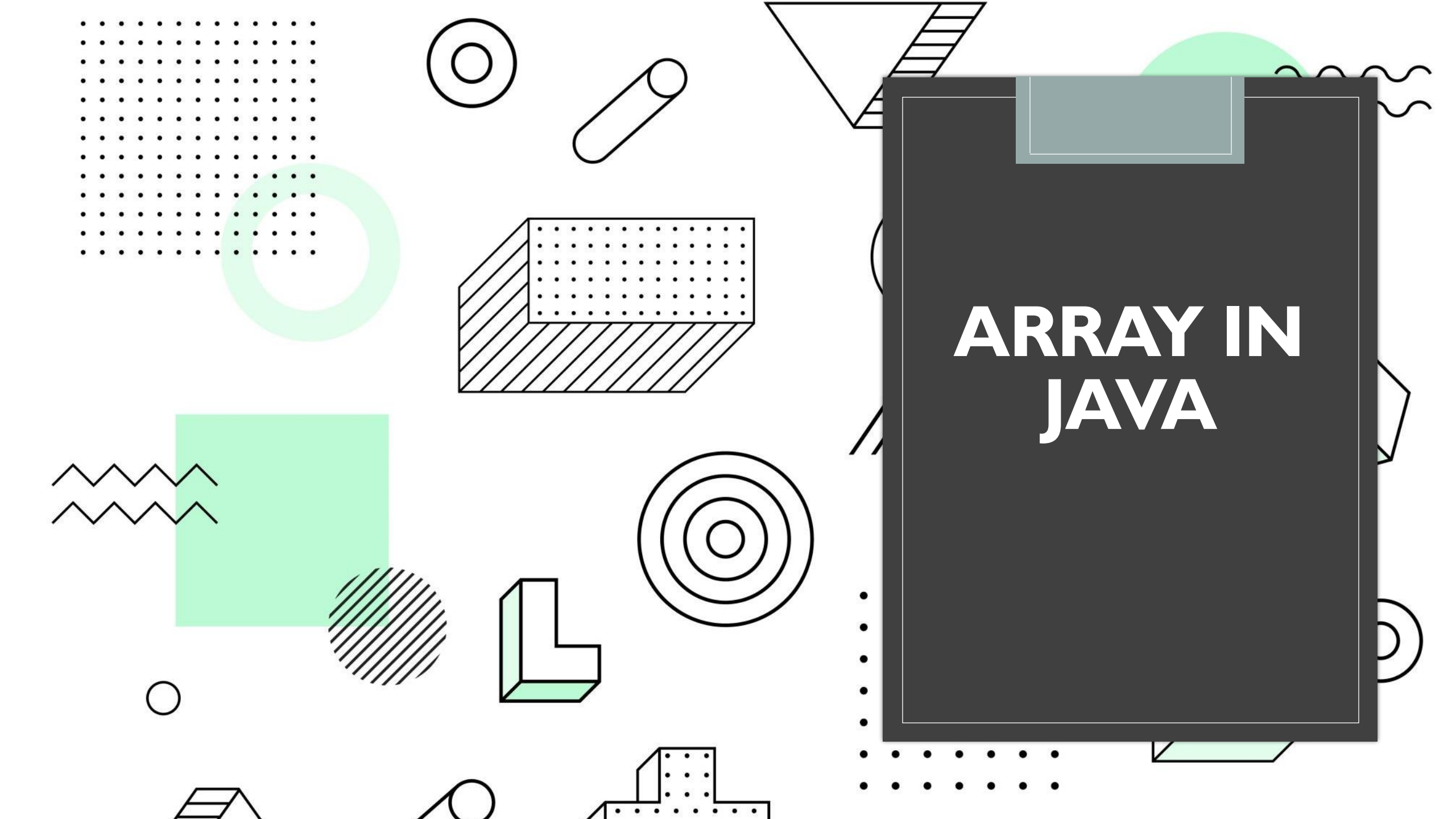
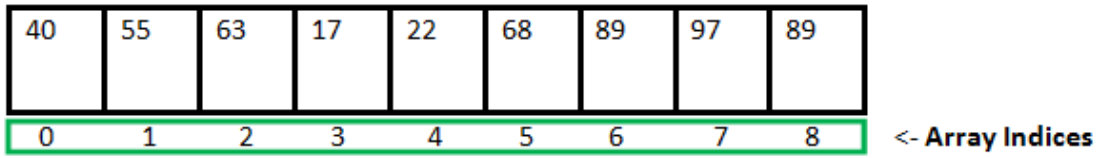

ARRAY IN
JAVA

What is Array?

Array is a collection/group of similar data type variables that are referred by common name.

An Array is an indexed collection of fixed number of homogeneous data elements.

In Java array are implemented as objects.



Array Length = 9
First Index = 0
Last Index = 8

- As array are objects, we can find out length of array using member attribute **length [no of element an array can hold].**
- Size of an array can be specified in integer value.
- Direct super class of an array is **Object.**
- In Java all array are dynamically allocated.

Advantages

We can access array elements randomly using indexes value.

We can store multiple values of same type under single name.

we can use it to implement other data structure like stack,queue,graphs..

Disadvantages

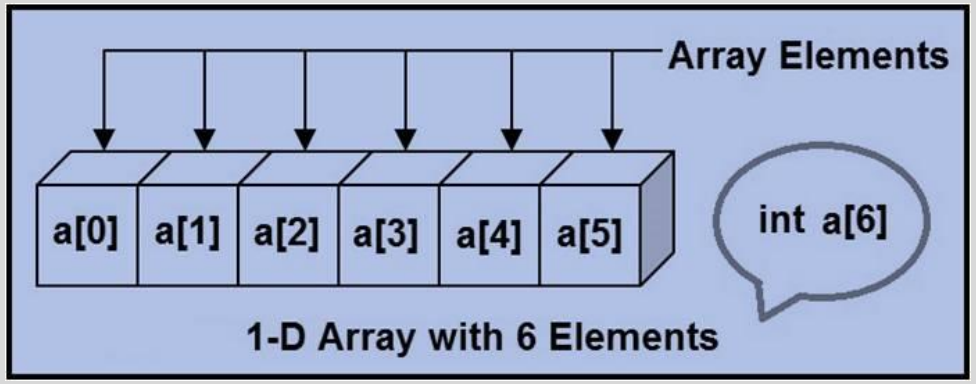
As array size is fixed wastage of memory is possible if we do store less elements than specified size.

Process of insertion and deletion of an element on specific position is time consuming and complex.

It is not possible to increase size of an array once it is declared..

Types of Array

- **Single Dimensional Array**



- **Multi-Dimensional Array**

	Column 0	Column 1	Column 2
Row 0	X[0][0]	X[0][1]	X[0][2]
Row 1	X[1][0]	X[1][1]	X[1][2]
Row 2	X[2][0]	X[2][1]	X[2][2]

2D-Array

Single Dimension Array Declaration

Declaration Syntax `type var_name[];`

Example `int days[];`

here , **days** is an array of int.

this statement indicates declaration only no actual array exist.

to link **days** with actual physical array of integers we must allocated memory using **new** and assign it to **days.**

Array Creation Syntax

var_name = new type[Size]; **days = new int[7];**

type var_name = new type[size]; **int months = new int[12];**

here , var_name name of array variable and size specifies no of elements can be stored in array.

Elements in array allocated by new will automatically initialized to zero(numeric) ,false (boolean),null(reference type).

Single Dimension Array Creation

Notes:

Obtaining array is a two step process

declaring an array of required type

allocate memory that will hold an array using new keyword and assign it to array variable.

thus , In java array are dynamically allocated.

we can not specify size of an array as negative value. [it will generate runtime exception]

Multi-Dimensional Array

It is an array of arrays

Data in multi dimensional array are stored in tabular form.

Total no of elements that can be stored in multi dimensional array can be calculated by multiplying size of all dimensions.

Declaration Syntax

type var_name[][];

Example

int A[][];

Two Dimension Array Declaration

Creation of Array

var_name = new type[size1][size2];

Example

A = new int [3][4];

When we allocate memory to multi dimensional you need to specify memory for the first left most dimensional only.

you can allocate memory for remaining dimension separately.

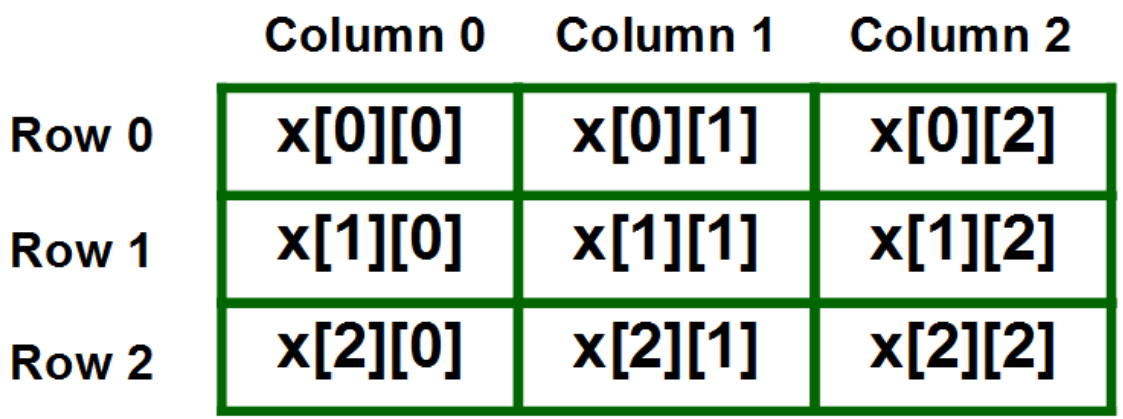
Multi-Dimension Array Creation

Example

- int b[][] = new int[3][];
- b[0] = new int[5];
- b[1] = new int[5];
- b[2]=new int [5];
- Two dimensional array can be initialized directly also.
- Example :
- int c[2][3]={ { 1,3,2} ,{ 4,5,6} };

Multi-Dimension Array Creation

TYPES OF 2-D ARRAY



RECTANGULAR ARRAY

- Rectangular array is an array of arrays where members of the each array must be of same size.
- No of columns for each row are same.
- **int a[][] = new [3][3];**

Non-Rectangular Array OR Jagged Array

Non-rectangular array is an array of arrays where members of the arrays can be of different size.

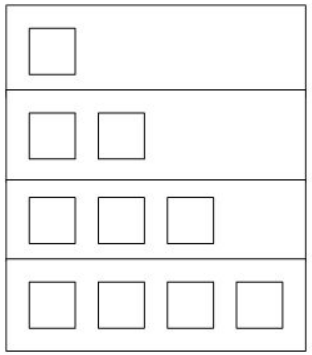
We can create 2-D array with variable no of columns in each row.

Such type of array is also called as jagged array.

Non-Rectangular Arrays

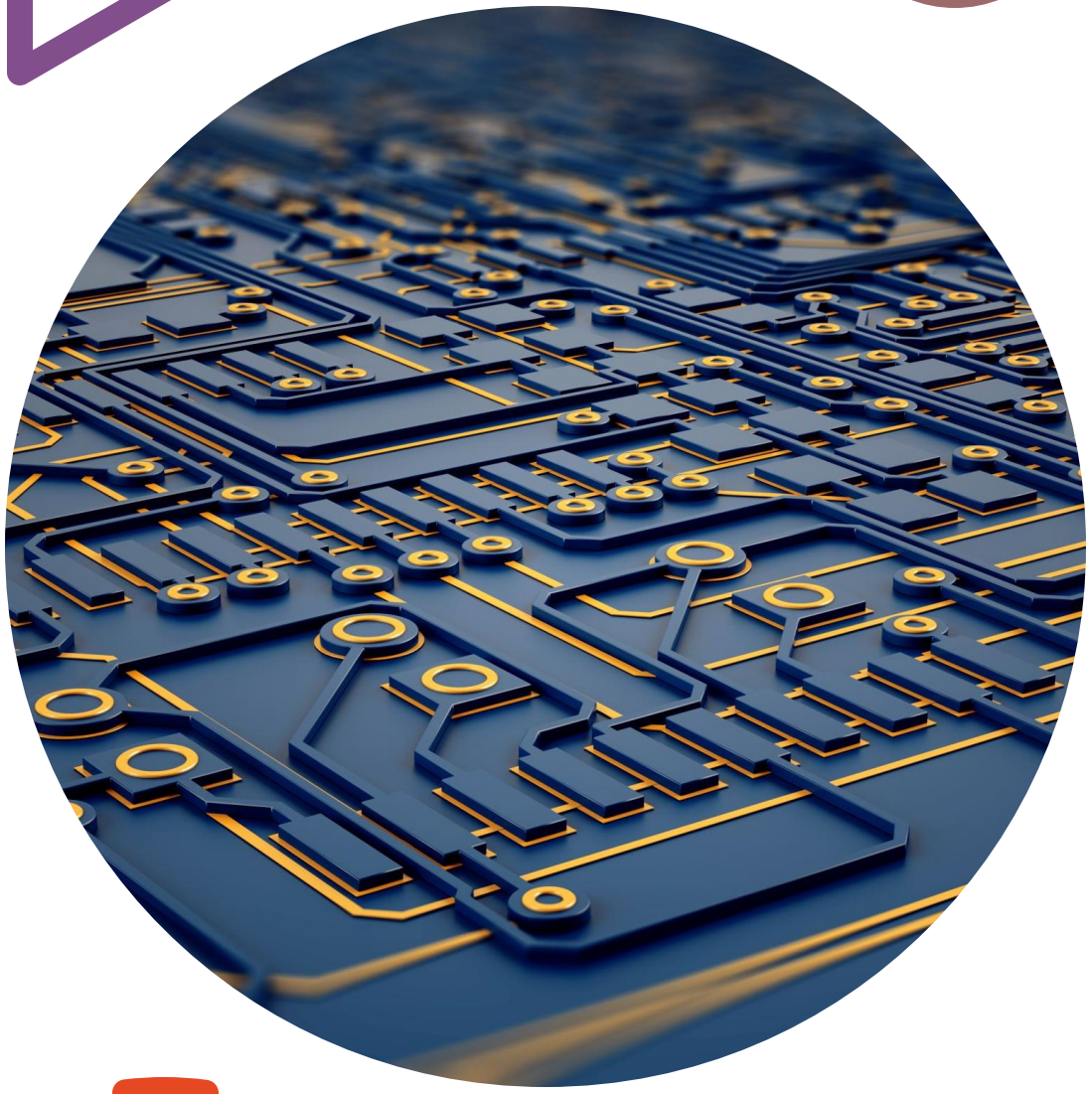
- Need not allocate subarrays of same size

```
int[][] nonrect = new int[4][];
nonrect[0] = new int[1];
nonrect[1] = new int[2];
nonrect[2] = new int[3];
nonrect[3] = new int[4];
```



- Non-rectangular array can be used very effectively in a situation where we want to stores variable no of columns in each rows.

Fundamentals of Java Programming



Data Types

Every variable in Java has a datatype.

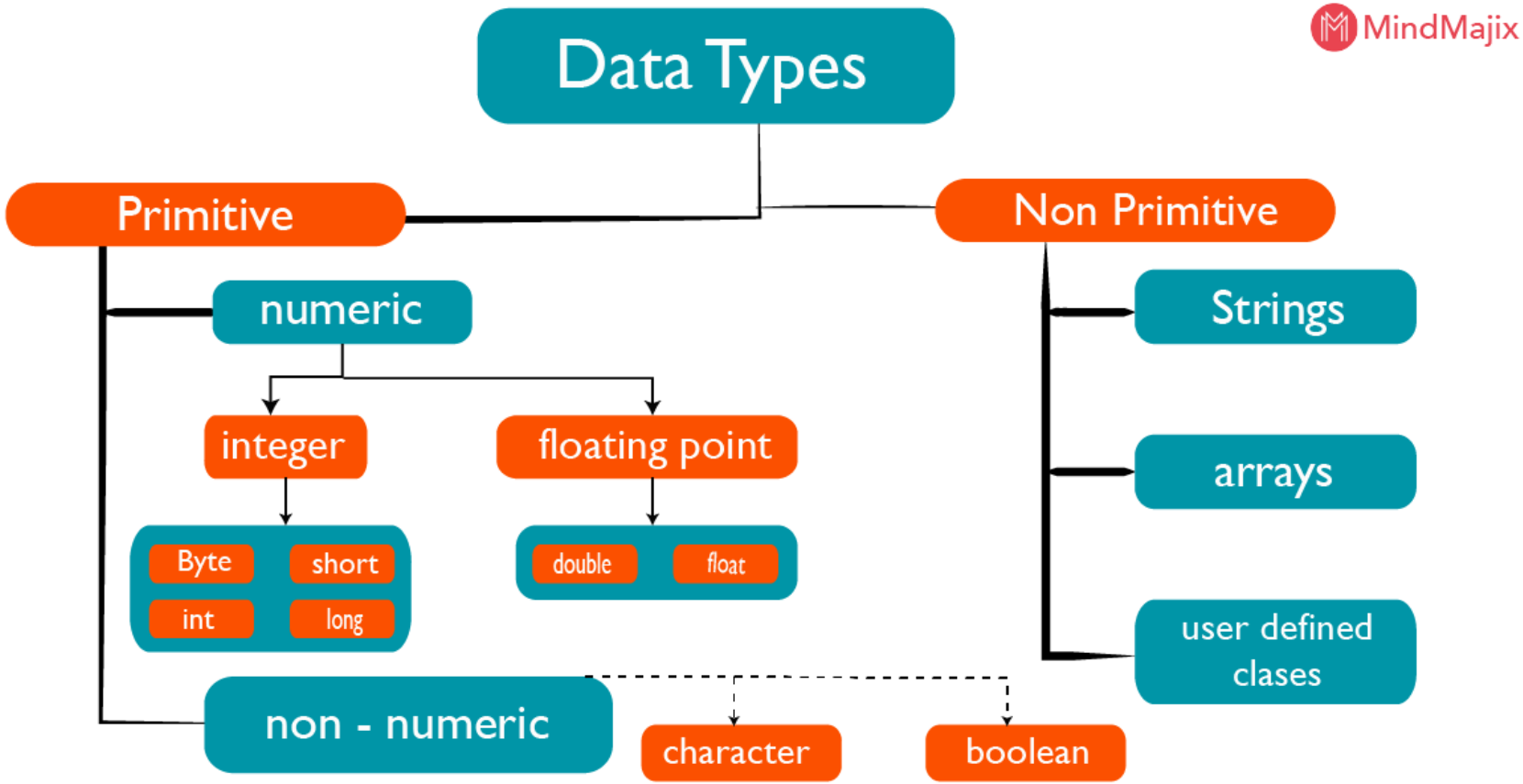
Datatype specifies type and size of the values that can be stored in variable.

Java is strongly typed language.

Every variable and expression has a type which is strictly defined.

All the expressions and assignments are checked for type compatibility at compile time.

Classification



Primitive Data Types

It is referred as simple/fundamental types.

Primitives types represents single values.

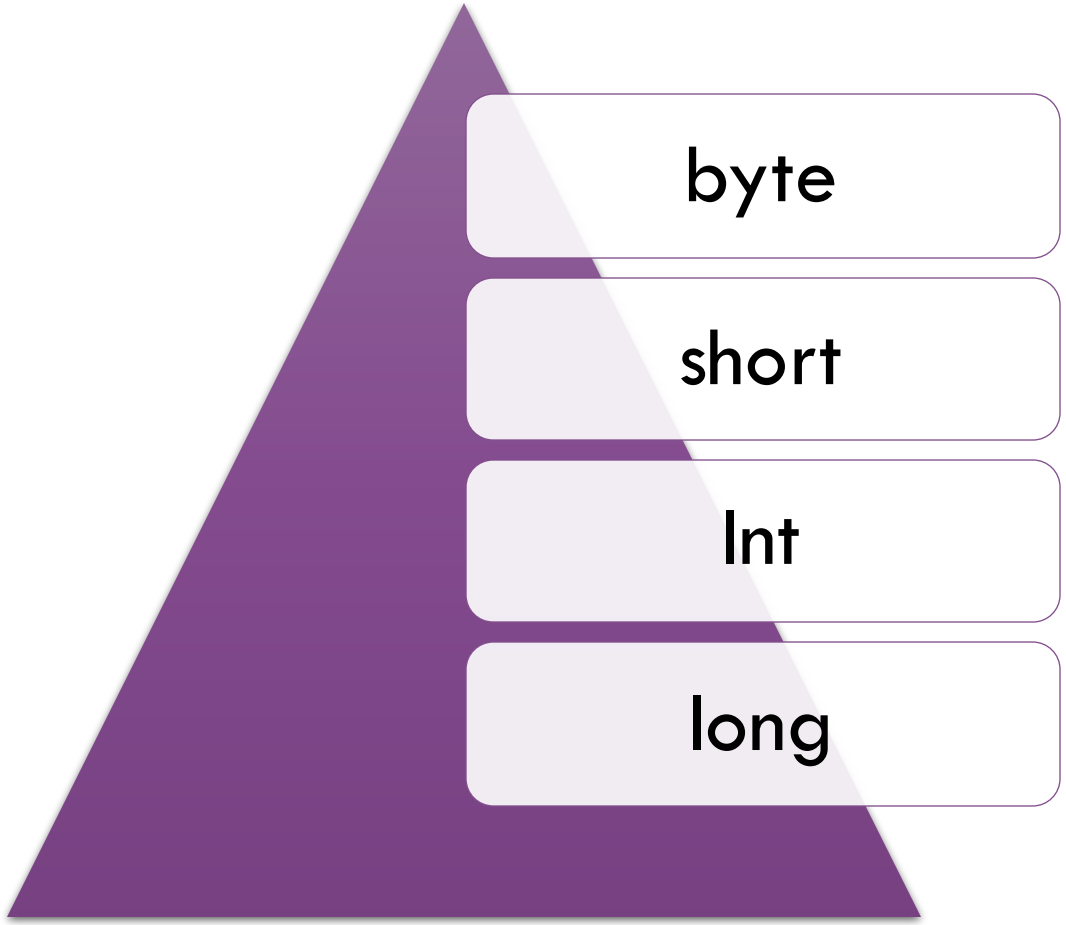
Primitives datatypes are those whose variable allows us to store only one value.

Primitive datatype variable can hold maximum one value at a time.

They form the basis for all other types of data that you can create.[array and class].

Java defines eight primitive types of data: **byte, short, int, long, char, float, double, and boolean.**

Integer



byte

- The smallest integer type is byte.
- This is a signed 8-bit type that has a range from –128 to 127.
- Variables of type byte are especially useful when you're working with a stream of data from a network or file.
- Example:
  - **byte** b, c;

short

- short is a signed 16-bit type.
- It has a range from –32,768 to 32,767.
- It is probably the least used Java type.
- Example:
  - **short** b,c;

int

The most commonly used integer type is int.

It is a signed 32-bit type that has a range from –2,147,483,648 to 2,147,483,647.

variables of type int are commonly employed to control loops and to index arrays.

Example :

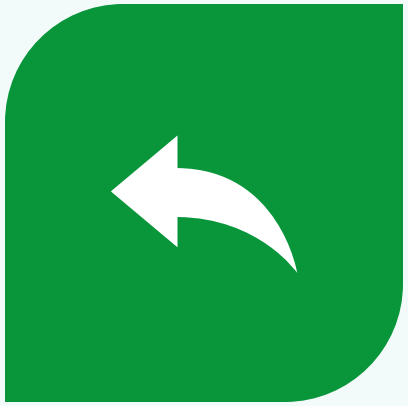
```
int a, b;
```

long

- long is a signed 64-bit type
- it is useful for those occasions where an int type is not large enough to hold the desired value.
- Example:
  - **long** a,b;

Floating Point

FLOAT



DOUBLE

float

- The type float specifies a single-precision value that uses 32 bits of storage.
- Variables of type float are useful when you need a fractional component. but don’t require a large degree of precision.
- For example : dollars and cents.

double

Double precision, as denoted by the double keyword, uses 64 bits to store a value.

All transcendental math functions, such as sin( ), cos( ), and sqrt( ), return double values.

Character

In Java, the data type used to store characters is **char.**

Java uses Unicode to represent characters

Unicode defines a fully international character set that can represent all of the characters found in all human languages

Character

It is a unification of dozens of character sets, such as Latin, Greek, Arabic, Cyrillic, Hebrew, Katakana, Hangul, and many more.

For this purpose, it requires 16 bits. Thus, in Java char is a 16-bit type.

Example: char ch ='y'.

Boolean

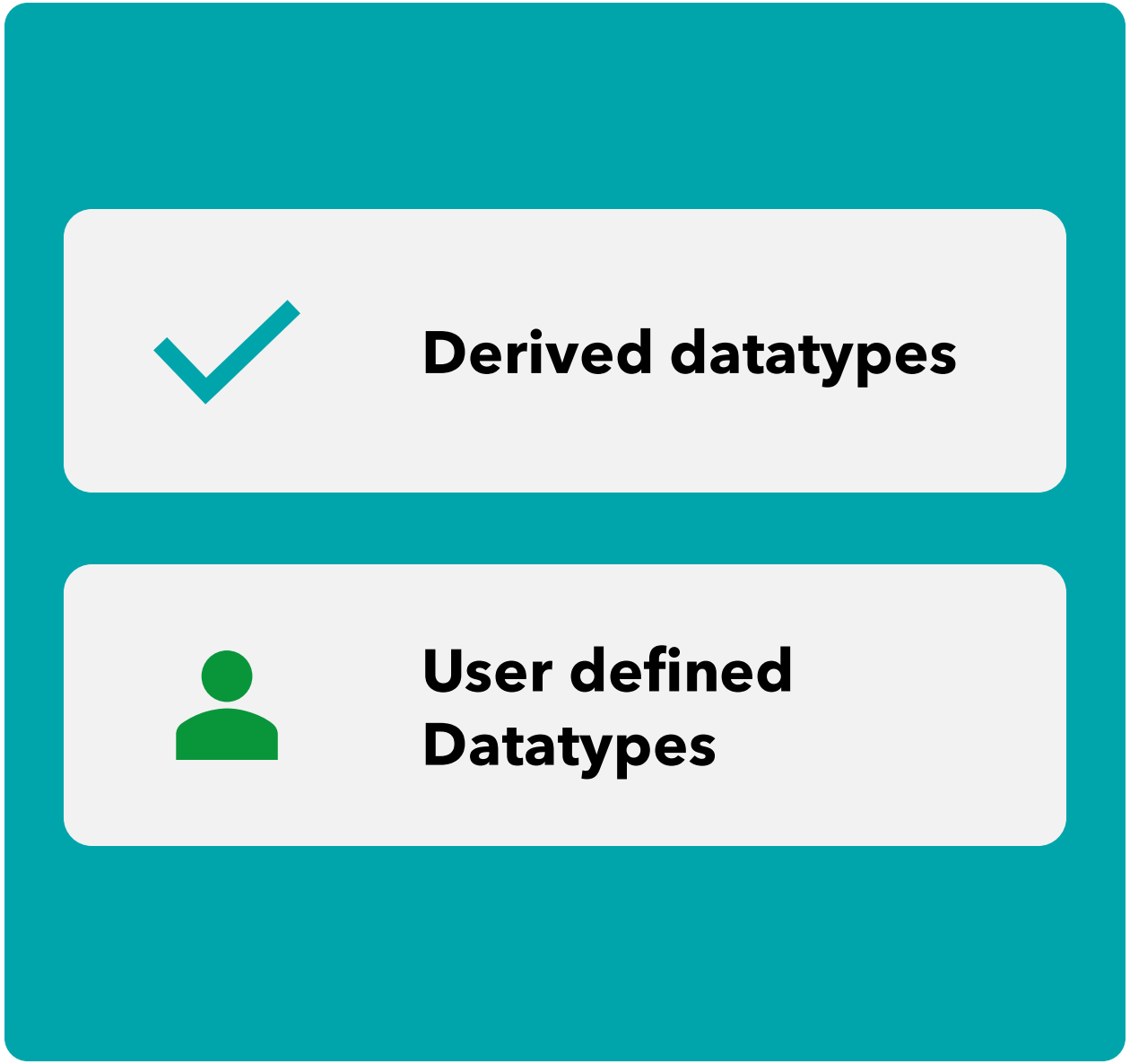
Java has a primitive type, called boolean, for logical values.

It can have only one of two possible values, **true or false.**

This is the type returned by all relational operators, as in the case of a < b.

Example: boolean val =true;

Non Primitive datatypes



Derived datatypes

Derived data types are those whose variables allow us to store multiple values of same type.

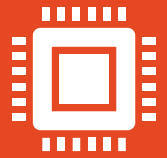
These are the data type whose variable can hold more than one value of similar type. In general derived data type can be achieve using **array.**

Example: int a[]={10,20,30};

User defined Datatypes:

User defined data types are those which are developed by programmers by making use of appropriate features of the language.

User defined data types related variables allows us to store multiple values either of same type or different type or both.



This is a data type whose variable can hold more than one value of dissimilar type, in java it is achieved using class concept.

Note

- In java both derived and user defined data type combined name as reference data type.
- Reference datatypes in java are those which contains reference/address of dynamically created objects. These are not predefined like primitive data types.
- In C language, user defined data types can be developed by using struct, union,enum etc.
- In java programming user defined datatype can be developed by using the features of classes and interfaces.

Literals

Java Literals	Java Literal Examples			
Integer Literal	100	0	27530	18
Floating Point Literal	3.14	-0.536	26783.087	0.9
String Literal	"India"	"838365"	"Good"	"cat"
Character Literal	'A'	'a'	'#'	')'
Boolean Literal	true	false		

A constant value in Java is created by using a literal representation of it.

They are called literals because they explicitly identify specific datatype values.

GARBAGE COLLECTION IN JAVA

WHY WE NEED ?

- In C/C++ it is responsibility of programmers for creation and destruction of objects.
- Usually programmer neglects destruction of useless objects which will leads to **Out Of Memory Errors.**

WHAT IS GARBAGE COLLECTION?

Garbage means **unreferenced** objects.

An Object is said to be unreferenced iff it does not contain any reference to it.

Garbage collection is the process of reclaiming runtime **unreferenced** and unused memory automatically.

Main objective of Garbage collector is to free heap memory by destroying unreferenced objects.

MAKE AN OBJECT ELIGIBLE FOR GARBAGE COLLECTION

Nullifying the reference variable.

Re-assigning the reference variable.

Anonymous Objects.

NULLIFYING THE REFERENCE VARIABLE

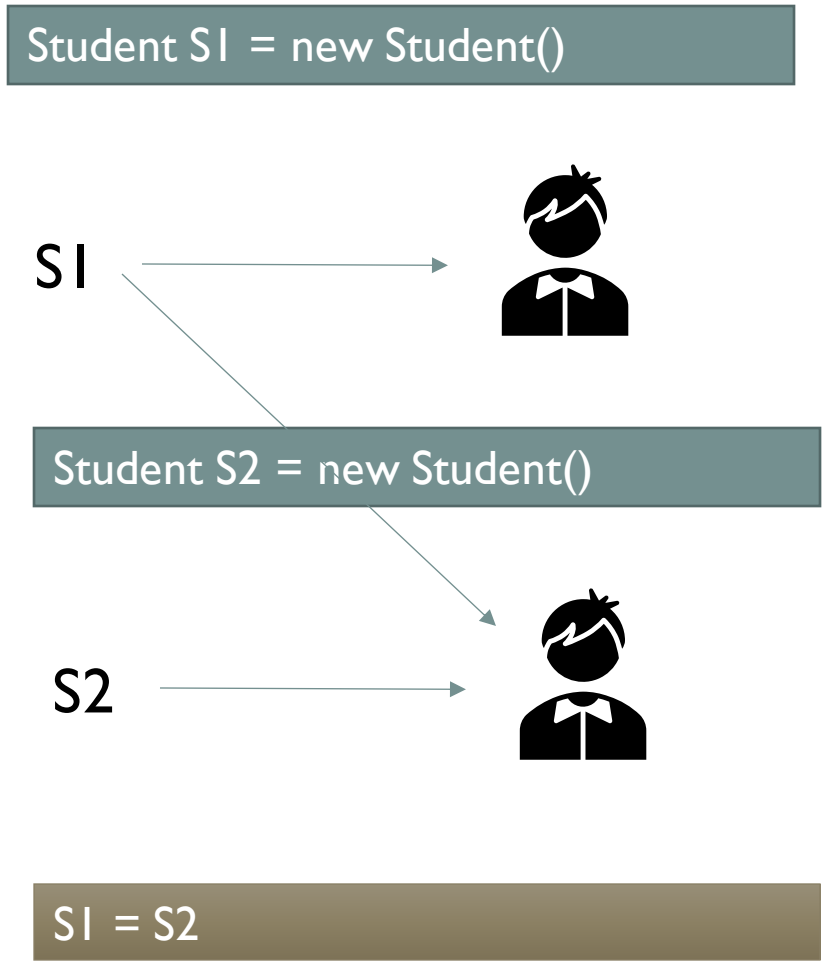
- When all the reference variables of an objects is changed to **null** , it becomes unreachable or unreferenced thus it is eligible for garbage collection.

```
Student S1 = new Student()
```


```
S1 = null
```

RE-ASSIGNING THE REFERENCE VARIABLE

- When reference of one object is assigned to reference of some other object then previous object has no longer any reference to it and become unreachable.



ANONYMOUS OBJECT

- Reference of an anonymous object is not stored anywhere.

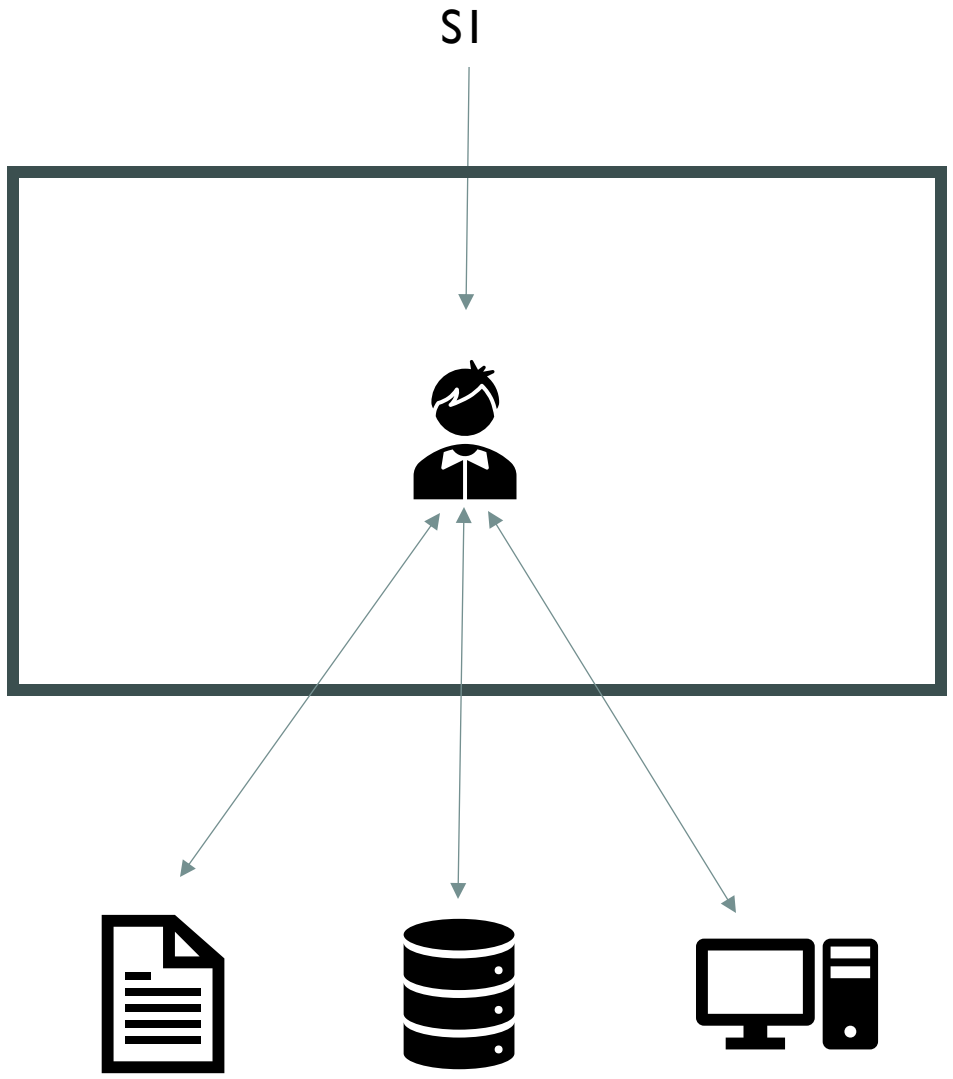
```
new Student()
```

FINALIZE() METHOD

Sometime object need to perform some action before being destroyed.

if object is holding some non java resources like file handler then programmer might make sure that the resources is freed before object is being destroyed.

It is a method that the garbage collector always called just before destroying each object which is eligible for garbage collection to perform **cleanup activity.**



- Cleanup activity means closing connection of resources with that object like database connection , network connection.
- **Syntax:**
  - **protected void finalize() { }**
  - **Object** class contains an empty implementation finalize() method .

NOTE

- So JAVA provide finalize() method using which we can define specific action that will occure when an object is just about to be reclaimed by the garbage collector.
- finalize() is called just prior to garbage collection .it is not called when object goes out of scope. so we cannot know when or even if **finalize()** will be executed.

GC() METHOD

- garbage collector run periodically to reclaim unused objects.
- But sometime we want to collect discarded objects memory prior to garbage collector's next appointed round.
- We can call garbage collector on demand / explicitly by calling **gc() method.**

- **gc() Method** does not guarantee that JVM will perform garbage collection , it only request JVM for garbage collection.
- This method is present is **System** and **Runtime** class.
- **Example : System.gc()**

ADVANTAGES

- Programmer does not need to worry about deallocation of the object's memory , it is automatically done by JVM
- Increase memory efficiency and decrease the chances of memory wastage.

WRAPPER CLASS

What is Wrapper Class?

In Java we can not use primitive datatype as an object directly.

So Wrapper classes are used to wrap the primitive datatype into corresponding objects.

In java there wrapper classes for each primitive datatypes.

What is Wrapper Class?

Wrapper class encapsulate primitive datatypes within an object.

Each wrapper classes have different methods to perform various operation.

Wrapper classes are part of **java.lang** package.

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean

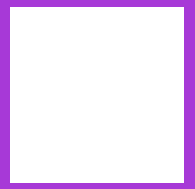
Purpose of Wrapper Class

it provide the mechanism to wrap the primitive types into corresponding object.

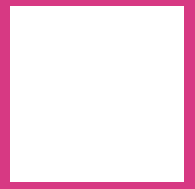
It provides different function to convert primitives to and from String objects and other different bases such as binary , octal and hexadecimal.

Many of the standard data structures implemented by Java operate on objects, which means that you can’t use these data structures to store primitive types.(ArrayList,HashMap)

Purpose of Wrapper Class



We cannot pass primitive datatype variable by reference to a method.



in java to pass primitive data values as a reference to a method first we need to wrap the primitive datatype variable in to object using appropriate wrapper class and that wrapped object reference is passed to a method.

How to use it?

01

Using Constructor

02

Using **ValueOf()** method

Using Constructor

The process of encapsulating a value within an object is called **boxing.**

Each wrapper class provides two types of constructors except Character

- **for its primitive types**
  - Integer i1 = new Integer(10);
- **for its String representation**
  - Integer i1 = new Integer("100");

Using valueOf() Method

ValueOf() is a static method so it can be invoked directly by the class.

Method 1 Example

Integer x1 = Integer.valueOf(10);

Method 2 Example

Integer I1 = Integer.valueOf("100",2);

Integer I2 = Integer.valueOf("1100",8);

In this method there are two parameters value and radix(base).

Unwrapping Wrapper Object

The process of extracting a value from a wrapper object is called unboxing.

- **xxxValue() Method**

it is used to convert wrapped numeric value into another primitive type.

```
Integer I1 = new Integer(10);

byte b =I1.byteValue();

int  i =I1.intValue();
```

Unwrapping Wrapper Object

- **parseXxx() Method**

it is used to represent primitive datatype value of String object.

```
int i = Integer.parseInt("100");

int j = Integer.parseInt("10110",2);
```

Note :

parseXxx() return primitive value of given numeric String representation

valueOf() return newly created wrapper object.

String representation of Numerical Object

- ToString() Method

```
Integer X =10;
String s  =X.toString();
```

- ToxxxString() Method

```
Integer.toBinaryString(10);
Integer.toOctalString(10);
Integer.toHexString(10);
```

AutoBoxing and Auto-unboxing

Autoboxing is the process by which a primitive type is automatically encapsulated (boxed) into its equivalent type wrapper whenever an object of that type is needed.

Integer iob =100;

Auto-unboxing is the process by which the value of a boxed object is automatically extracted (unboxed) from a type wrapper when its value is needed.

int i =iob;


TYPE
CONVERSION

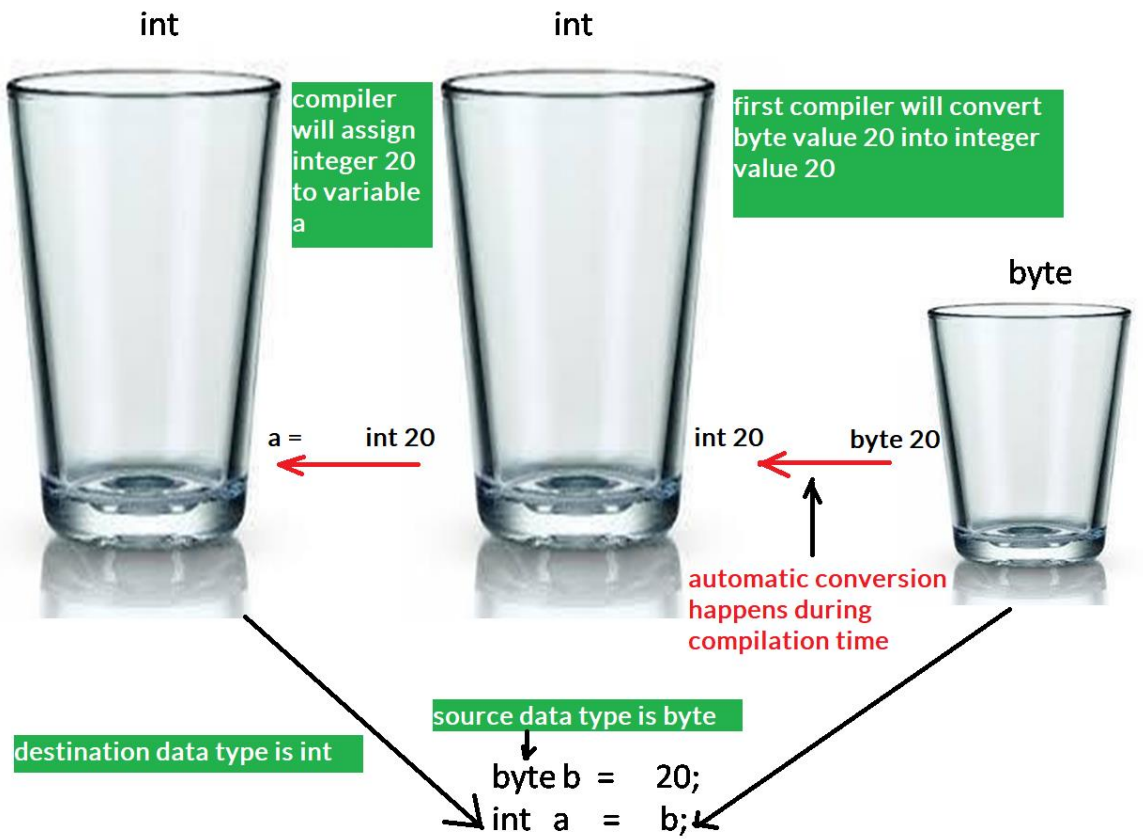
TYPE CONVERSION

it is fairly common to assign a value of one type to a variable of another type.

If the two types are compatible, then Java will perform the conversion automatically.

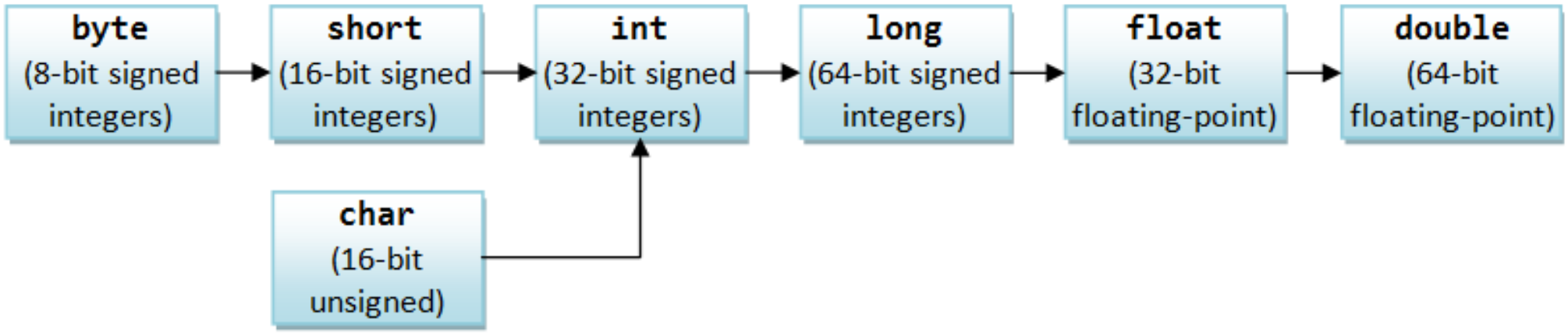
IMPLICIT TYPE CONVERSION

- When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:
  - The two types are compatible.
  - The destination type is larger than the source type.
- When these two conditions are met, a **widening conversion** takes place.



- It is performed by compiler automatically there is no loss of precision
- In Widening conversion rule is to promote the smaller type to a bigger type to prevent loss of precision
- For widening conversions, the numeric types, including integer and floating-point types, are compatible with each other
- However, there are no automatic conversions from the numeric types to char or Boolean.

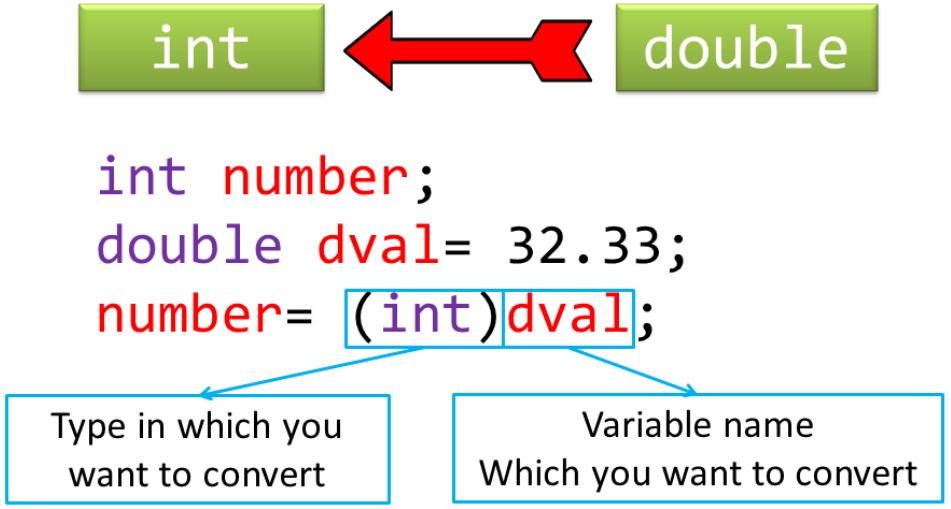
WIDENING CONVERSION



Orders of Implicit Type-Casting for Primitives

EXPLICIT TYPE CONVERSION

- It forces an explicit conversion of the type of a value.
- it is an operation which take one operand ,operates on its operand and return equivalent value in specified type.
- general syntax :
  - new value = **(typecast)**value;
  - if you want to assign an int value to a byte variable? This conversion will not be performed automatically, because a byte is smaller than an int.



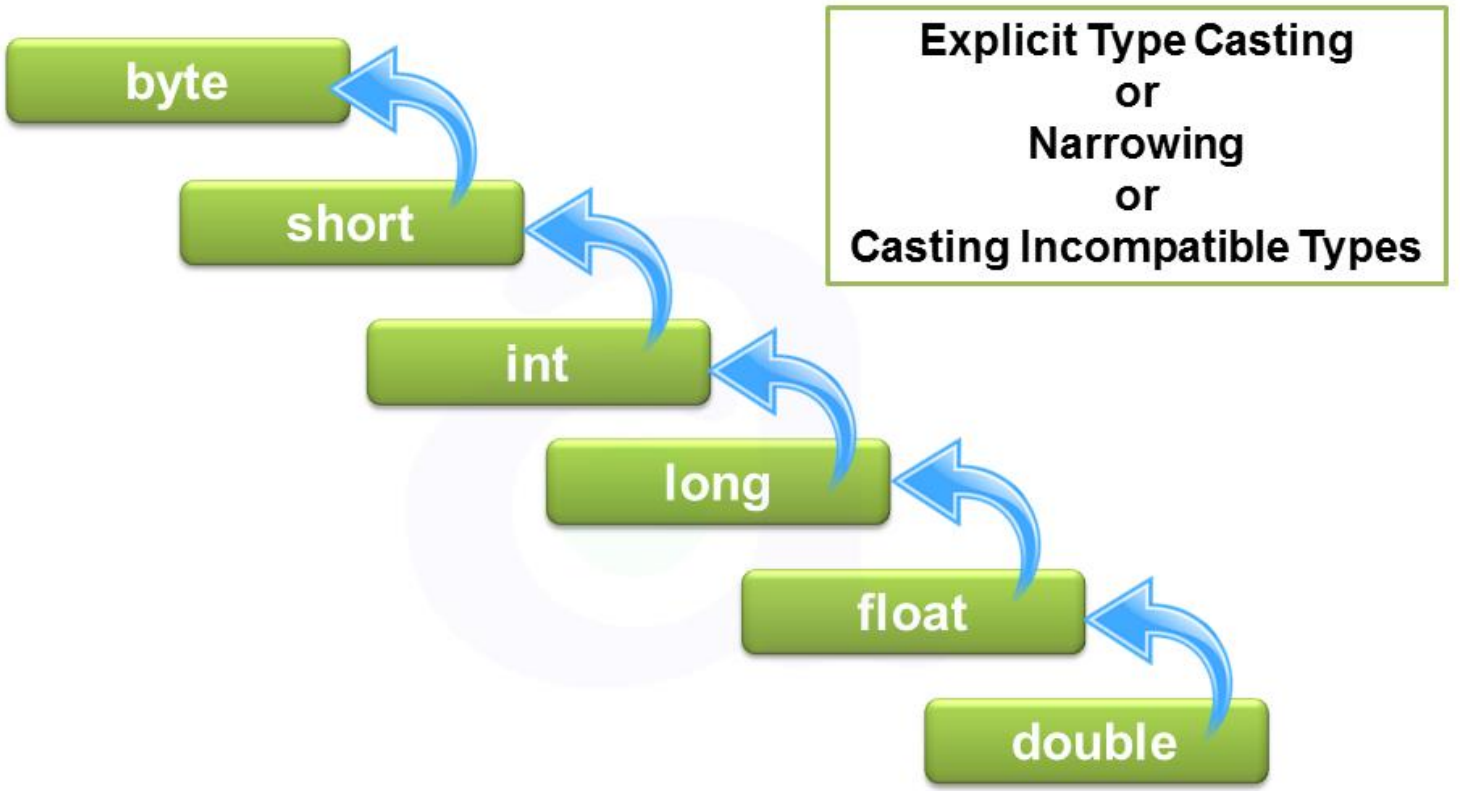
This kind of conversion is sometimes called a narrowing **conversion.**

since you are explicitly making the value narrower so that it will fit into the target type.

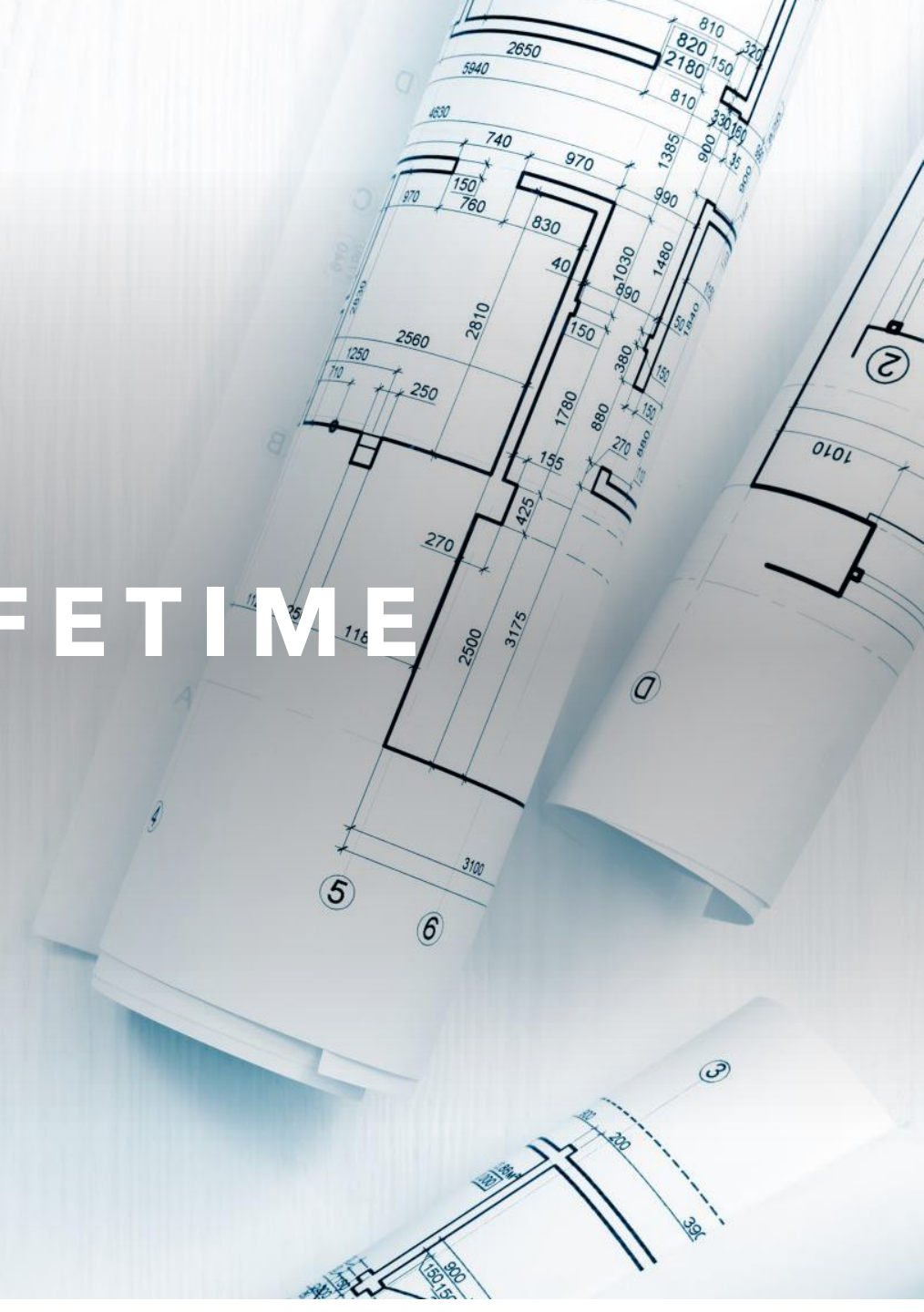
narrowing conversion informs the compiler that you are aware of possible loss of precision

boolean value cannot be type casted.

NARROWING CONVERSION



SCOPE AND LIFETIME



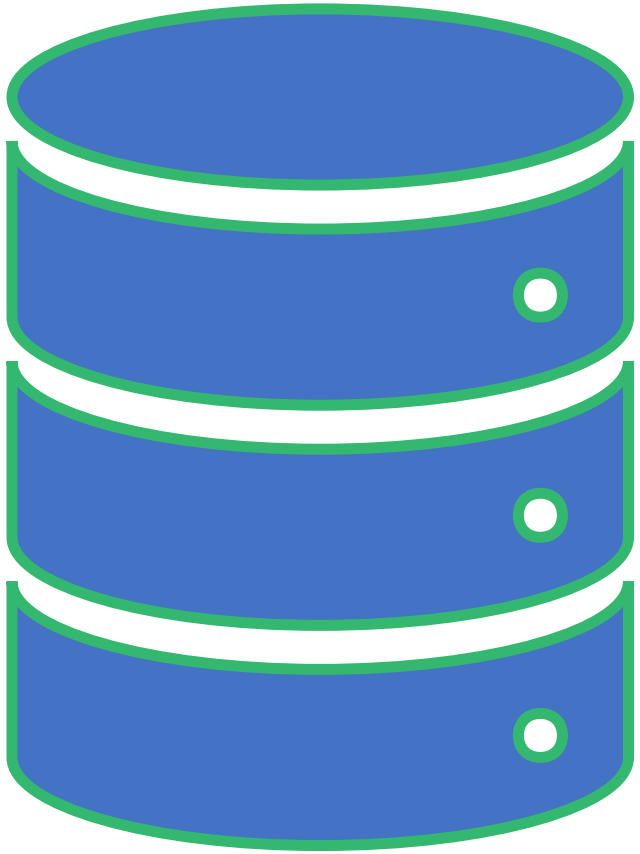
The scope of a variable specifies the region of the source program where that variable is known, accessible and can be used.

Generally, variables that are defined within a block of code are not accessible outside that block.

SCOPE OF A VARIABLE

LIFETIME OF A VARIABLE

- The lifetime of a variable refers to how long the variable exists before it is destroyed.
- Destroying variables refers to deallocating the memory that was allotted to the variables when declaring it.
- variables are created when their scope is entered and destroyed when their scope is left.
- This means that a variable will not hold its value once it has gone out of scope.



TYPES OF VARIABLES

INSTANCE VARIABLE

- Instance variables are those that are defined within a class itself and not in any method.
- They are known as instance variables because every instance of the class (object) contains a copy of these variables.
- The scope of instance variables is determined by the access specifier that is applied to these variables.
- The lifetime of these variables is the same as the lifetime of the object to which it belongs.

TYPES OF VARIABLES

ARGUMENT / FORMAL VARIABLE

- These are the variables that are defined in the header of a method.
- The scope of these variables is the method or constructor in which they are defined.
- The lifetime is limited to the time for which the method keeps executing. Once the method finishes execution, these variables are destroyed.

TYPES OF VARIABLES

LOCAL VARIABLE

- A local variable is the one that is declared within a method
- scope and lifetime are limited to the method itself.


OBJECT ORIENTED PROGRAMMING WITH JAVA
4341602

What is Java?

Java is a simple, object-oriented programming language.

Why Java Programming?

It is easy to learn

Java is an open source

Java's rich API

Great community support

Finds use in real world applications

TEACHING AND EXAMINATION SCHEME

Teaching Scheme			Total Credit	Examination Scheme				
				Theory Marks		Practical Marks		Total Marks
L	T	P	C	ESE	CA	ESE	CA	150
3	0	4	5	70	30	25	25	

Course Details

Unit No	Unit Title	Marks
1	Introduction to OO Paradigm	16
2	Object Oriented Programming concepts	12
3	Inheritance , Packages and Interfaces	16
4	Exception handling ,Multi threaded programming	16
5	File handling	10

Course Outcomes

- [x] Understand OOP (Object-Oriented Programming) concepts with java.
- [ ] Understand building blocks of OOPs language, inheritance, package and interfaces.
- [ ] Implement exception handling and multithreading in object oriented programs.
- [ ] Develop an object oriented program handling a Text file.

Application of Java in Real World

Desktop GUI based App

Mobile Apps

Web Applications

Embedded System

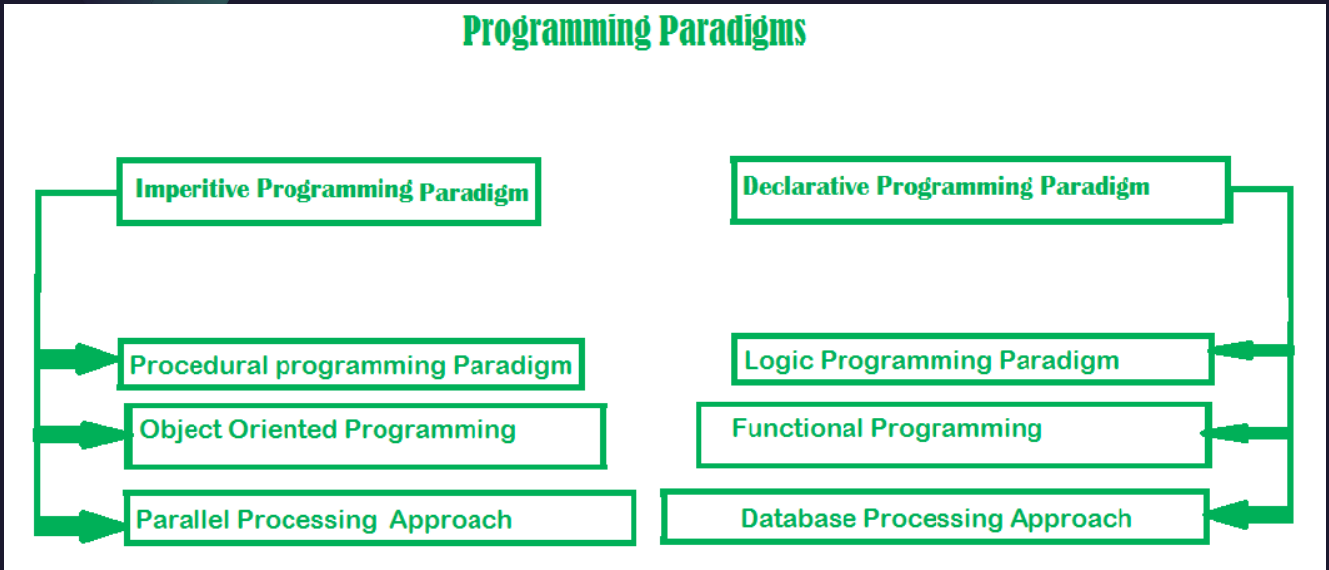
Cloud computing

Big data Analysis

Real World Application

- Adobe acrobat
- Amazon
- Netflix , Uber
- Java card
- Hadoop

What is Programming Paradigm?



- The term programming paradigm refers to a style of programming.
- There are lots of programming languages that are well-known but all of them need to follow some strategy when they are implemented. And that strategy is a paradigm.

Imperative programming paradigm

```c
#include <stdio.h>

int main()
{
    int sum = 0;
    sum += 1;
    sum += 2;
    sum += 3;
    sum += 4;
    sum += 5;
    sum += 6;
    sum += 7;
    sum += 8;
    sum += 9;
    sum += 10;

    printf("The sum is: %d\n", sum); //prints-> The sum is 55

    return 0;
}
```

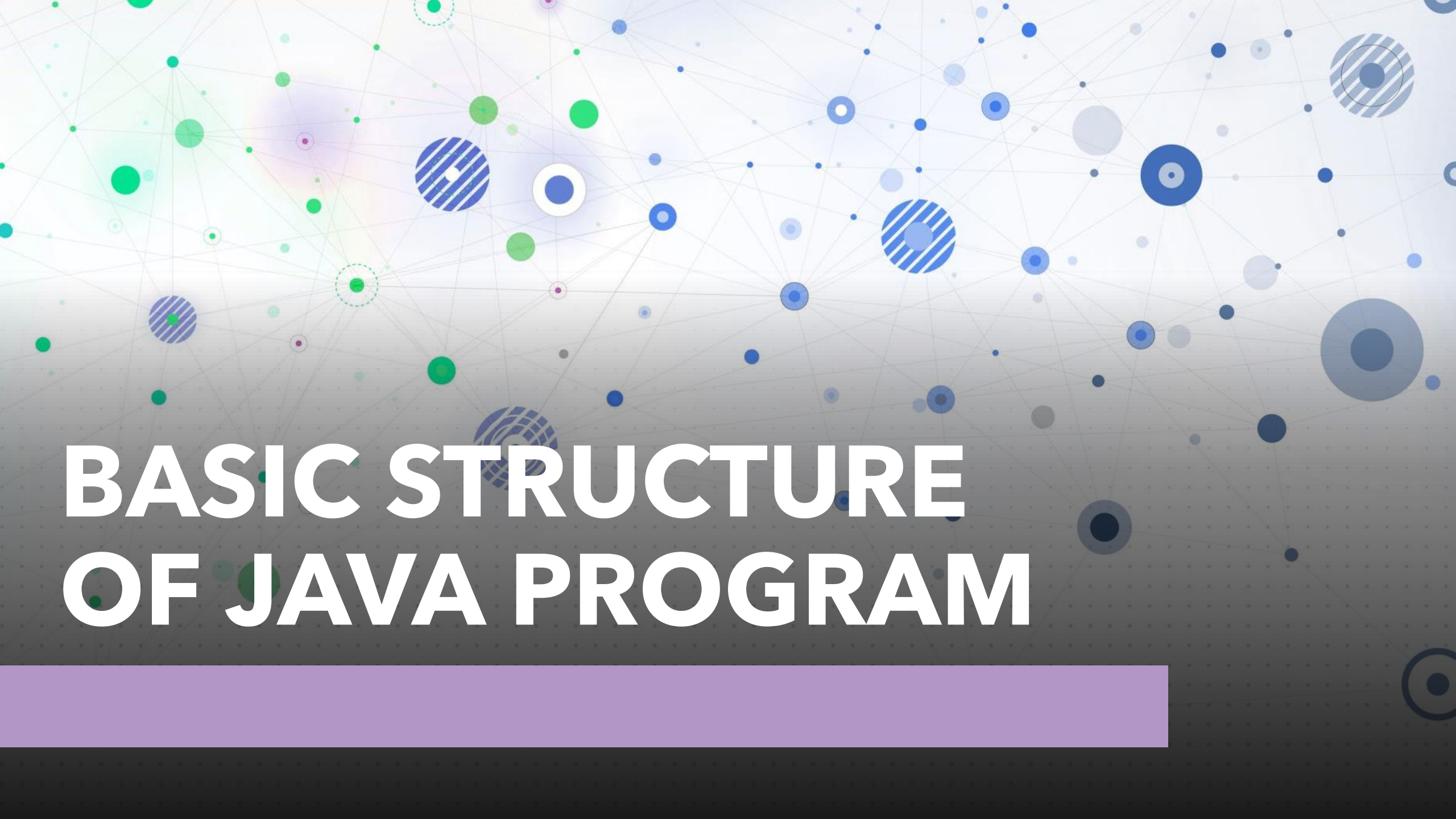
- The paradigm consists of several statements, and after the execution of all of them, the result is stored.
- It’s about writing a list of instructions to tell the computer what to do step by step.
- In an imperative programming paradigm, the order of the steps is crucial.

Declarative Programming Paradigm

Example in Prolog:

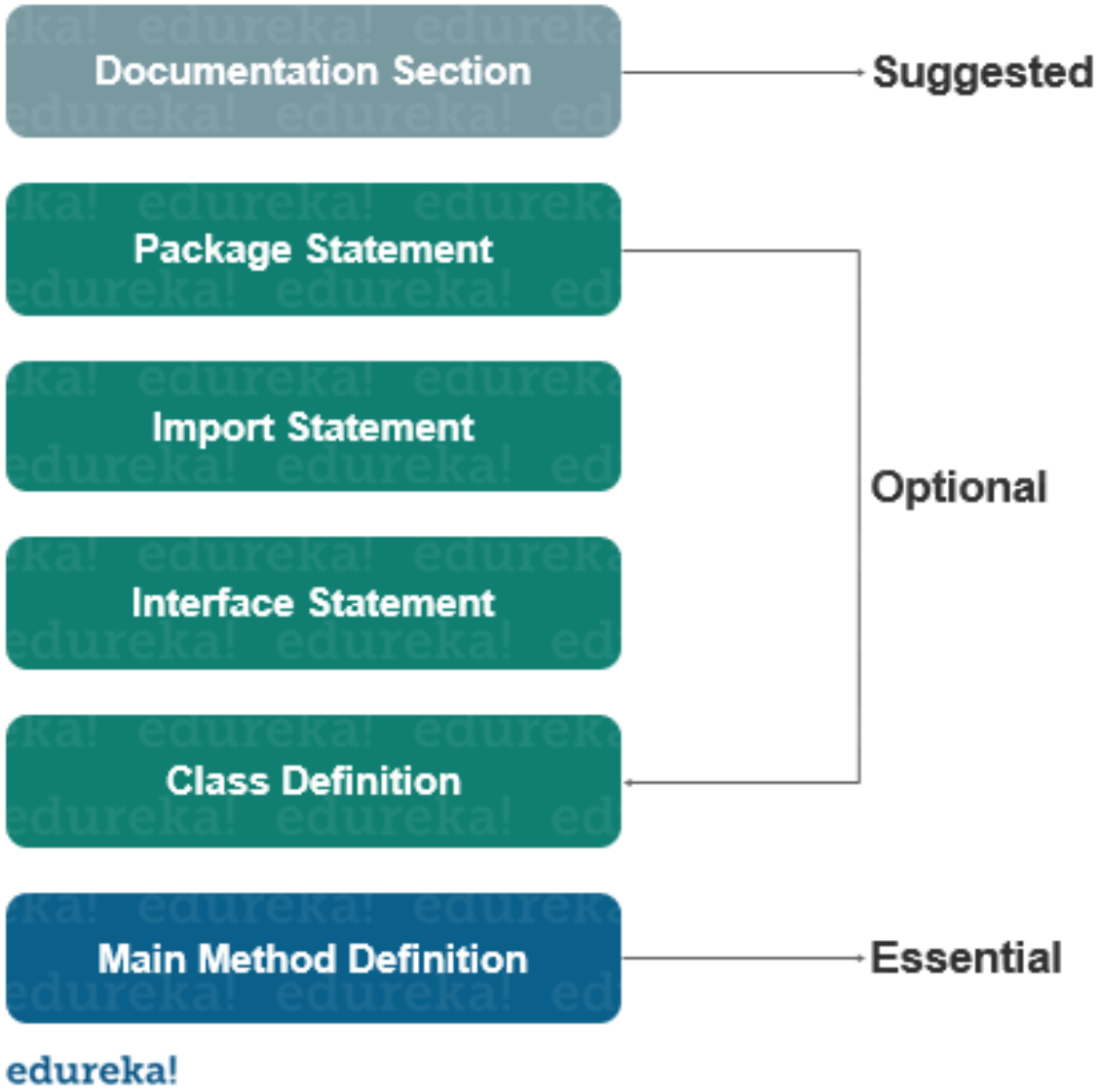
```
/*We're defining family tree facts*/
father(John, Bill).
father(John, Lisa).
mother(Mary, Bill).
mother(Mary, Lisa).

/*We'll ask questions to Prolog*/
?- mother(X, Bill).
X = Mary
```

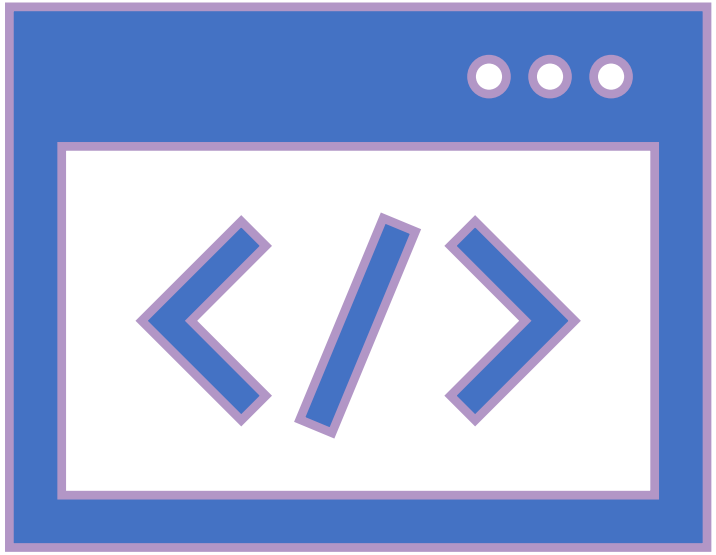
- Declarative programming is a programming paradigm in which the programmer defines what needs to be accomplished by the program without defining how it needs to be implemented.
- the approach focuses on what needs to be achieved instead of instructing how to achieve it.


BASIC STRUCTURE
OF JAVA PROGRAM

- In JAVA source file is called compilation unit.
- it is a text file that contains one or more class definitions.
- Java source file use the **.java** file extension.
- By convention name of the class should match the name of file that holds the program.
- Java is case sensitive language

Basic Structure





Documentation

- Java program begins with comments.
- The content of comments are ignored by compiler.
- the Comments describe the program to anyone who is reading the program
- it also describes how some part of program works and what specific feature does.
- Java supports 3 style of comments.
  - Multiline comments -**/* */**
  - Single line comments **//**
  - Documentation comment **/** */**
    - **i**t is used to produce an html file that documents your program.

Package Statement

- There is a provision in Java that allows you to declare your classes in a collection called package.
- There can be only one package statement in a Java program, and it has to be at the beginning of the code before any class or interface declaration.
- **Example : package** `student;`

Import Statement

- Many predefined classes are stored in packages in Java .
- An import statement is used to refer to the classes stored in other packages.
- An import statement is always written after the package statement but it has to be before any class declaration.
- **Example :**
- `Import java.util.*; //imports the all classes.`

Interface Section

This section is used to specify an interface in Java.

It is an optional section which is mainly used to implement multiple inheritance in Java.

An interface is a lot similar to a class in Java but it contains only constants and method declarations.

Class Definition

A Java program may contain several class definitions, classes are an essential part of any Java program.

A class is a collection of variables and methods that operate on the fields.

Every program in Java will have at least one class with the main method.

class keyword is used to declare new class

Example : **class class_name**

here **class_name** is any valid identifier which indicates name of the class

Entire class definition including allof its members will be between opening and closing curly braces.

Main Method Section

- The main method is from where the execution actually starts and follows the order specified for the following statements.

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

- Java program execution begins with main method.

public

keyword / access specifier

it allows programmer to control the visibility of class members.

if class member is precede by public keyword then that member can be accessed by code outside the class in which it is declared .

main() method is declared public because it must be call by the code outside of its class when program is started.

static

it allows main() to be called without having to instantiate a perticular instance of the class.

As main() method is called by java interpreter before any object is created.

void

- It tells the compiler that main() method does not return any value.

String args[]

- any information that you need to pass to method is received by variables specified within the set of parentheses that follow the name of method.
  - args is the array of objects of type String.
  - object of String type is used to store character string.
  - **args** receives data if any command line arguments are present while program is executed.
- all code that comprises a main() method must be occur between { } of main method.

System.out.println("Message")

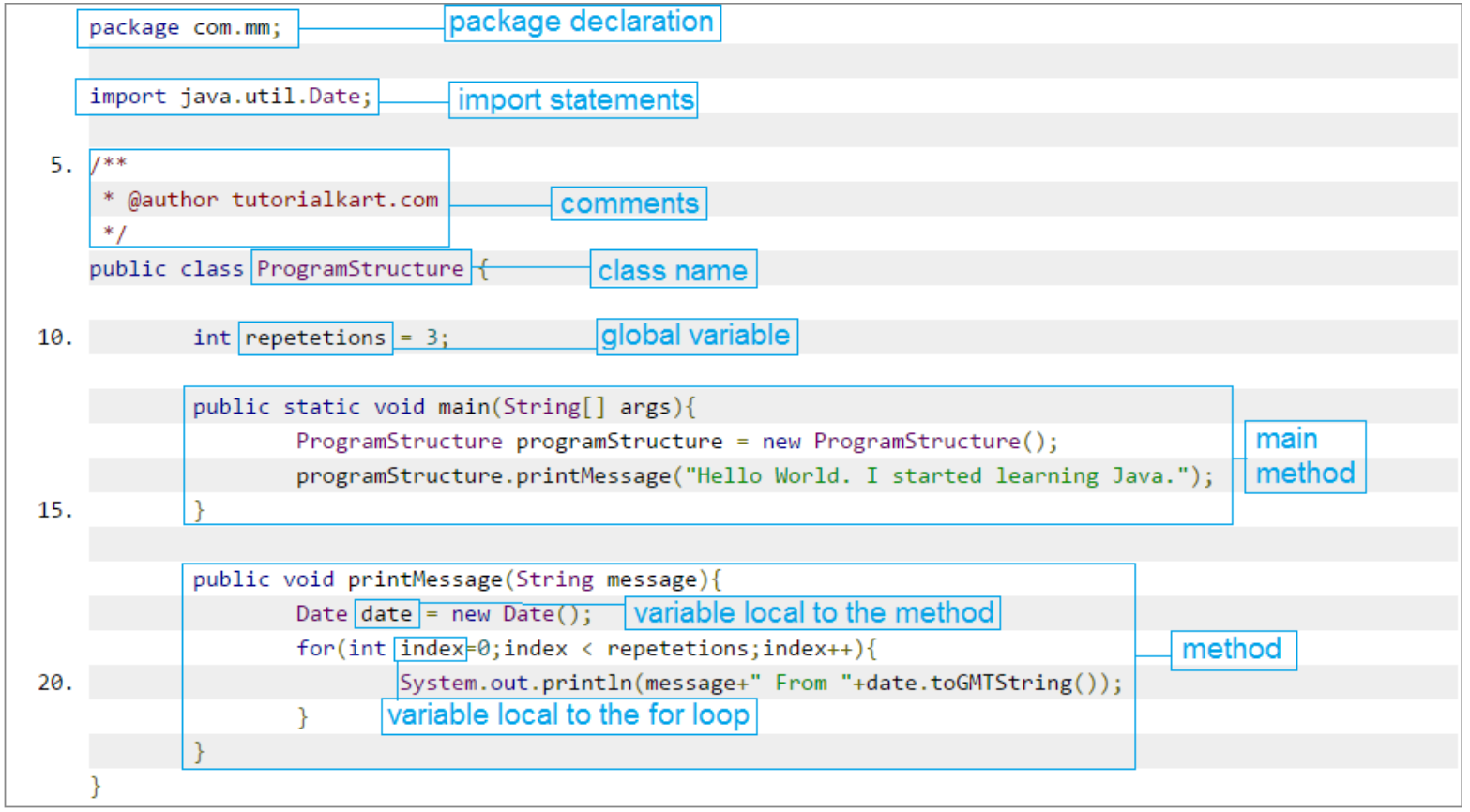
System is the predefined class that provide access to system.

out it is a output stream that is connected to console.

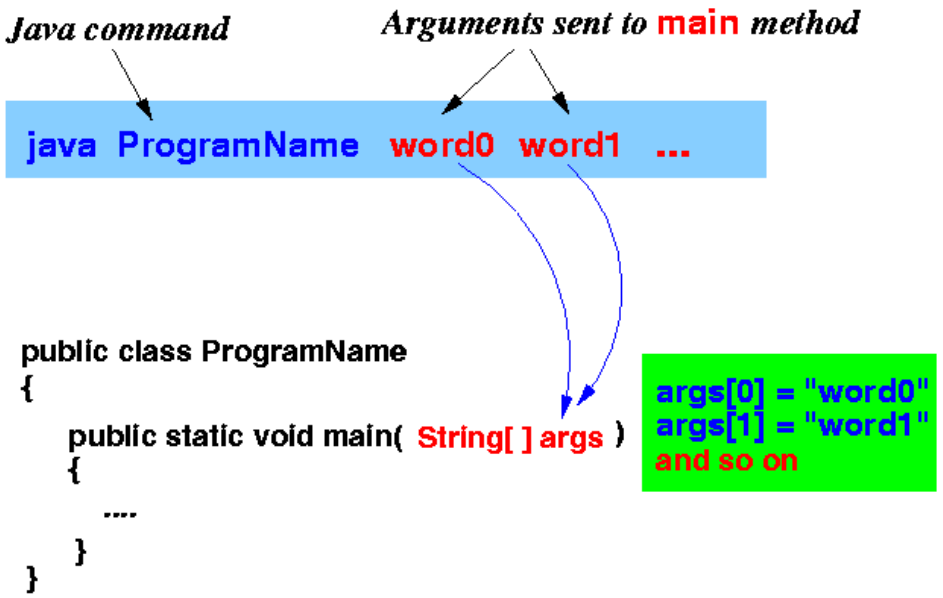
println method is used to display string which is passes in it.

A Complex program may have dozen of classes only one of which will need to have **main()** method to get started.

Sample Program




COMMAND LINE ARGUMENTS



COMMAND LINE ARGUMENT

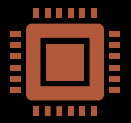
- During the execution of java program information/arguments passed following a program name in the command line is called command line arguments.
- In Java command line arguments are passed to main method using array of String objects.
- public static void main(**String args[]**){ }
- **args** is an array of String object which holds each command line arguments.


FEATURES
OF JAVA

SIMPLE

Java inherits syntax of c/c++ and many features of OOP

Java has rich set of APIS



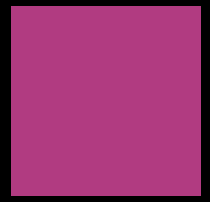
Java has automatic garbage collection which is always used to collect unreferenced memory location for improving performance of java program.

Java is free from pointer.

Platform Independent

language is said to be platform independent if and only if which can run on all available operating system with respect to its development and compilation.

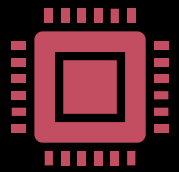
Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms.



Write Once and Run Anywhere(WORA).

Architectural Neutral

language is said to be architectural neutral if and only if which can run on all available processors in the real world without considering their architecture and vendor.



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

Portable

Application written using java can be executed on any kind of CPU or OS.

it facilitates you to carry the Java bytecode to any platform.

Portability = Platform independent + architecture neutral

Object Oriented

Everything in Java is an object.

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

All datatypes are objects except Primitive datatypes.

We can use the concept of abstraction,inheritance,encapsulation, Polymorphism to simplify software development and maintenance.

Robust

Robust means Strong.

There is a lack of pointers that avoids security problems.

There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.

There are exception handling and the type checking mechanism in Java. All these points make Java robust.

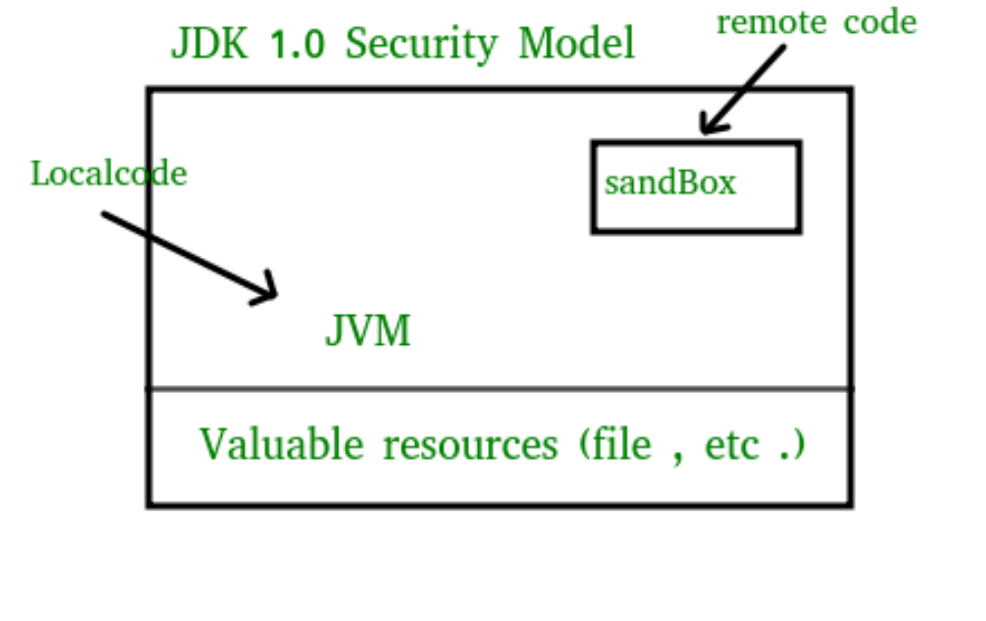
Secure

All java code is converted into byte code after compilation which is not human readable.

No use of explicit pointer.

java program runs inside the virtual machine sandbox to prevent activity from untrusted source.

Sand Box

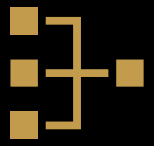


a sandbox is an isolated computing environment in which a program or file can be executed.

The original security model provided by the Java platform is known as the sandbox model, which existed in order to provide a very restricted environment in which to run untrusted code obtained from the open network.

Dynamic

Supports dynamic memory allocation.

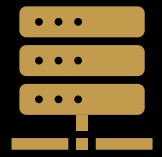


'new ' operator can be used for dynamic memory allocation

It supports dynamic loading of classes. It means classes are loaded on demand

Distributed

it allows users to create distributed applications in Java.



RMI and EJB are used for creating distributed applications.

This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Multithreaded

We can write Java programs that deal with many tasks at once by defining multiple threads.

The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area.

Threads are important for multi-media, Web applications, etc.

Supports multi process synchronization

High Performance

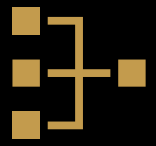
using bytecode which can be easily converted into native machine code.

Garbage collector that collects the unused memory space and improve performance.

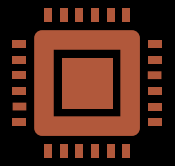
use of JIT improves the performance.

Networked

it is used to develop web application.



J2EE is used for developing networked based application.



Socket programming can be done using TCP/IP protocol.

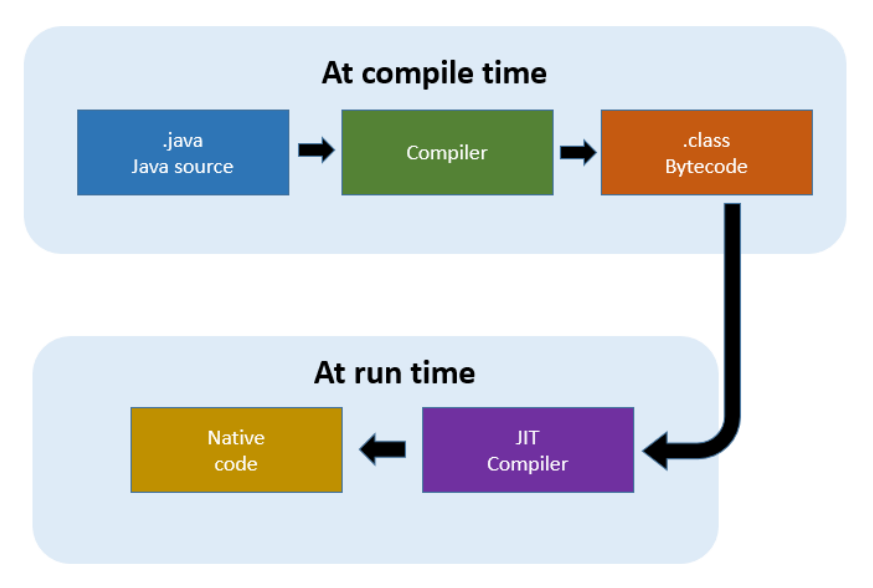
Interpreted

Java Bytecode is interpreted by JVM and converted machine specific code

sun micro system has developed program called JIT.

it is added as a part of JVM to speed up the interpretation phase.

JIT



Java interpreter interprets the bytecode faster but execute it slowly.

when same method is called multiple times everytime interpretation is required.

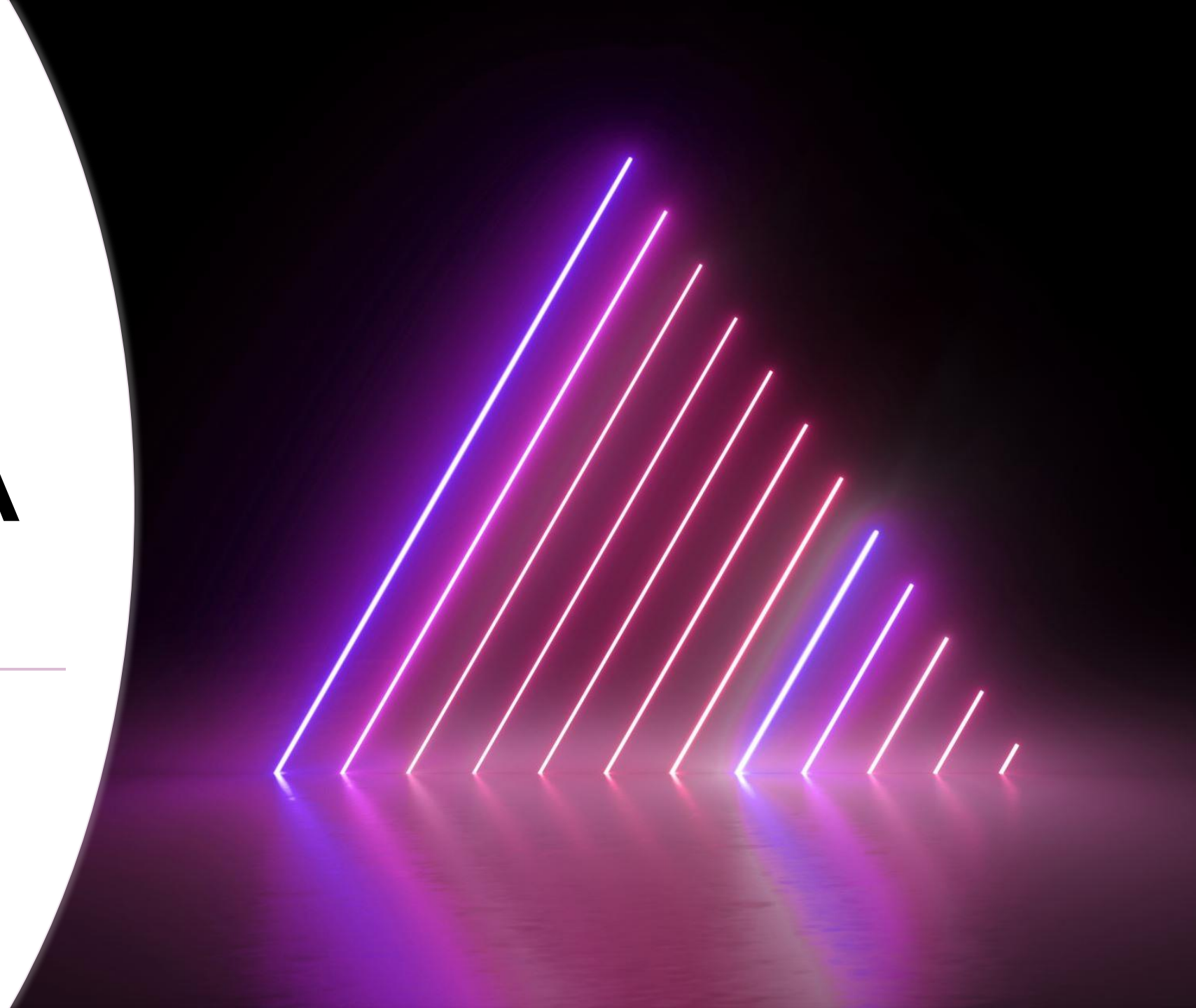
Java runtime system take help of interpreter in converting bytecode but when it finds the repeated code it uses the JIT compiler.

JIT compiles the entire sequence of bytecode into machine code which can be used directly for repeated calls.

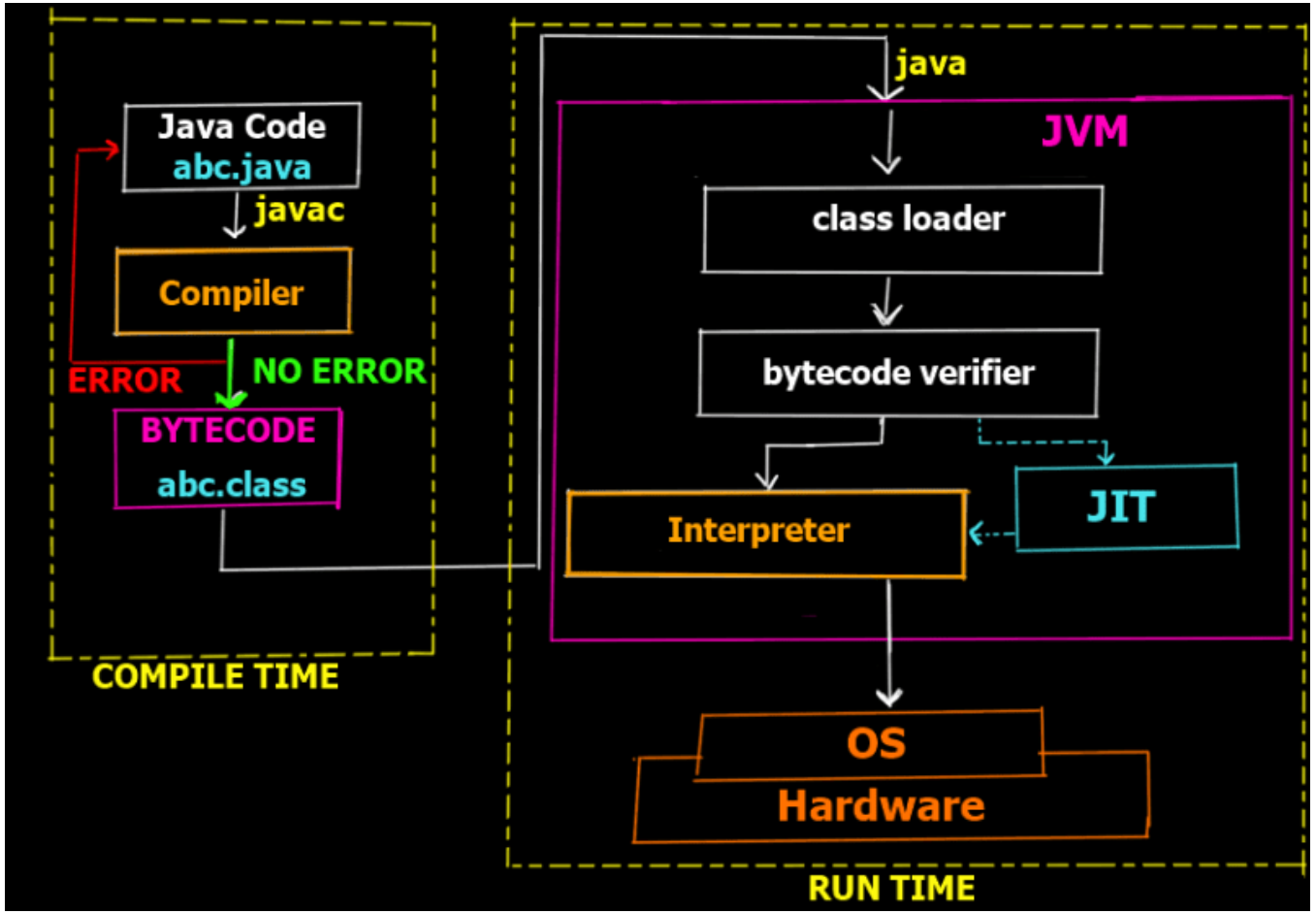
J2SE vs J2ME vs J2EE

- J2SE(Java Standard Edition) is also known as Core Java, this is the most basic and standard version of Java.
- J2SE is mainly used to create applications for Desktop environment.
- J2ME(Java Micro Edition)is mainly concentrated for the applications running on embedded systems, mobiles and small devices.
- J2EE(Java Enterprise Edition)has a much larger usage of Java, like development of web services, networking, server side scripting and other various web based applications.

COMPILING AND RUNNING A SIMPLE JAVA PROGRAM



JAVA EXECUTION FLOW DIAGRAM



1. Writing Code

We can write a Java Program using the editors like notepad , notepad++,sublime Text , etc.

This Java program file name ends with .java file extension.

We can also use some IDE - Netbeans , Eclipse etc, which has buitin editors that provides smoother programming environment.

2.Compile Code

Java Compiler translates the java program into byte codes .

This Byte code is platform independent and intermediate code which can be understood by JVM

Using the javac command .java file will be converted into .class file - Bytecode.

3.Class Loader

Class loaders are responsible for loading Java classes during runtime dynamically to the JVM.

Java ClassLoader is an abstract class.

It belongs to a **java.lang** package

4.Bytecode Verifier

Bytecode verifier ensures that the bytecode are valid and they do not violate java's security restrictions.

Java enforces such security because java code arriving over the network should not be able to cause damage to local file system.

5. Execute Program

Java interpreter inside the JVM converts each line of the bytecode into executable machine code and passed it to the OS/Hardware.