a group of strings, which is called a string array. It is used to hold the command line arguments in the form of string values.


Java ()main method
Makes it class mehtod so that it can be called using class name without creating an object of the class.
Name of the method which is called by JVM.
public
static
void
main(String[] args)
To call by JVM from anywhere
main method does not return value to JVM.
The main() method accepts one argument of type string array.

Print Statement

System.out.println();

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

First Java Program

```
public class MyFirstJavaProgram {

   public static void main(String []args) {

      System.out.println("Hello World");

   }

}
```

1. Open notepad and add the code as above.
2. Save the file as: MyFirstJavaProgram.java.
3. Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
4. Type 'javac MyFirstJavaProgram.java' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line (Assumption : The path variable is set).
5. Now, type ' java MyFirstJavaProgram ' to run your program.
6. You will be able to see ' Hello World ' printed on the window.

Java Keywords

Java has a set of keywords that are reserved words that cannot be used as variables, methods, classes, or any other identifiers.

List of keywords and what they do : https://www.w3schools.com/java/java_ref_keywords.asp

Complete List of Java Keywords

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert					

Java Operators Operators are used to perform operations on variables and values.

There are many types of operators in Java which are given below:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators

https://www.w3schools.com/java/java_operators.asp

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description
+	Addition	Adds together two values
-	Subtraction	Subtracts one value from another
*	Multiplication	Multiplies two values
/	Division	Divides one value by another
%	Modulus	Returns the division remainder
++	Increment	Increases the value of a variable by 1
--	Decrement	Decreases the value of a variable by 1

Java Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Java Comparison Operators

Comparison operators are used to compare two values and always returns a *boolean*

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Java Logical Operators

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Primitive data types

Primitive types are the most basic data types available within the Java language. There are 8: *boolean*, *byte*, *char*, *short*, *int*, *long*, *float* and double. These types serve as the building blocks of data manipulation in Java

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Java Variables

Variables are containers for storing data values.

In Java, there are different types of variables, for example:

- `String` - stores text, such as "Hello". String values are surrounded by double quotes
- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `float` - stores floating point numbers, with decimals, such as 19.99 or -19.99
- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `boolean` - stores values with two states: true or false

Syntax

```
type variableName;

type variableName = value;
```

Example

```
int number;

int number = 10;
```

Variable Memory Size

Example

```
int number;
```


```
double number;
```

Java Constant a variable whose value cannot change once it has been assigned.

In Java, constants(const) are declared using the final keyword

Syntax:

```
final type variableName = value;
```

Example:

```
final int number = 10;
```

Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

- Widening Casting (automatically) - converting a smaller type to a larger type size
  `byte -> short -> char -> int -> long -> float -> double`
- Narrowing Casting (manually) - converting a larger type to a smaller size type
  `double -> float -> long -> int -> char -> short -> byte`

Widening Casting

Widening casting is done automatically when passing a smaller size type to a larger size type:

Example

```
public class Main {
  public static void main(String[] args) {
    int myInt = 9;
    double myDouble = myInt; // Automatic casting: int to double

    System.out.println(myInt);      // Outputs 9
    System.out.println(myDouble);   // Outputs 9.0
  }
}
```

Try it Yourself »

Narrowing Casting

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

Example

```
public class Main {
  public static void main(String[] args) {
    double myDouble = 9.78d;
    int myInt = (int) myDouble; // Manual casting: double to int

    System.out.println(myDouble);   // Outputs 9.78
    System.out.println(myInt);      // Outputs 9
  }
}
```

Try it Yourself »

IDE

Integrated development environment

1. Syntax highlighting
2. Code completion
3. Refactoring
4. Version control
5. Debugging
6. Code search
7. Visual programming

Popular IDE

- Eclipse
- Intellij
- VSCode
- PyCharm
- RubyMine
- XCode