

What is Java

- **programming language** and a **platform**.
- high level, robust, secured and object-oriented programming language.
- **Platform:** Any hardware or software environment in which a program runs, is known as a platform.
- Since Java has its own runtime environment (JRE) and API, it is called platform.

Where it is used

- According to Sun, 3 billion devices run java.
- Desktop Applications such as acrobat reader, media player, antivirus etc.
- Web Applications such as irctc.co.in, javatpoint.com etc.
- Enterprise Applications such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games etc.

Types of Java Applications

1) Standalone Application

- Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Example of standalone applications are: Media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

2) Web Application

- An application that runs on the server side and creates dynamic page, is called web application.
Currently, [Servlet](#), [JSP](#), [Struts](#), [Spring](#), [Hibernate](#), [JSF](#) etc. technologies are used for creating web applications in java.

3) Enterprise Application

- An application that is distributed in nature, such as banking applications etc. is called enterprise application. It has the advantage of high level security, load balancing and clustering. In java, [EJB](#) is used for creating enterprise applications.

4) Mobile Application

- An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

Java Platforms / Editions

1) Java SE (Java Standard Edition)

- It is a java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, [String](#), Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection etc.

2) Java EE (Java Enterprise Edition)

- It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, [JPA](#) etc.

3) Java ME (Java Micro Edition)

- It is a micro platform which is mainly used to develop mobile applications.

4) JavaFx

- It is used to develop rich internet applications. It uses light-weight user interface API.

Why Java Programming named as "Java"

- **Why had they chosen java name for java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.
- According to James Gosling "Java was one of the top choices along with **Silk**". Since java was so unique, most of the team members preferred java.
- Java is an island of Indonesia where first coffee was produced (called java coffee).
- Notice that Java is just a name not an acronym.
- Originally developed by James Gosling at [Sun Microsystems](#) (which is now a subsidiary of Oracle Corporation) and released in 1995.
- In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.
- JDK 1.0 released in (January 23, 1996).

Java Version History

There are many java versions that has been released. Current stable release of Java is Java SE 10.

- JDK Alpha and Beta (1995)
- JDK 1.0 (23rd Jan, 1996)
- JDK 1.1 (19th Feb, 1997)
- J2SE 1.2 (8th Dec, 1998)
- J2SE 1.3 (8th May, 2000)
- J2SE 1.4 (6th Feb, 2002)
- J2SE 5.0 (30th Sep, 2004)
- Java SE 6 (11th Dec, 2006)
- Java SE 7 (28th July, 2011)
- Java SE 8 (18th March, 2014)
- Java SE 9 (21st Sep, 2017)
- Java SE 10 (20th March, 2018)

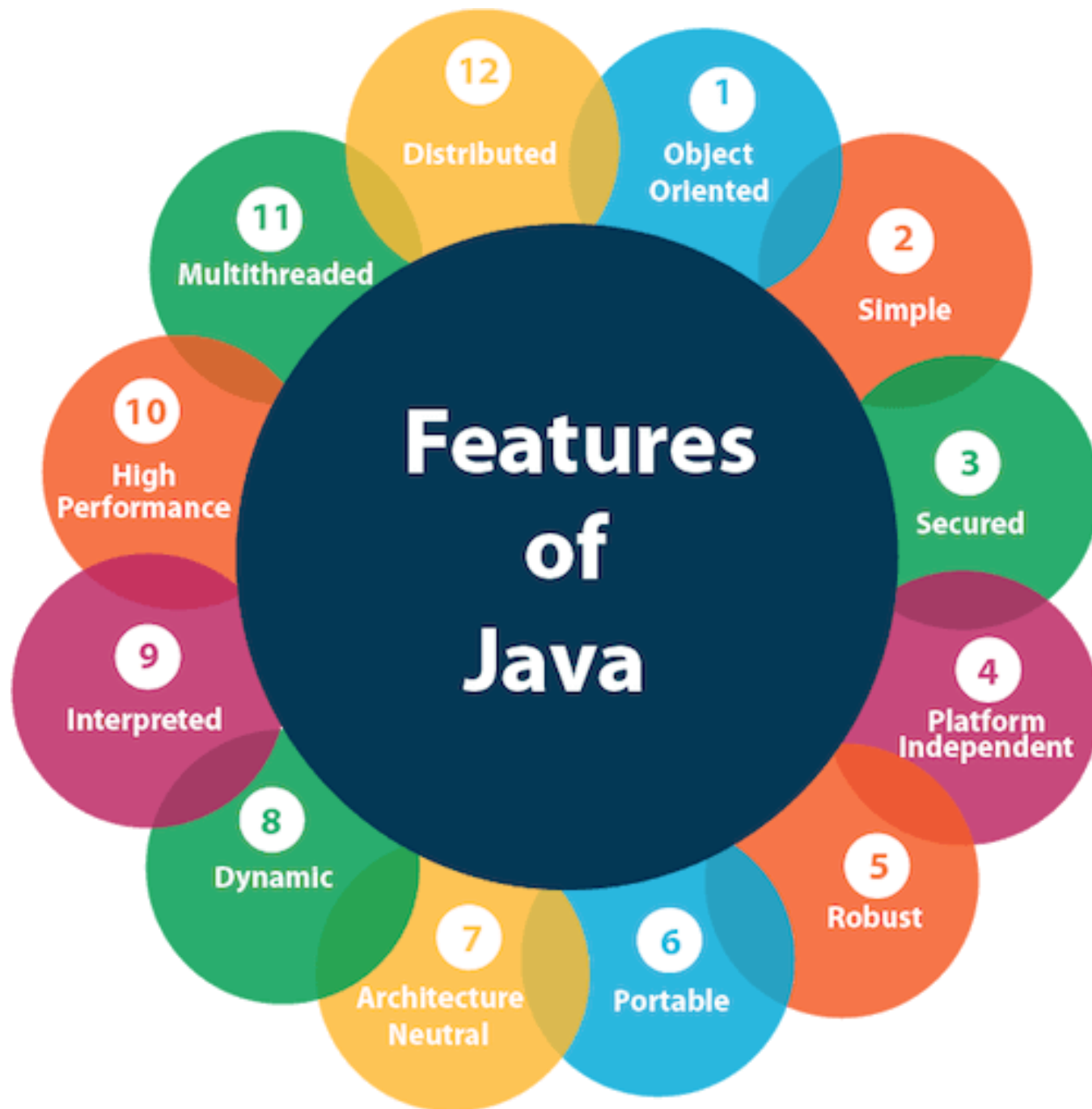
A list of most important features of Java language are given below.

- **Simple** - syntax is simple, clean and easy to understand.
- **Object-Oriented** – Object, Class, Inheritance, Polymorphism, Abstraction, Encapsulation
- **Portable** - Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent. It facilitates you to carry the java bytecode to any platform.
- **Platform independent** - write once, run anywhere language
e.g. Windows, Linux, Sun Solaris, Mac/OS etc
- **Secured** - With Java, we can develop virus-free systems
- **Robust** - strong memory management, automatic garbage collection
- **Architecture neutral** - no implementation dependent features e.g. size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

A list of most important features of Java language are given below.

- **Interpreted** -The Java interpreter can execute Java bytecodes directly on any machine to which the interpreter and run-time system have been ported.
- **High Performance** - Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.
- **Multithreaded** - We can write Java programs that deal with many tasks at once by defining multiple threads.
- **Distributed** - it facilitates users to create distributed applications in java RMI (Remote Method Invocation) and EJB(Enterprise JavaBeans) are used
- **Dynamic** - It supports dynamic loading of classes. It means classes are loaded on demand.

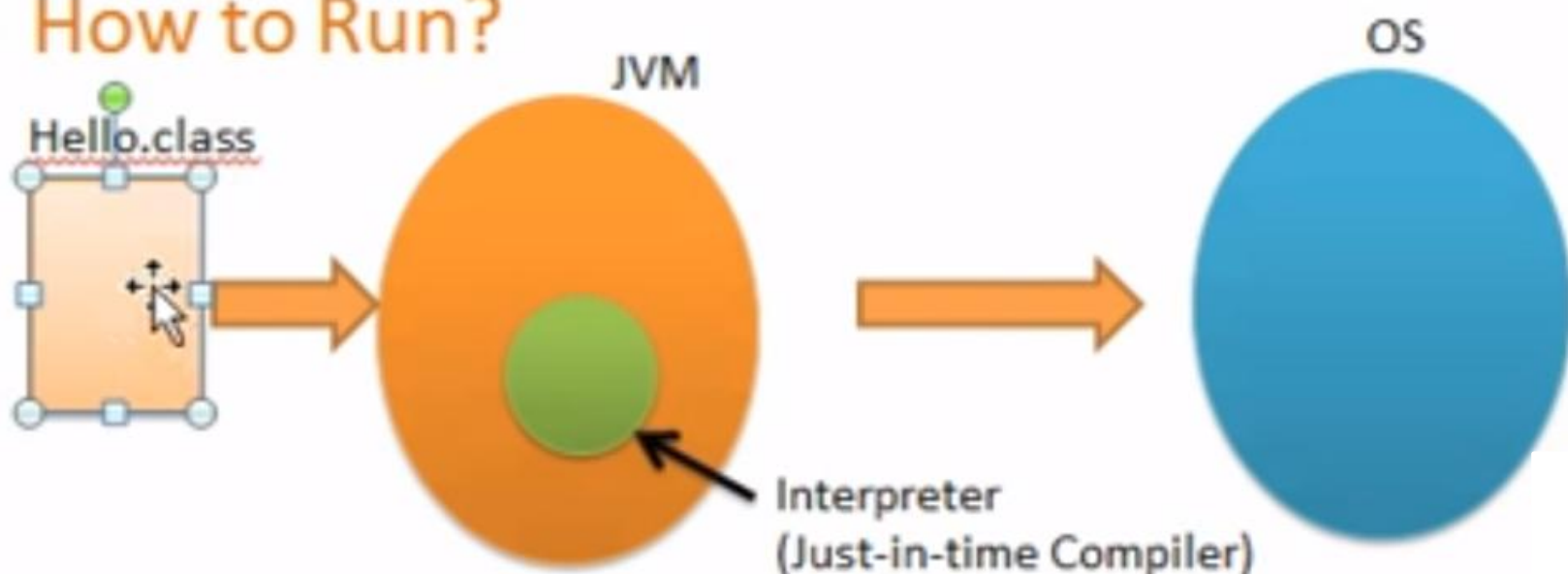


Compile and Run

How to Compile?



How to Run?



Installation and Setup

❑ <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

The screenshot shows the Oracle Java SE Downloads page. At the top, there is a navigation bar with tabs: Overview, Downloads, Documentation, Community, Technologies, and Training. The 'Downloads' tab is selected. Below the navigation bar, the heading 'Java SE Downloads' is displayed. There are two main download cards. The first card features the Java logo and a 'DOWNLOAD' button; a blue arrow labeled 'Click Here' points to this button. Below this card is the text 'Java Platform (JDK) 8u11'. The second card features the NetBeans logo and a 'DOWNLOAD' button, with the text 'JDK 8u11 & NetBeans 8.0' below it. At the bottom of the page, there is a section titled 'Java Platform, Standard Edition' containing the text 'Java SE 8u11' and a paragraph stating: 'This release includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Learn more ▶'.

Click Here

Overview Downloads Documentation Community Technologies Training

Java SE Downloads

Java Platform (JDK) 8u11

JDK 8u11 & NetBeans 8.0

Java Platform, Standard Edition

Java SE 8u11
This release includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.
[Learn more ▶](#)

Verifying Installation

- ☐ Go to the specified path, where you install your JAVA.
- ☐ You will see a JAVA folder
- ☐ Inside JAVA folder you will see two folders, one is for jdk and other is for jre

JDK

- ☐ **Java Developer Kit** contains tools needed to develop the Java programs
- ☐ These tools could be Compiler (javac.exe), Application Launcher (java.exe), etc

JRE

- ☐ **Java Runtime Environment**
- ☐ It contains **JVM** (Java Virtual Machine) and Java Package Classes (Java Library)

JVM

- ☐ JVM is platform dependent
- ☐ The ***Java Virtual Machine*** provides a platform-independent way of executing code
- ☐ Java Virtual Machine interprets the byte code into the machine code depending upon the underlying operating system and hardware combination.

C++ vs Java

| Comparison Index | C++ | Java |
|---------------------------------|--|---|
| Platform-independent | C++ is platform-dependent. | Java is platform-independent. |
| Mainly used for | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| Design Goal | C++ was designed for systems and applications programming. It was an extension of C programming language . | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a wider audience. |
| Goto | C++ supports goto statement. | Java doesn't support goto statement. |
| Multiple inheritance | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java . |
| Operator Overloading | C++ supports operator overloading . | Java doesn't support operator overloading. |
| Pointers | C++ supports pointers . You can write pointer program in C++. | Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java. |
| Compiler and Interpreter | C++ uses compiler only. C++ is compiled and run using compiler which converts source code into machine code so, C++ is platform dependent. | Java uses compiler and interpreter both. Java source code is converted into byte code at compilation time. The interpreter executes this byte code at run time and produces output. Java is interpreted that is why it is platform independent. |

C++ vs Java

| Comparison Index | C++ | Java |
|--|---|--|
| Call by Value and Call by reference | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| Structure and Union | C++ supports structures and unions. | Java doesn't support structures and unions. |
| Thread Support | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in thread support. |
| Documentation comment | C++ doesn't support documentation comment. | Java supports documentation comment (/** ... */) to create documentation for java source code. |
| Virtual Keyword | C++ supports virtual keyword so that we can decide whether or not to override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |
| unsigned right shift >>> | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| Inheritance Tree | C++ creates a new inheritance tree always. | Java uses single inheritance tree always because all classes are the child of Object class in java. Object class is the root of inheritance tree in java. |
| Hardware | C++ is more nearer to hardware. | Java is not so interactive with hardware. |
| Object oriented | C++ is an object-oriented language. But in C language, single root hierarchy is not possible. | Java is also an object-oriented language. But in Java, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object. |

Resolving an error "javac is not recognized as an internal or external command" ?

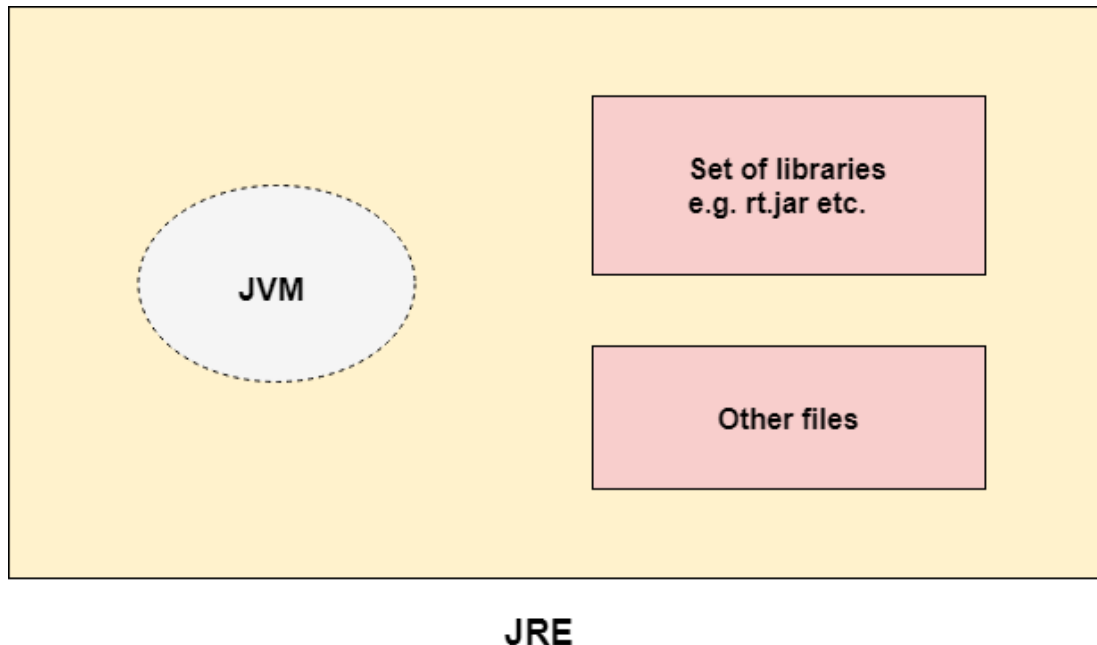
- you need to set path.
- Since DOS doesn't know javac or java, we need to set path.
- Path is not required in such a case if you save your program inside the jdk/bin folder.
-

JVM

- JVM (Java Virtual Machine) is an abstract machine.
- It is called virtual machine because it doesn't physically exist.
- It is a specification that provides runtime environment in which java bytecode can be executed.
- JVMs are available for many hardware and software platforms.
- JVM, JRE and JDK are platform dependent because configuration of each OS are different from each other.
- But, Java is platform independent. The JVM performs following main tasks:
- The JVM performs following operation:
 - Loads code
 - Verifies code
 - Executes code
 - Provides runtime environment

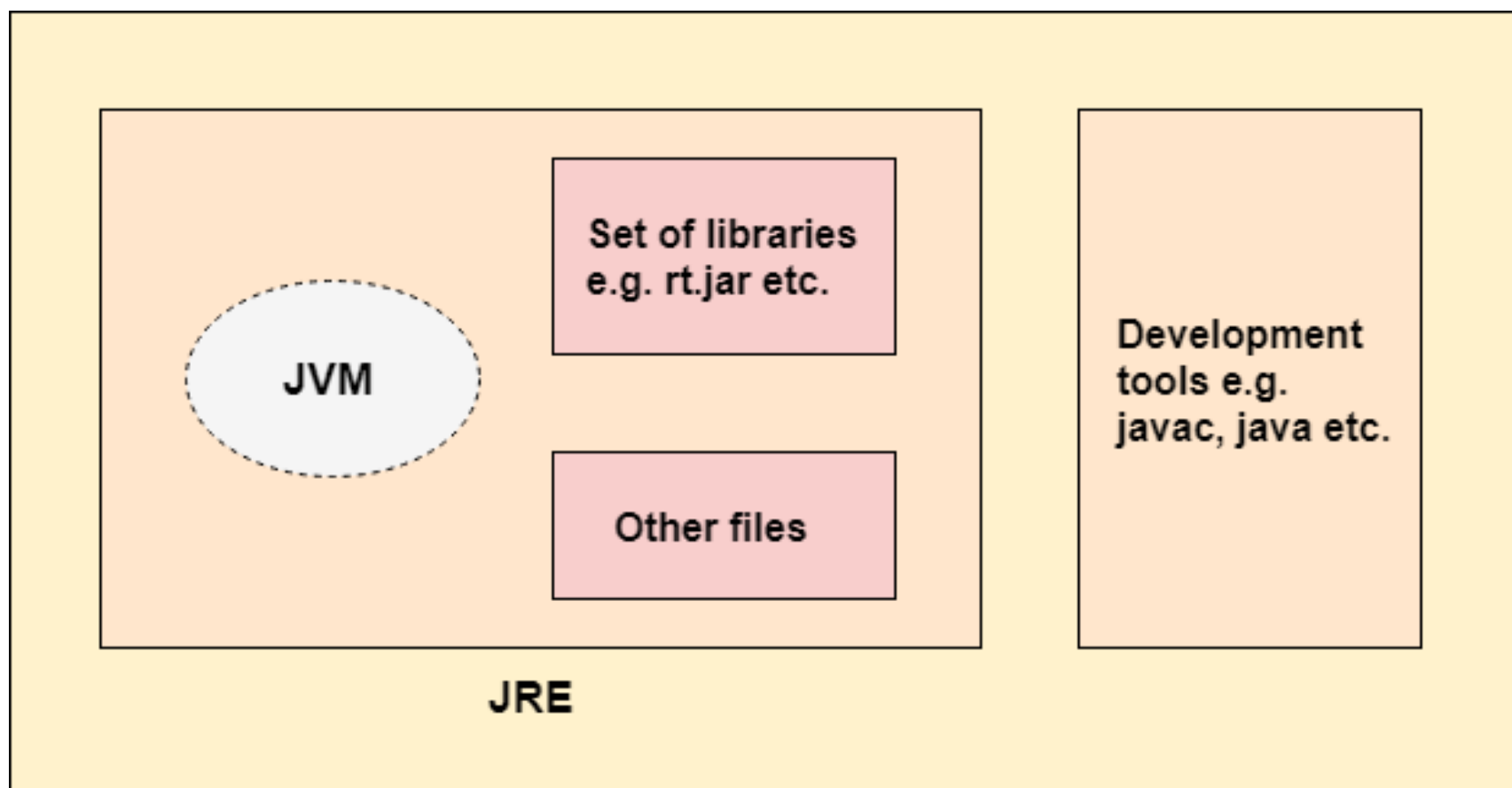
JRE

- JRE is an acronym for Java Runtime Environment.
- The Java Runtime Environment is a set of software tools which are used for developing java applications.
- It is used to provide runtime environment.
- It is the implementation of JVM.
- It physically exists.
- It contains set of libraries + other files that JVM uses at runtime.



JDK

- The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets.
- It physically exists. It contains JRE + development tools.
- JDK is an implementation of any one of the below given Java Platforms released by Oracle corporation:
 - Standard Edition Java Platform
 - Enterprise Edition Java Platform
 - Micro Edition Java Platform
- The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) etc. to complete the development of a Java Application.



JDK

Types of Variables

There are three types of variables in java:

- 1) Local variable** - A variable declared inside the body of the method is called local variable.
 - You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
 - A local variable cannot be defined with "static" keyword.
- 2) Instance variable** - A variable declared inside the class but outside the body of the method, is called instance variable.
 - Created when object of the class are instantiated, so they are associated with the objects and take different values for each object.
 - It is not declared as static.
 - It is called instance variable because its value is instance specific and is not shared among instances.
- 3) Static variable** - A variable which is declared as static is called static variable.
 - It cannot be local.
 - You can create a single copy of static variable and share among all the instances of the class.
 - Memory allocation for static variable happens only once when the class is loaded in the memory.

Types of Variables

```
class A
```

```
{
```

```
int data=50;    //instance variable
```

```
static int m=100;    //static variable
```

```
void method()
```

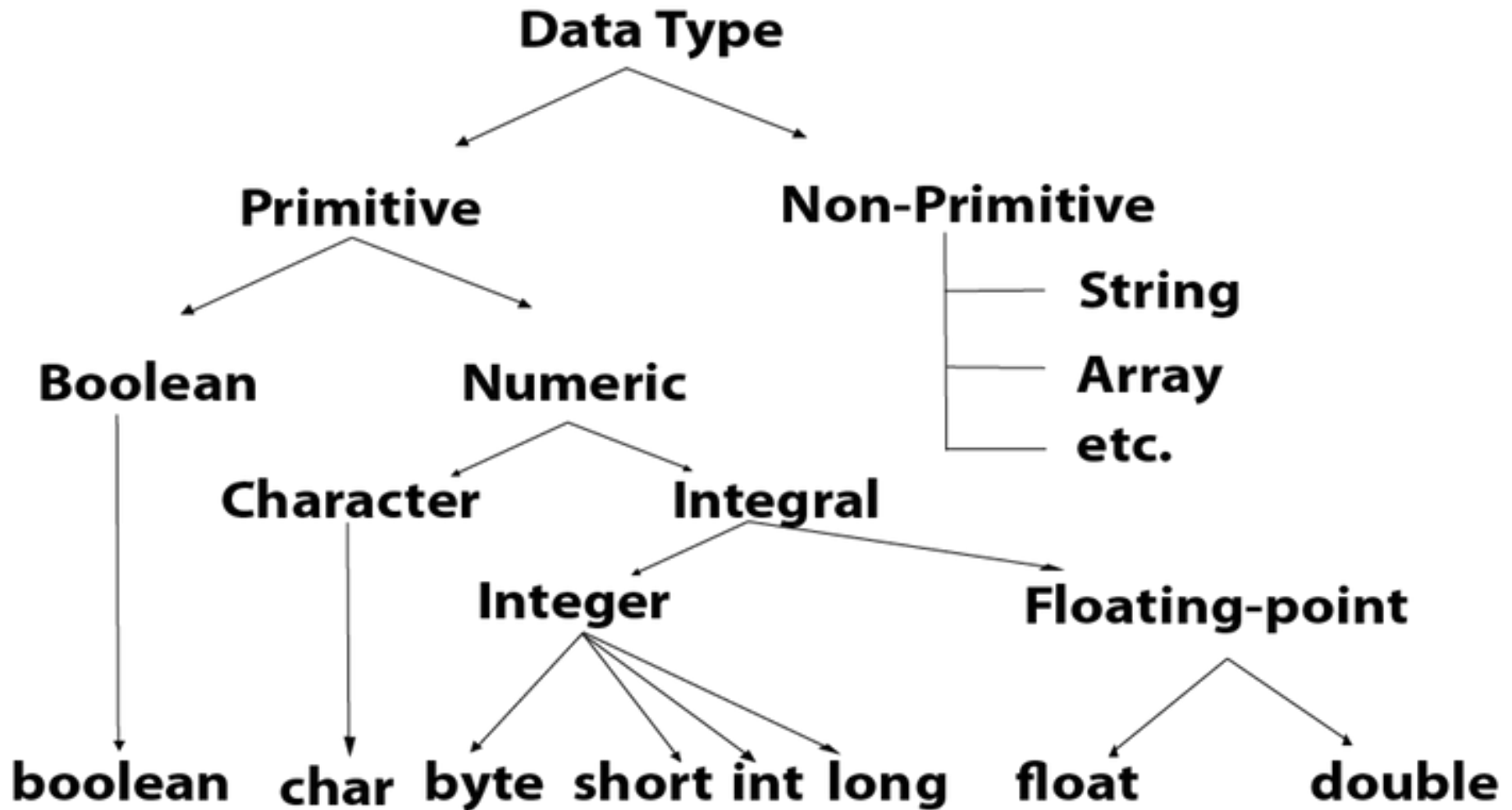
```
{
```

```
int n=90;        //local variable
```

```
}
```

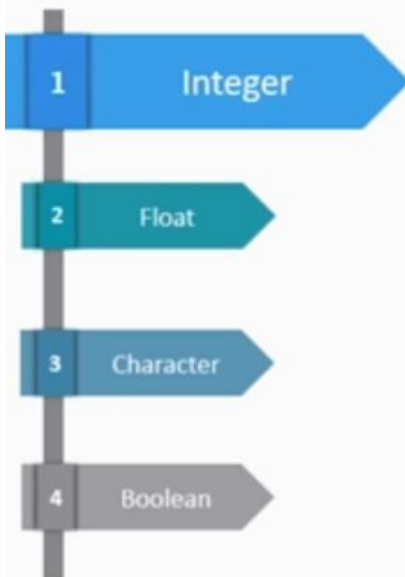
```
}                //end of class
```

Data Types



Numeric Data Type

- ☐ Integer data types are used to store numeric values.
- ☐ There are four different integer types in Java:



| | |
|------------|---|
| byte Type | -128 to 127 |
| short Type | -32768 to 32767 |
| int Type | -2147483648 to 2147483647 |
| long Type | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |

Float Data Type

- ☐ Float data types are used to store numeric values along with their decimal value.
- ☐ There are two different float types in Java:

1

Integer

2

Float

3

Character

4

Boolean

float Type



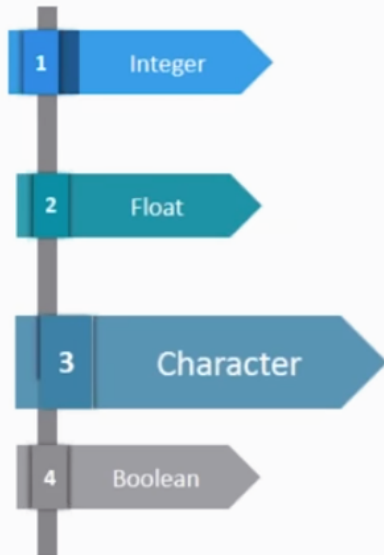
$\pm 3.40282347\text{E}+38\text{F}$

double Type

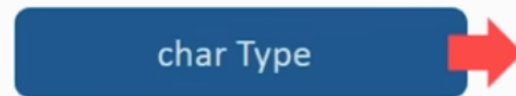


$\pm 1.79769313486231570\text{E}+308$

Character Data Type



- ☐ Character data types are used to store character values.
- ☐ char data type is a single 16-bit Unicode character:



0 to 65,535

Boolean Data Type

- ☐ Boolean data types are used to store boolean values along.
- ☐ This data type is used for simple flags that track true/false conditions

1

Integer

2

Float

3

Character

4

Boolean

boolean Type



True or false


| Data Type | Default size |
|-----------|--------------|
| boolean | 1 bit |
| char | 2 byte |
| byte | 1 byte |
| short | 2 byte |
| int | 4 byte |
| long | 8 byte |
| float | 4 byte |
| double | 8 byte |

Type Casting


Definition: Assigning a value of one type to a variable of another type is known as **Type Casting**.

In Java, type casting is classified into **two** types,

- **Widening Casting (Implicit)**

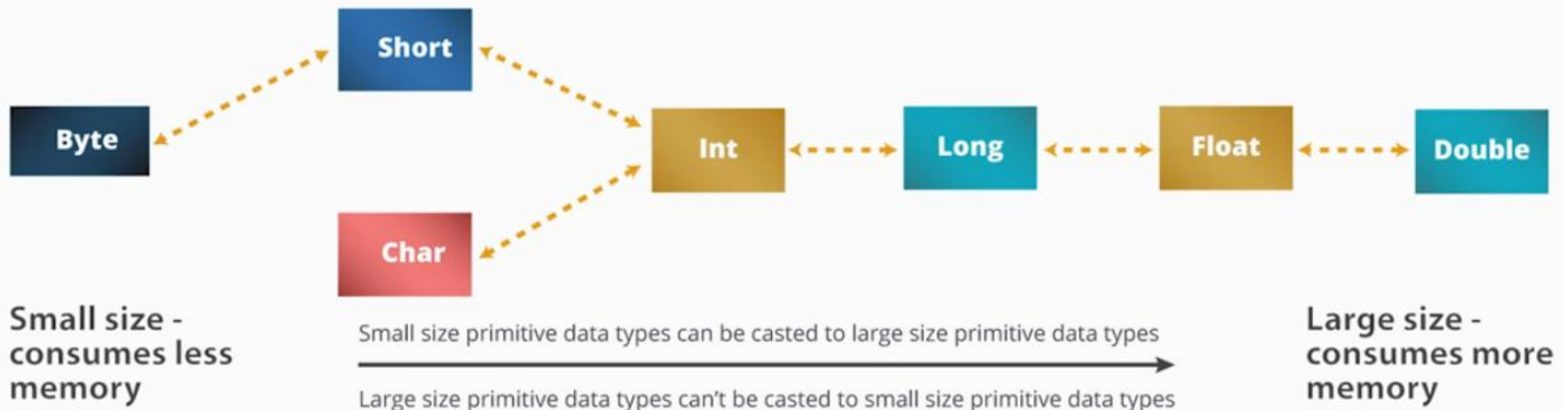
byte → short → int → long → float → double

widening

- **Narrowing Casting (Explicitly done)**

double → float → long → int → short → byte

Narrowing

Type Promotions

Implicit promotion / casting of primitive data types in java



Widening Conversion

Widening conversion is allowed in the following cases:

byte can be converted to short, int, long, float, or double

float can be converted to double

Short can be converted to int, long, float, or double

long can be converted to float or double



char can be converted to int, long, float, or double

int can be converted to long, float, or double

JAVA is 99% OOP

- ❑ Java is an object-oriented language and as said everything in java is an object.



But what
about the
primitives?

- ❑ They are sort of left out in the world of Objects, that is, they cannot participate in the object activities

Wrapper Classes

- ❑ As a solution to this problem, Java allows you to include the primitives in the family of objects by using what are called **wrapper classes**.
- ❑ There is a wrapper class for every primitive data type in Java.

Wrapper Classes...

- ❑ This class encapsulates a single value for the primitive data type
- ❑ For instance the wrapper class for int is Integer, for float is Float, and so on

Primitive type → Wrapper Class

| | | |
|----------------|---|-----------|
| <u>boolean</u> | → | Boolean |
| byte | → | Byte |
| char | → | Character |
| short | → | Short |
| <u>int</u> | → | Integer |
| long | → | Long |
| float | → | Float |
| double | → | Double |

Useful methods of wrapper class

☐ valueOf()

- Static method.
- Return Object reference of relative wrapper class

☐ parseXxx()

- Static method
- Xxx can be replaced by any primitive type
- It returns xxx type value

☐ xxxValue()

- Instance method of wrapper class
- Xxx can be replaced by any primitive type
- Returns corresponding primitive type

Operators in java

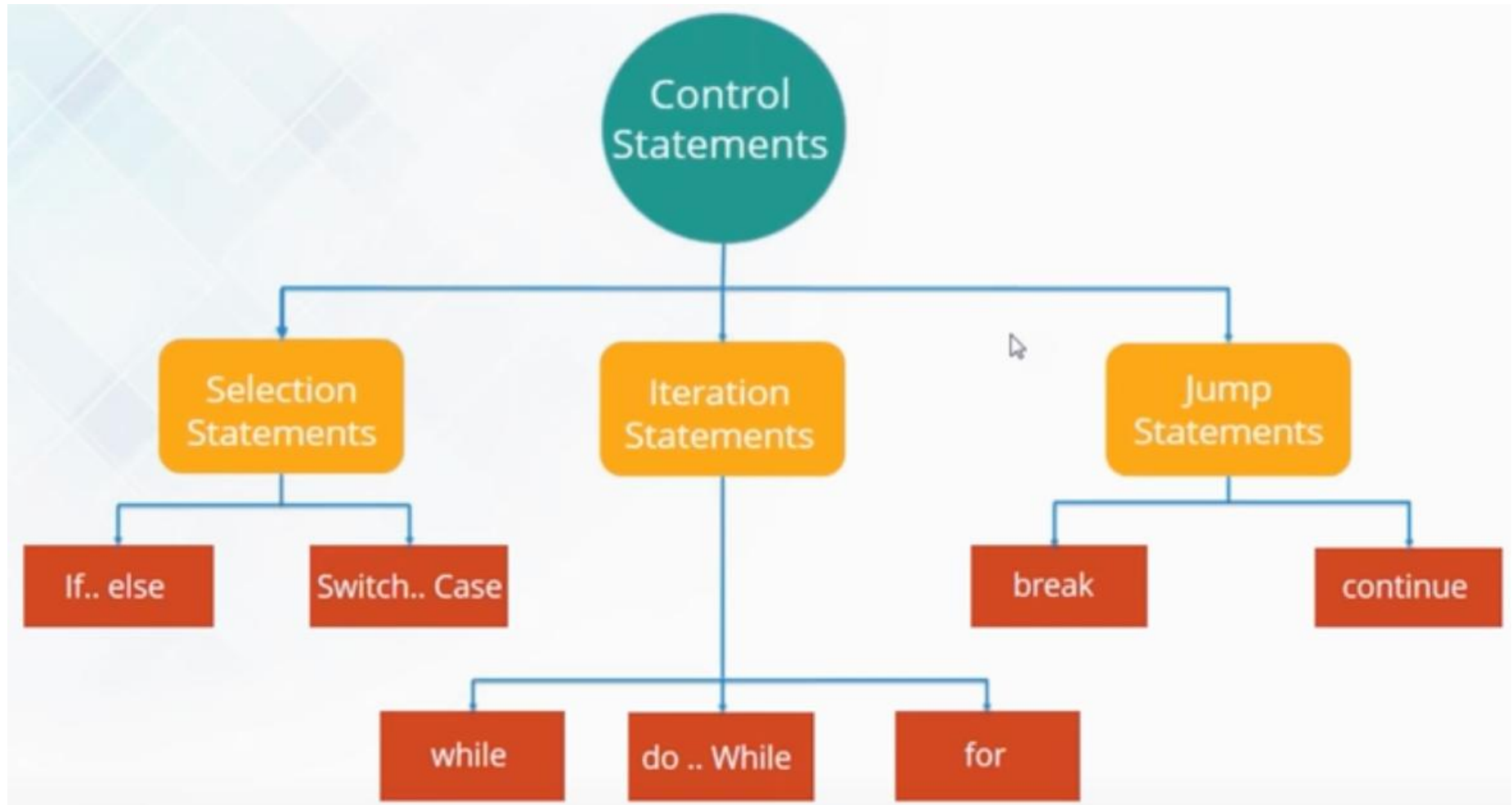
Operator in java is a symbol that is used to perform operations.

For example: +, -, *, / etc.

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

- Unary Operator : ++ , --
- Arithmetic Operator: +, -, *, /, %
- Shift Operator: << ,>> ,>>>
- Relational Operator: == ,!=, <, >, <=,>=
- Bitwise Operator: bitwise AND (&), bitwise exclusive OR(^), bitwise inclusive OR (|)
- Logical Operator: &&, ||
- Ternary Operator: ? :
- Assignment Operator: =, +=, -=, *=, /= etc.

Control Statements in java



1. If statement

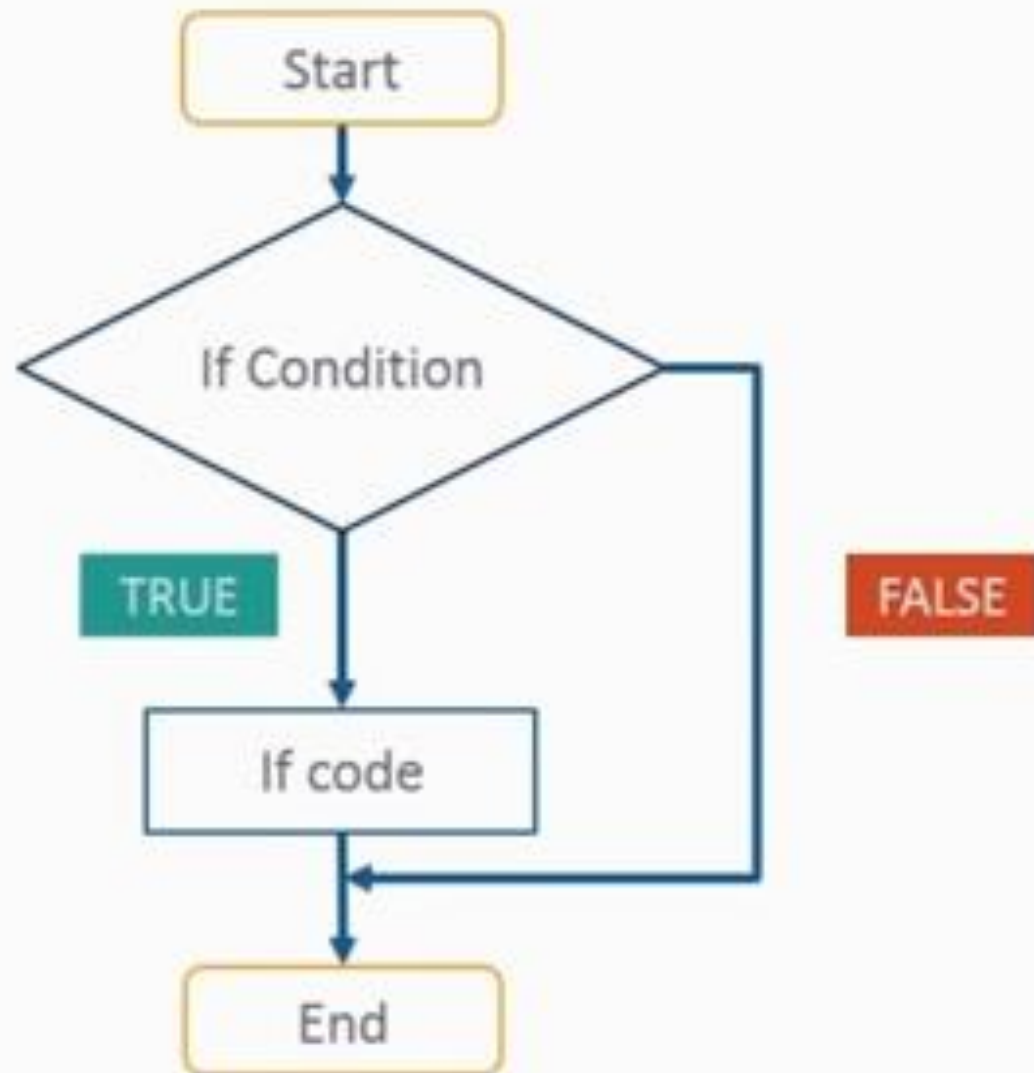
Syntax:

if (condition):

statements 1 ...

else:

statements 2 ...

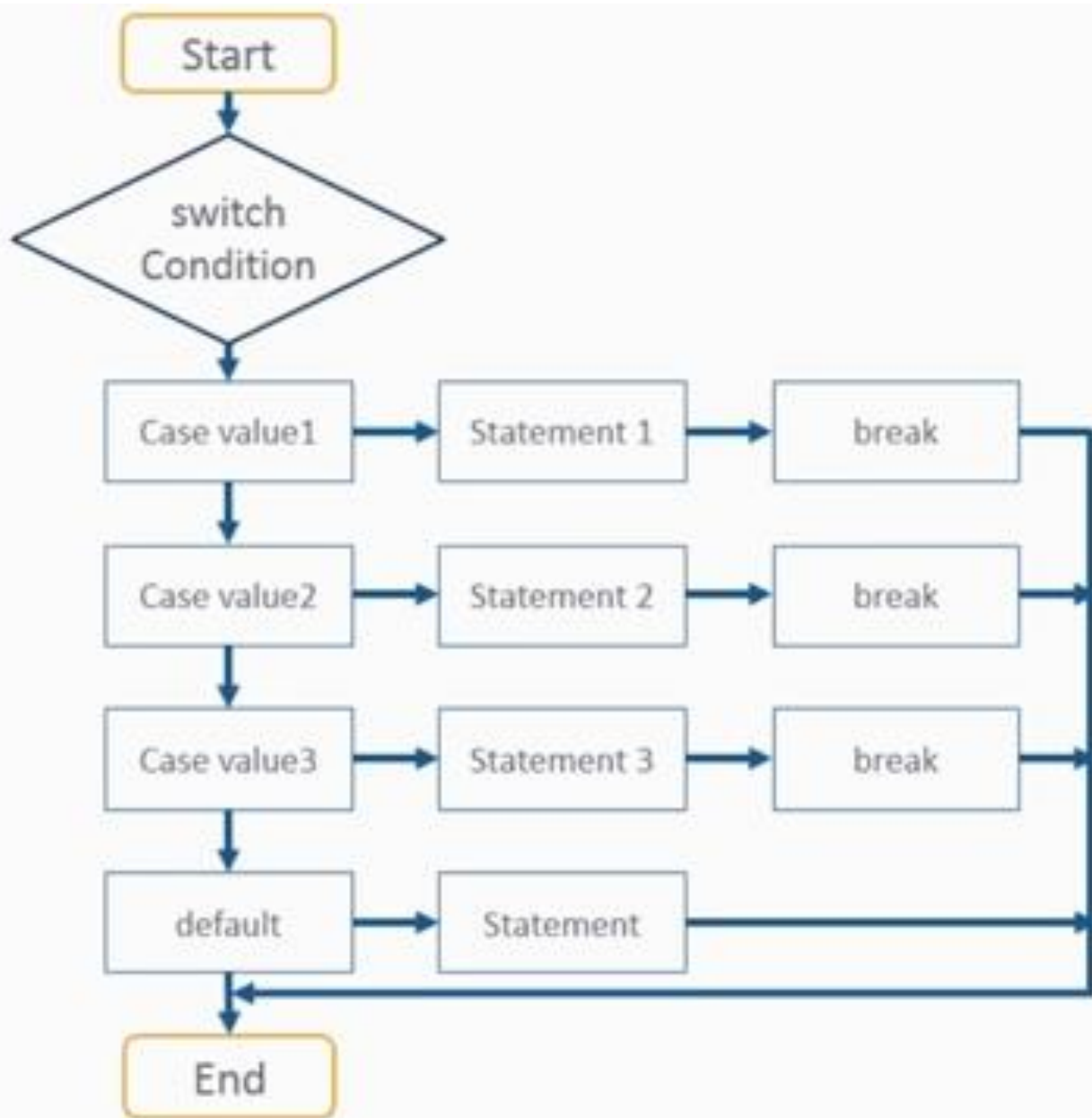


```
public class If_else {  
    public static void main(String args[]) {  
        int num = 25000;  
        if (num < 100 && num >= 10) {  
            System.out.println("Its a two digit number");  
        } else if (num < 1000 && num >= 100) {  
            System.out.println("Its a three digit number");  
        } else if (num < 10000 && num >= 1000) {  
            System.out.println("Its a four digit number");  
        } else if (num < 100000 && num >= 10000) {  
            System.out.println("Its a five digit number");  
        } else {  
            System.out.println("number is not between 1 & 99999");  
        }  
    }  
}
```

Switch case Statement

Syntax:

```
1  switch (expression)
2  {
3      case Value1: //Statement 1
4                  break;
5      case Value2: //Statement 2
6                  break;
7      case Value3: //Statement 3
8                  break;
9      default: //default statement
10 }
```



```
public class conditional {  
    public static void main(String[] args) {  
        int Day = 4;  
        switch (Day) {  
            case 1:   
                System.out.println("Sunday");  
                break;  
            case 2:  
                System.out.println("Monday");  
                break;  
            case 3:  
                System.out.println("Tuesday");  
                break;  
            case 4:  
                System.out.println("Wednesday");  
                break;  
            case 5:  
                System.out.println("Thursday");  
                break;  
            case 6:  
                System.out.println("Friday");  
                break;  
            case 7:  
                System.out.println("Saturday");  
                break;  
            default:  
                System.out.println("Invalid day");  
                break;  
        }  
    }  
}
```

LOOPS REPEAT ACTIONS

SO YOU DON'T HAVE TO ...

Execute once and then
repeats things until loop
condition is true

Do While

While

Repeat things until the
loop condition is true

Repeat things till the
given number of times

For

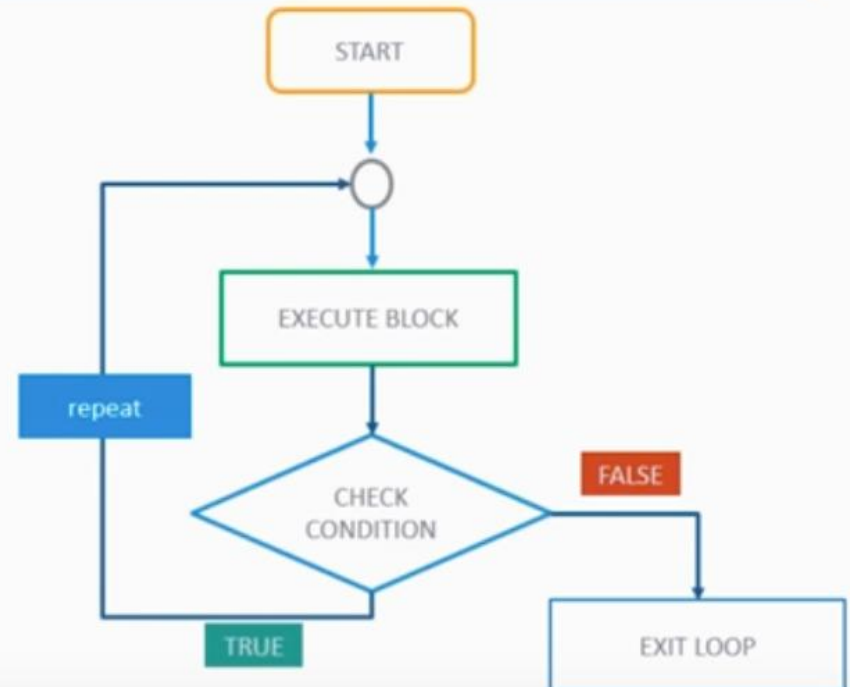
Do While Loop

Do While Loop

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition after executing the loop body.

Syntax:

```
1 do
2 {
3     //Statements
4 } while (condition);
5
```



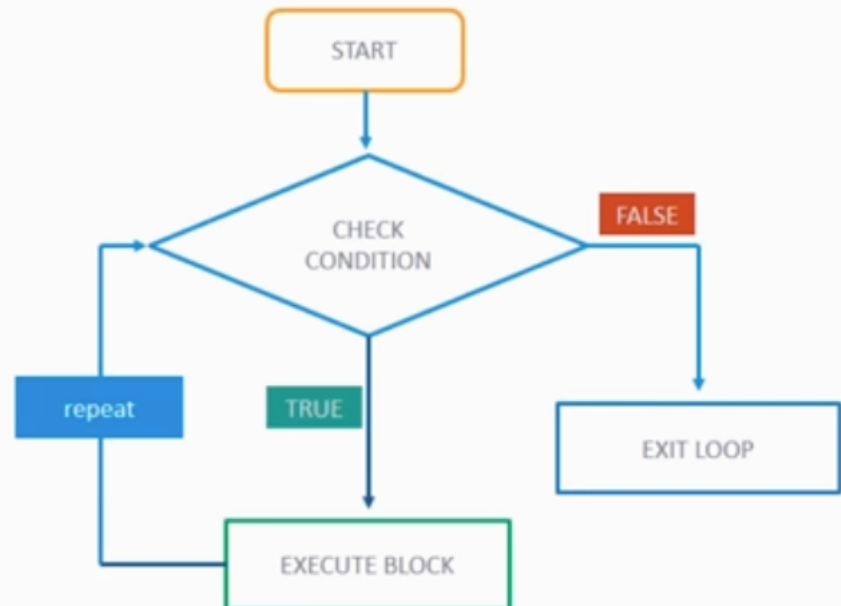
While Loop

While Loop

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

Syntax:

```
1 while (condition)
2 {
3     //Statements
4 }
5
```



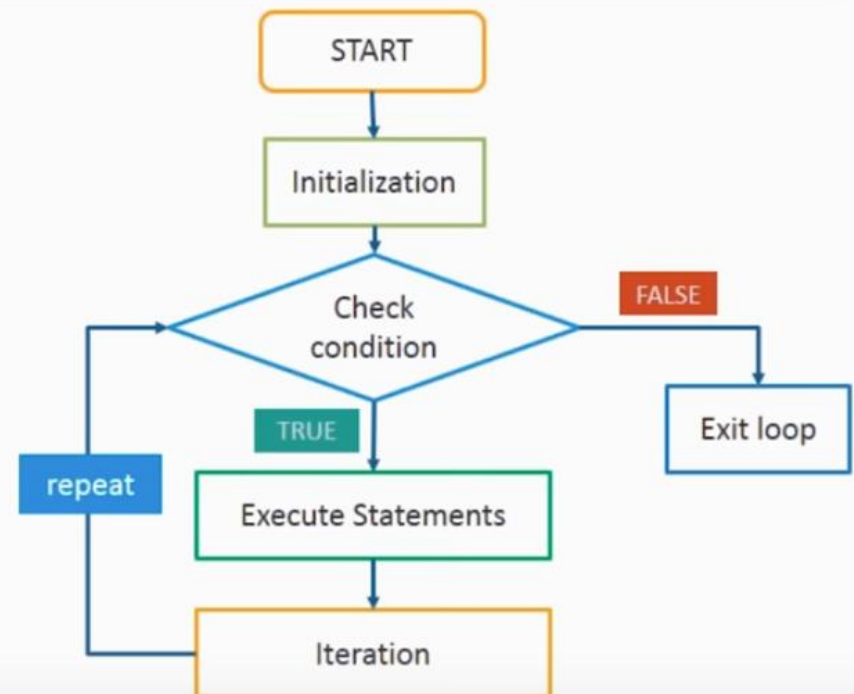
For Loop

For Loop

Repeats a statement or group of for a fixed number of times. It tests the condition before executing the loop body.

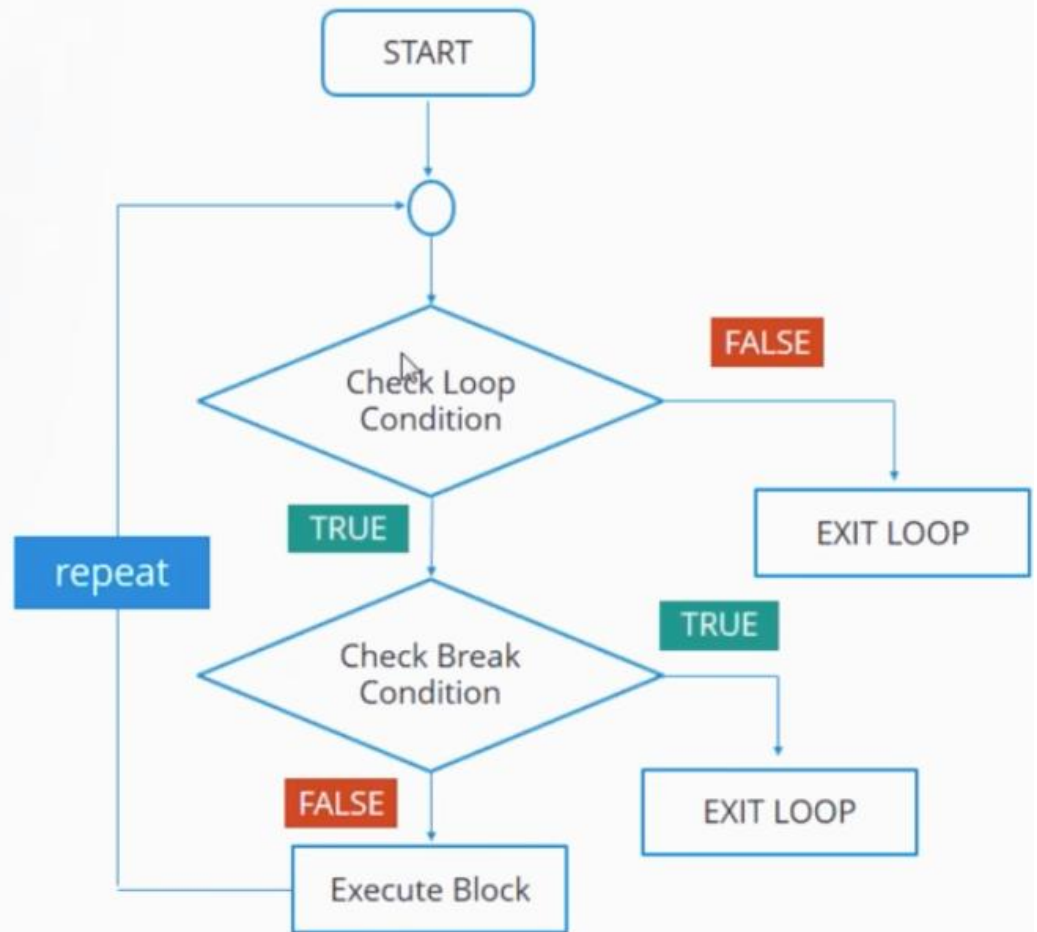
Syntax:

```
1 for(initialization;condition;iteration)
2 {
3     //Statements
4 }
5
```



6. break

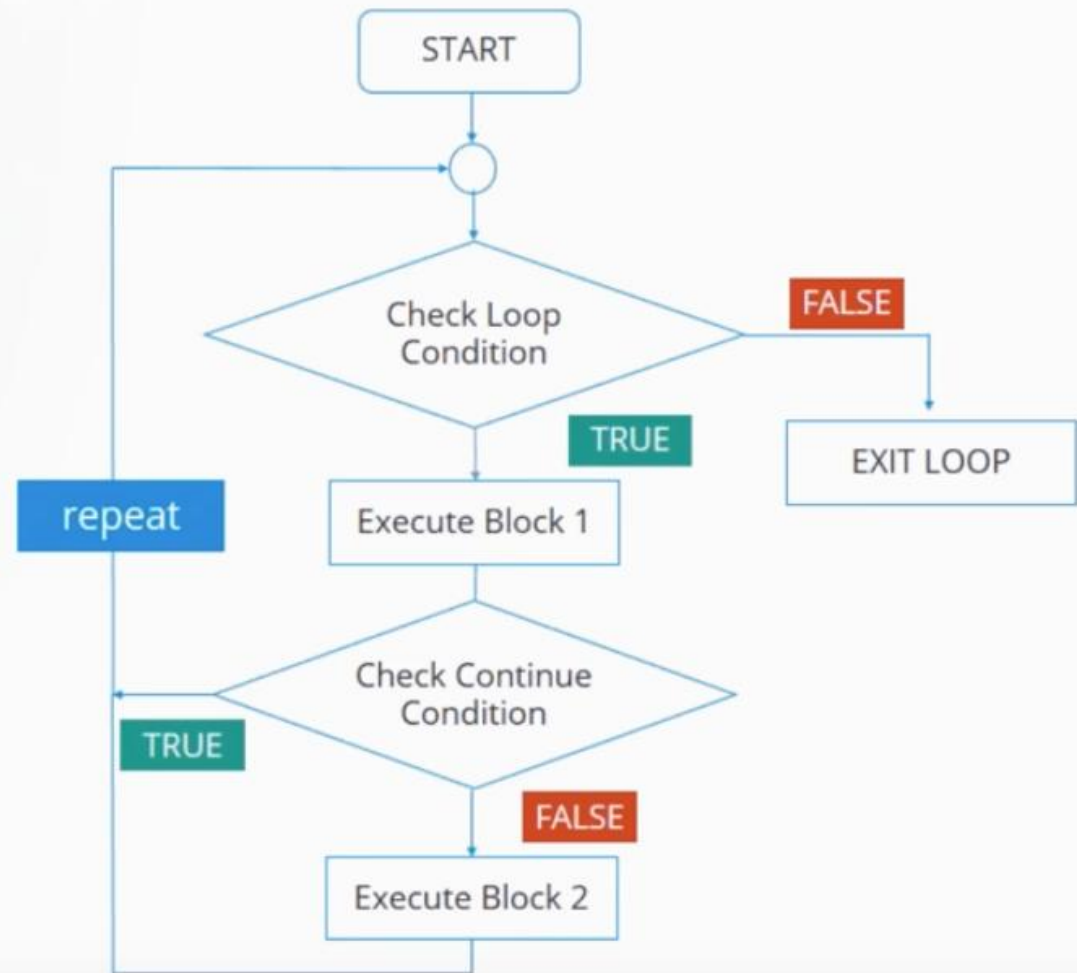
Syntax:
`break;`



7. continue

Syntax:

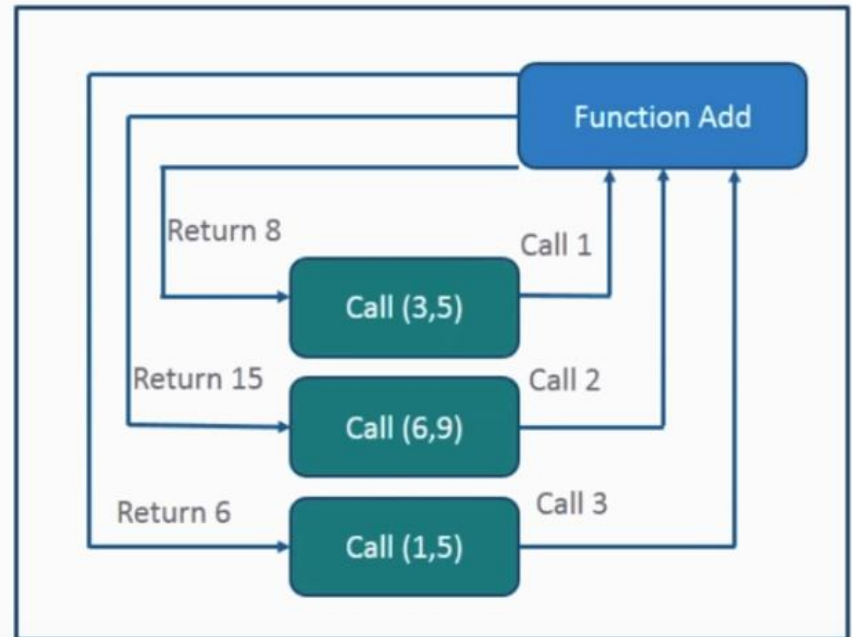
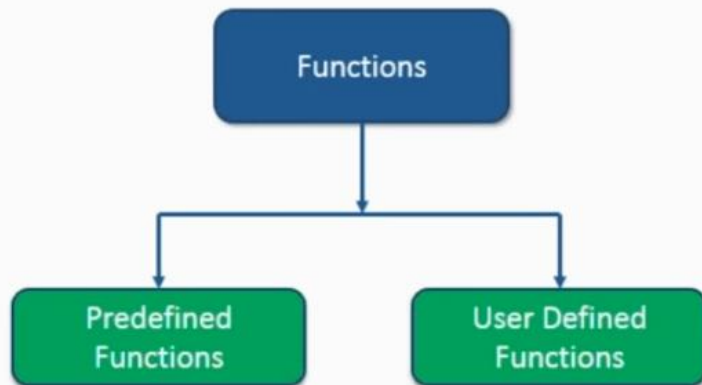
```
continue;
```



Functions In Java

Functions

A function is a block of organized, reusable code that is used to perform a single, related action.



```
1 class fibo{
2     static int n1=0,n2=1,n3=0;
3     static void printFibo(int count){
4         if(count>0){
5             n3 = n1 + n2;
6             n1 = n2;
7             n2 = n3;
8             System.out.print(" "+n3);
9             printFibo(count-1);
10        }
11    }
12    public static void main(String args[]){
13        int count=25;
14        System.out.print(n1+" "+n2);
15        printFibo(count-2);
16    }
17 }
```

Problems Javadoc Declaration Console

<terminated> fibo [Java Application] C:\Program Files\Java\jdk-9.0.4\bin\javaw.exe (27-Feb-2018, 7:32:18 PM)

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368

Marry is a school teacher. She conducted the final exam for the year. The exam included the following subject:

| | | | | |
|-------------|-----------|---------|---------|------------------|
| Mathematics | Chemistry | Physics | English | Computer Science |
|-------------|-----------|---------|---------|------------------|

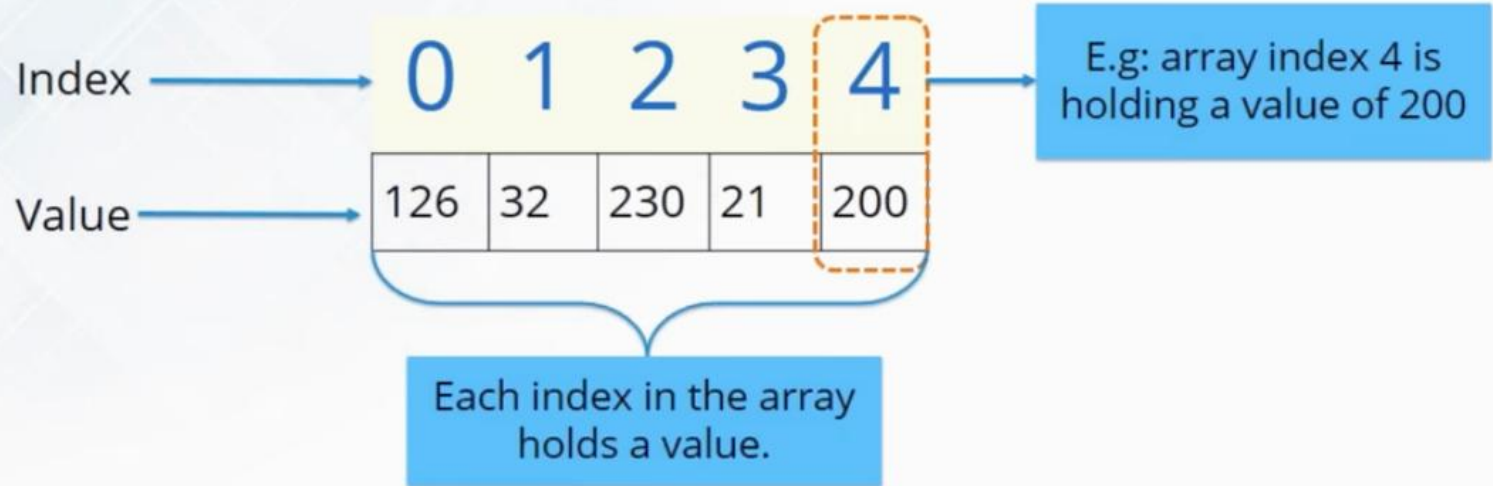
Since there are multiple students, she wanted to create an automated system that will take the marks for these subjects and provide a grade for the final score. Below is the grade distribution :

| Percentage | Grade |
|------------|-----------|
| Below 40 | Poor |
| 40-59 | Average |
| 60-79 | Good |
| 80-89 | Very Good |
| 90+ | Excellent |



Array in Java

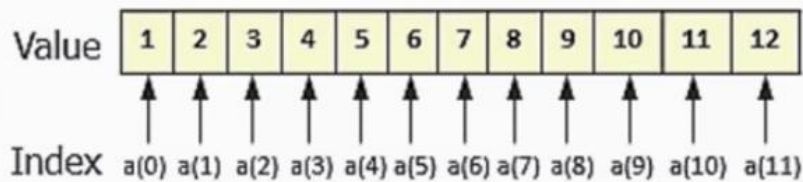
- ❑ An array is a data structure which holds the sequential elements of the same type.
- ❑ Arrays of any type can be created and may have one or more dimensions.



❑ Arrays can be of 2 types:

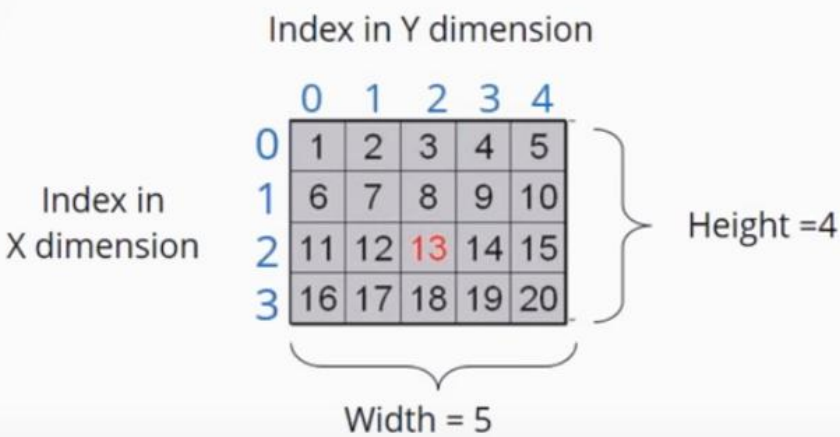
Single Dimensional Array

Initialization: `int a[] = new int[12]`



Multi Dimensional Array

Initialization: `int table[][] = new int[4][5];`



Stages

- Declaration
- Construction
- Initialization

- Declaring Arrays:

```
int[] marks;
```

```
byte[] age;
```

Less readable:

```
int marks[];
```

```
byte age[];
```

Construction

```
int[] marks;
```

```
marks = new int[5];
```

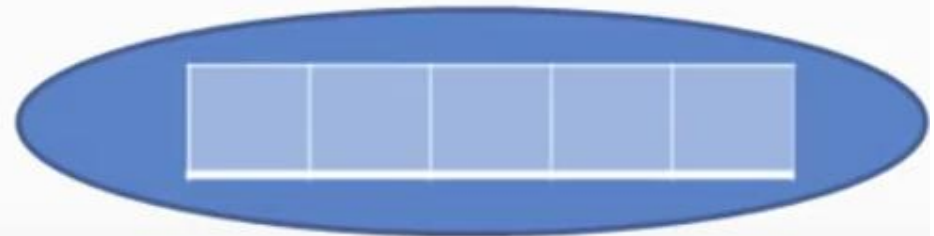
//Every array is an object hence the **new** keyword

“The size of the array is mandatory”

In single line:

```
int[] marks = new int[5];
```

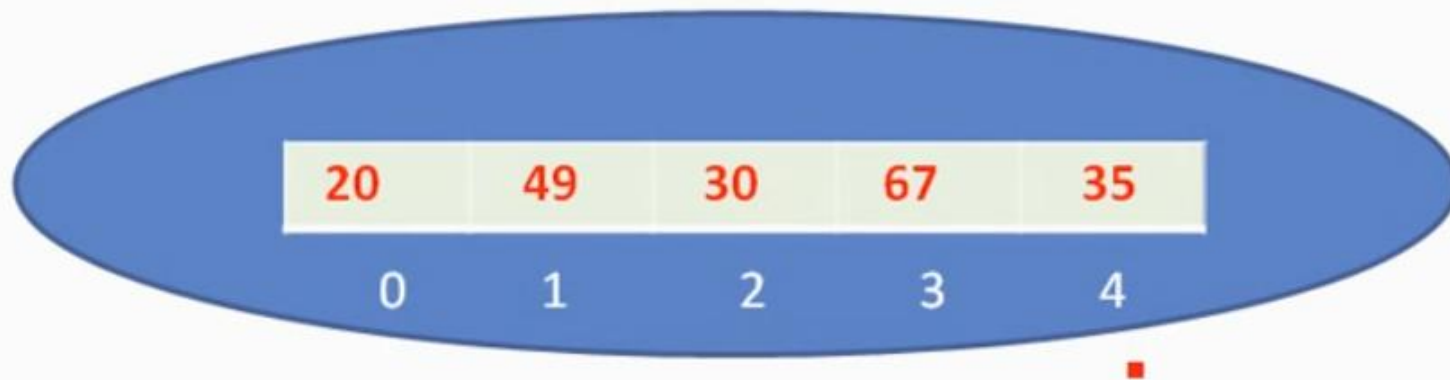
marks



Initialization

- Initialization is loading the array with the values.

```
int[] marks = new int[5];
```



```
marks[0] = 20;  
marks[1] = 49;  
marks[2] = 30;  
marks[3] = 67;  
marks[4] = 35;
```

```
class Pen{ }
```

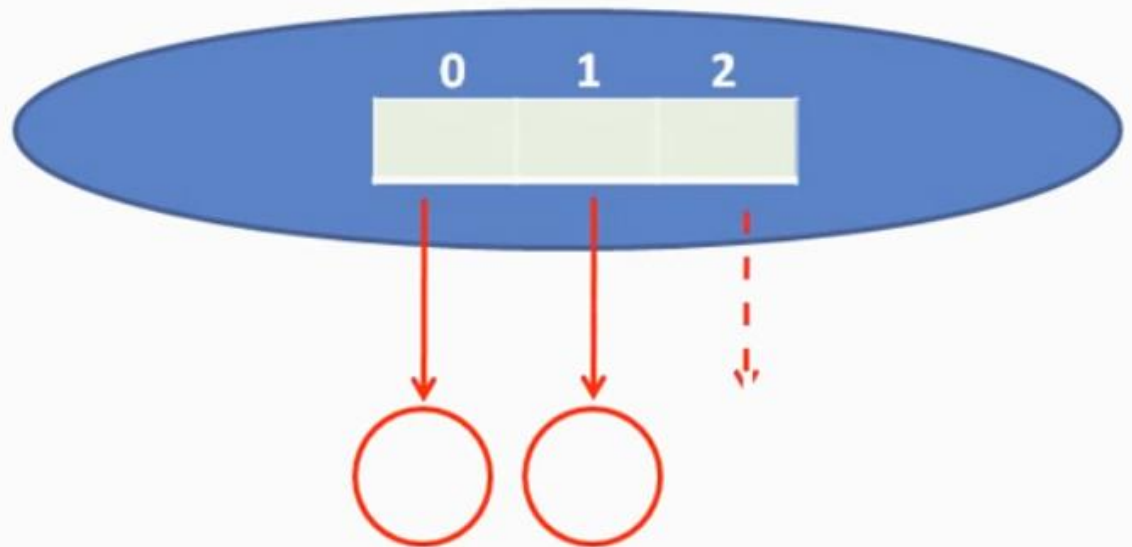
```
Pen[] pens = new Pen[3];
```

```
pens[0] = new Pen();
```

```
pens[1] = new Pen();
```

```
pens[2] = new Pen();
```

```
pens
```



With the array of objects type the array index stores the memory locations of objects where as in case of primitive data types the array stores values.

Two dimensional Array

- `int[][] marks = new int[2][3];`

`marks[0][0] = 30;`

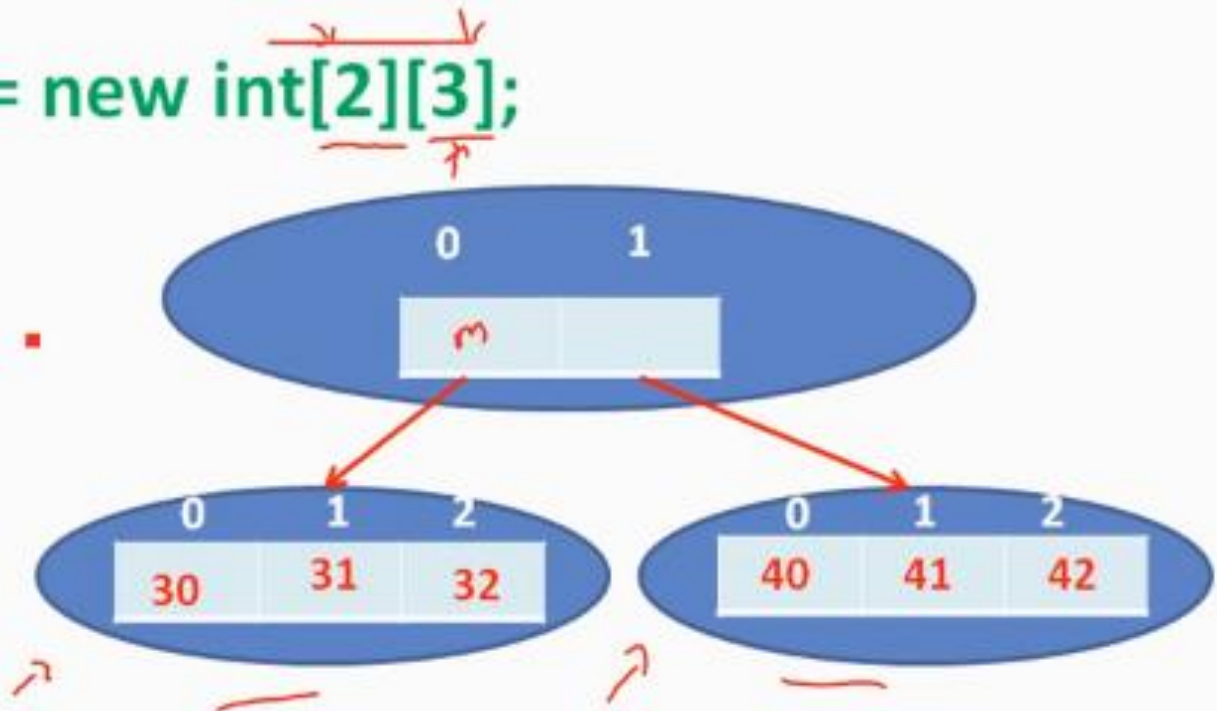
`marks[0][1]=31;`

`marks[0][2]=32;`

`marks[1][0]=40;`

`marks[1][1]=41;`

`marks[1][2]=42;`



Multidimensional Arrays

```
int[][][] marks = new int[2][3][2];
```

