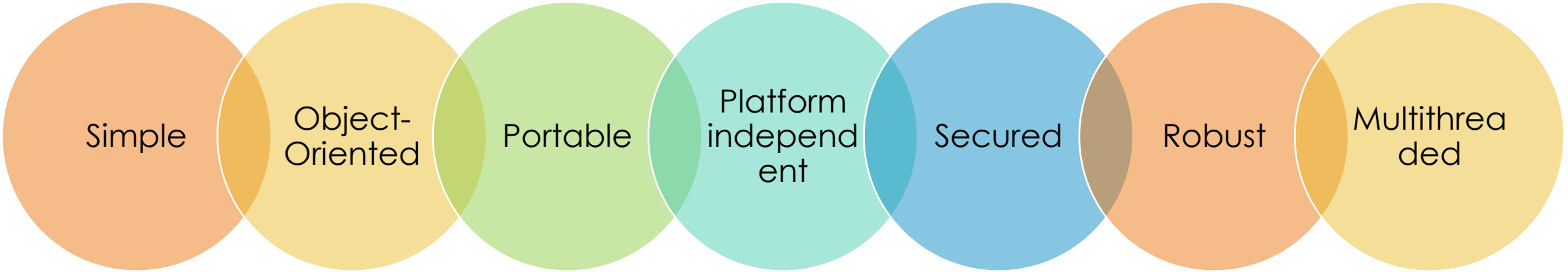
INTRODUCTION TO

A Powerful Programming Language

JAVA BASICS

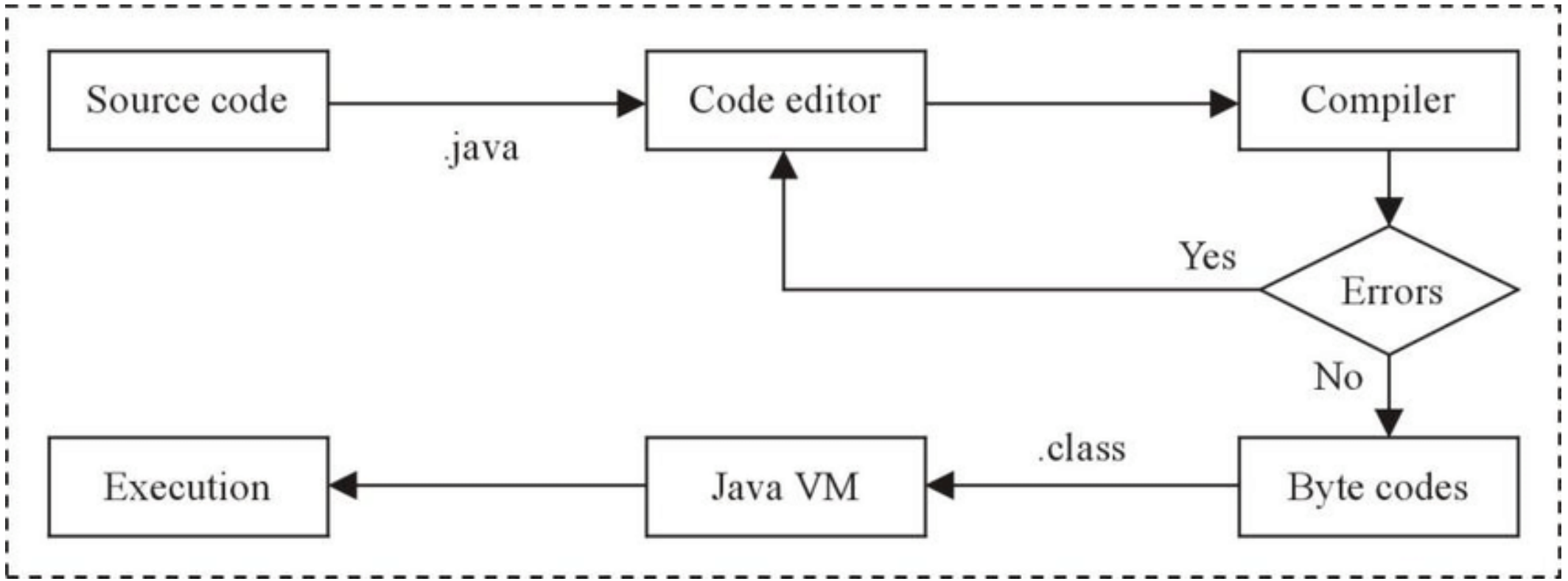
- Java is one of the most popular and widely used programming language and platform.
- A platform is an environment that helps to develop and run programs written in any programming language.
- **Java** is object-oriented programming language.
- It is intended to let application developers ***write once, run anywhere* (WORA)**,
- meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

JAVA FEATURES

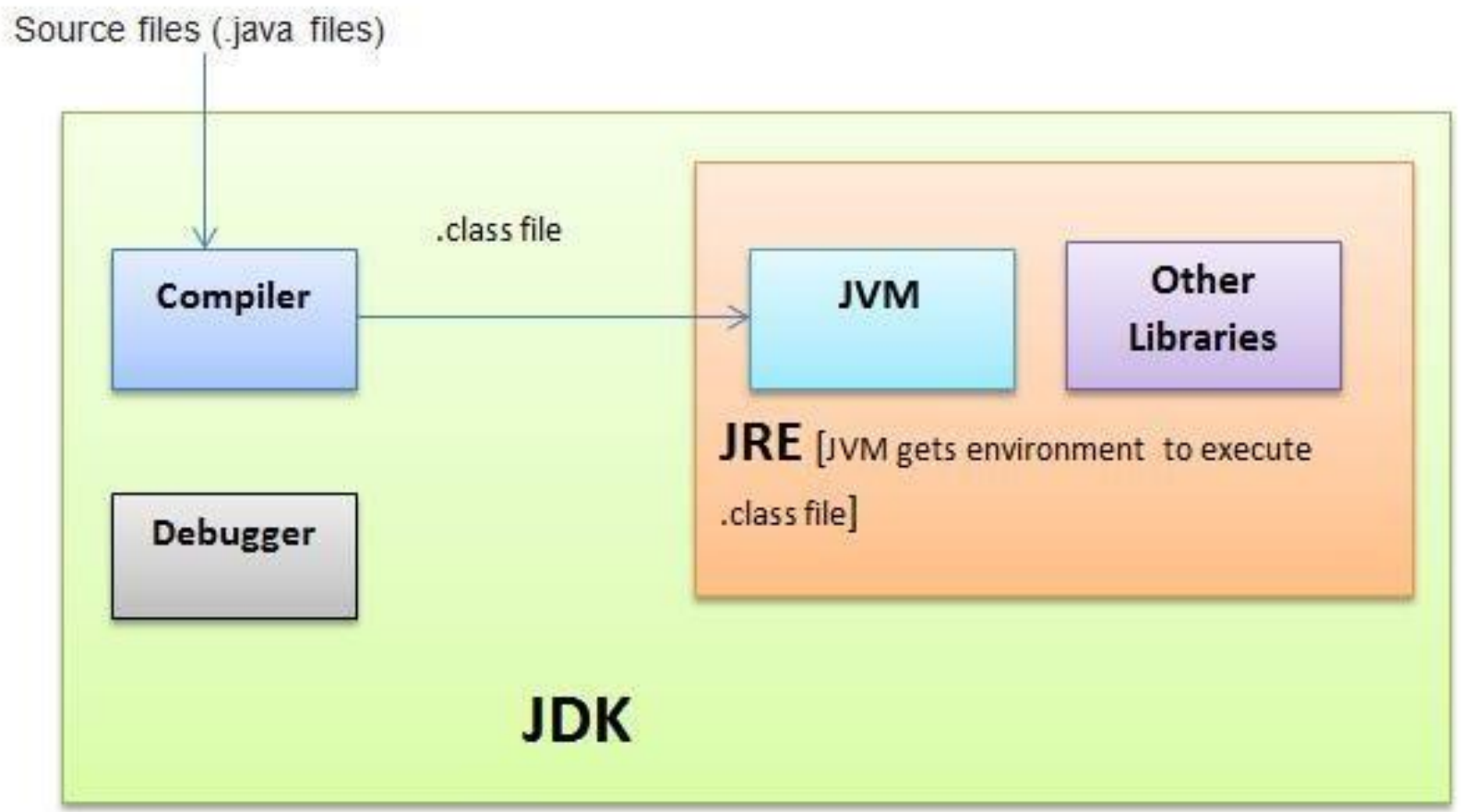


JAVA PLATFORM INDEPENDENT

- **Platform independent** language means once compiled you can execute the program on any **platform** (**OS**).
- The Java compiler converts the source code to bytecode, which is an intermediate Language.
- The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.
- **Java is platform independent but JVM is platform dependent.**


Source code
.java
Code editor
Compiler
Yes
Errors
No
Execution
Java VM
.class
Byte codes

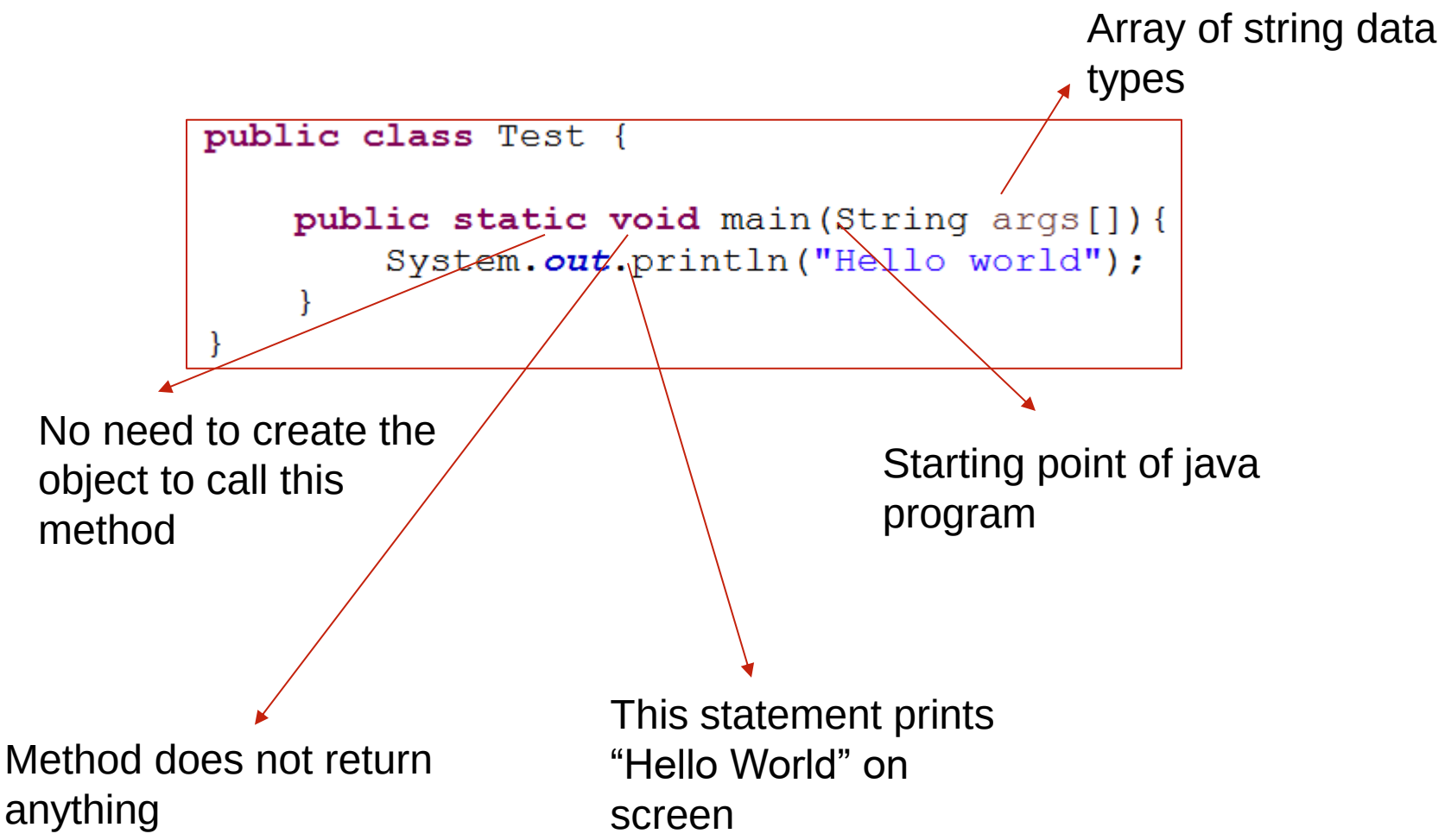
JDK, JRE AND JVM



JDK, JRE, JVM

- JDK (Java Development Kit) is a software development kit required to develop applications in Java. When you download JDK, JRE is also downloaded with it.
- In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger, etc).
- JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.
- JRE is the superset of JVM.
- JVM (Java Virtual Machine) is an abstract machine that enables your computer to run a Java program.
- When you run the Java program, Java compiler first compiles your Java code to bytecode. Then, the JVM translates bytecode into native machine code (set of instructions that a computer's CPU executes directly).

HELLO WORLD PROGRAM



VARIABLES

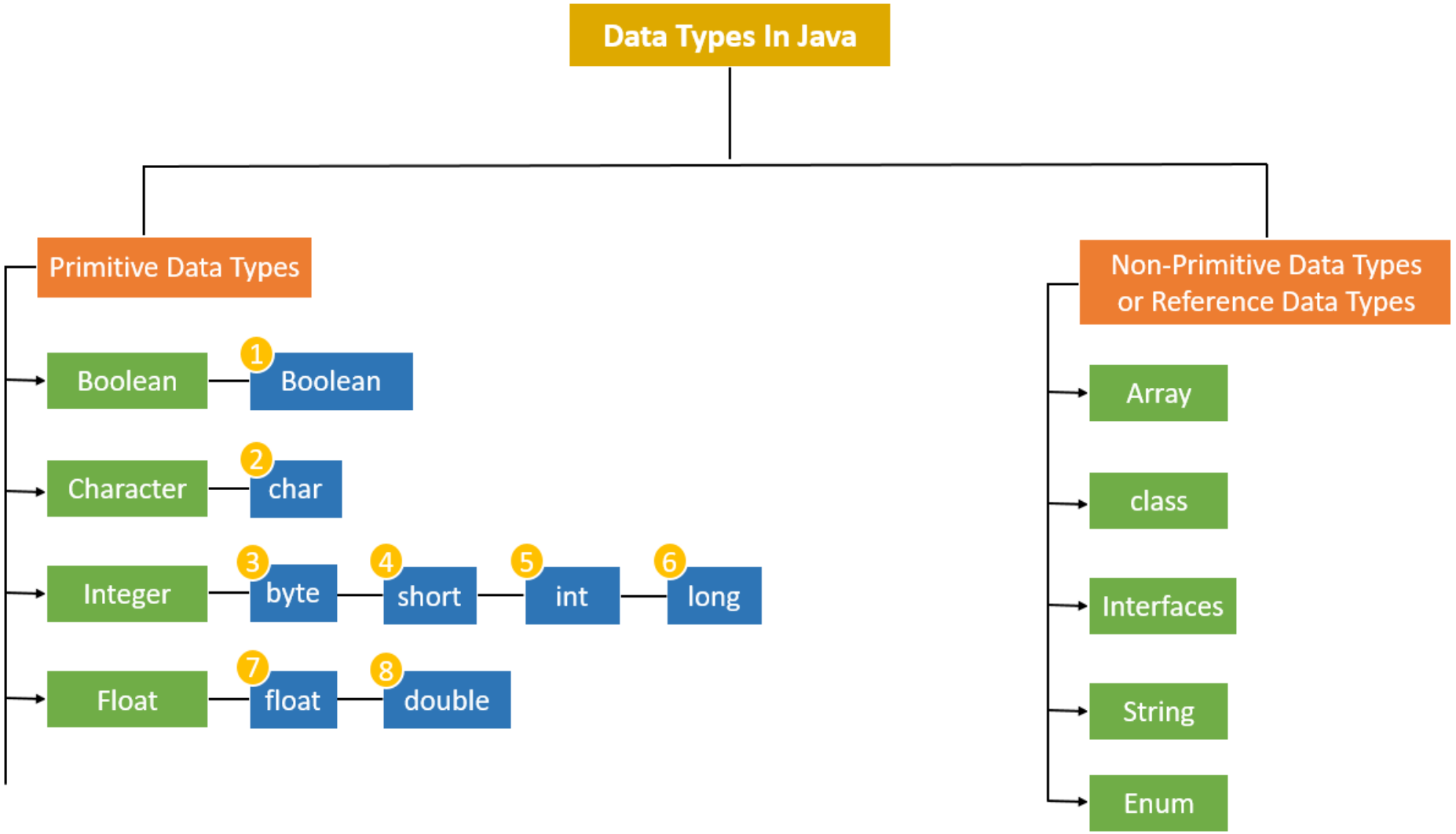
- A variable in Java is a container that holds data that can be changed during the execution of a program. Variables have three main components:
- **Type:** Specifies the type of data the variable can hold.
- **Name:** The identifier used to refer to the variable.
- **Value:** The data held by the variable.

Variable declaration:

```
int age;
String name;
double salary;
```

Variable initialization:

```
int age = 25;
String name = "John";
double salary = 50000.0;
```


Data Types In Java
Primitive Data Types
Boolean
1
Boolean
Character
2
char
Integer
3
byte
4
short
5
int
6
long
Float
7
float
8
double
Non-Primitive Data Types or Reference Data Types
Array
class
Interfaces
String
Enum

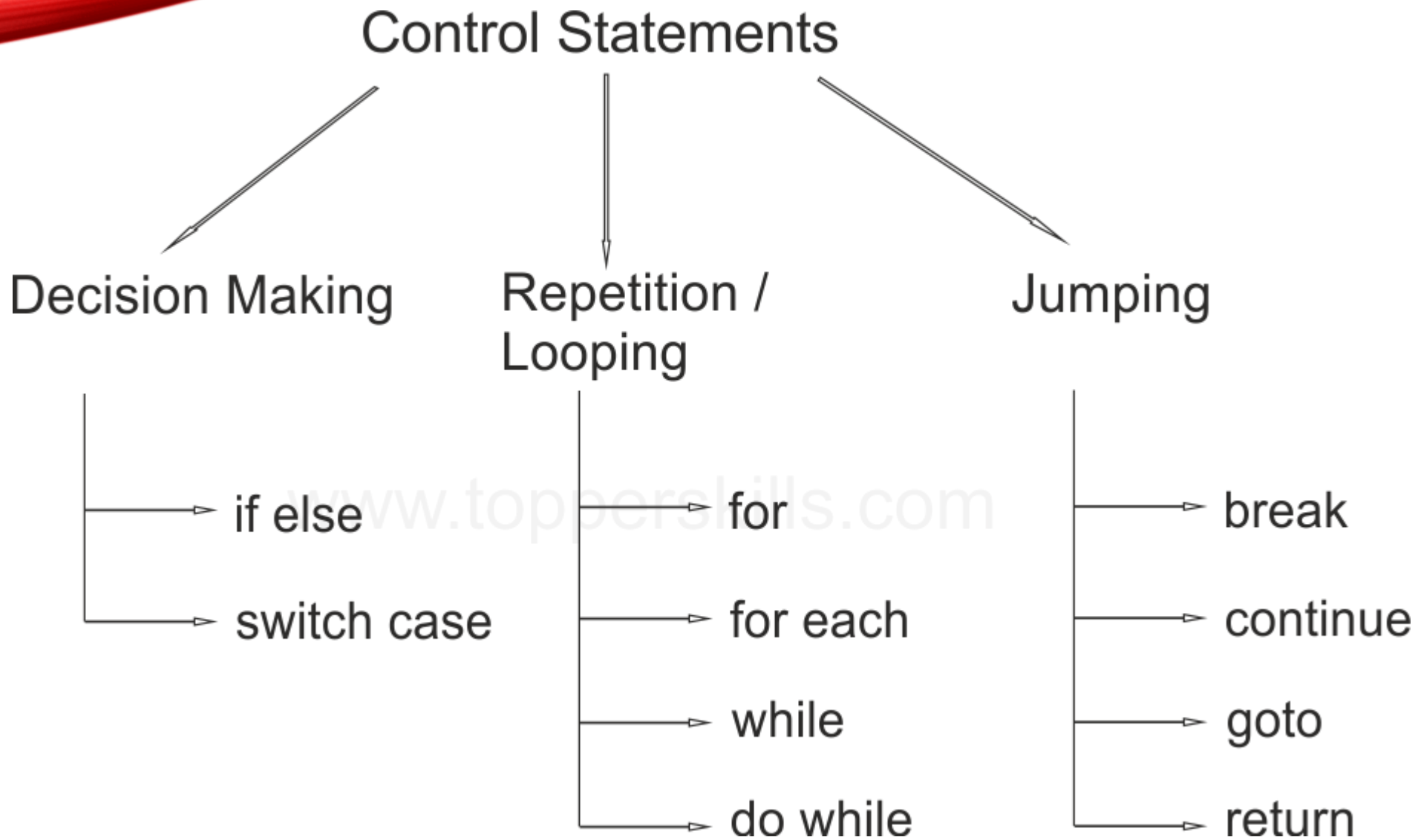
TYPECASTING

- Widening Casting(Implicit)

- Narrowing Casting(Explicitly done)

OPERATORS

1. Arithmetic Operators : +, -, *, /, %
2. Assignment Operators : =, +=, -=, *=, /=, %=, &=, |=, ^=
3. Relational Operators : >, <, ==, >= , <= , !=
4. Unary Operator : ++, --, !
5. Logical Operators : &&, | |, !
6. Bitwise Operator : &, |, =, ^, >>, <<
7. Ternary Operator: ?:



ARRAY

- Array in Java are a group of like-typed variables that are referred to by a common name. Array is a collection of homogenous elements that have **contiguous** memory location.
- The **size** of an array must be specified by an **int** value and not long or short.
- Array can contain primitive data types as well as objects of a class depending on the definition of array.
- In case of primitive data types, the actual values are stored in **contiguous** memory locations.
- In case of objects of a class, the actual objects are stored in heap area.

ARRAY

- Creating and using arrays in Java involves several steps: declaration, instantiation, initialization, and usage.
- Array Declaration:

```
int[] numbers;
String[] names;
double[] values;
```

- Instantiate an Array:

```
numbers = new int[5];
names = new String[3];
values = new double[4];
```

WORKING WITH ARRAY

```
numbers[0] = 1;
numbers[1] = 2;
numbers[2] = 3;

names[0] = "John";
names[1] = "Jane";
names[2] = "Doe";

values[0] = 1.1;
values[1] = 2.2;
values[2] = 3.3;
```

Access Data in variables:

```
int firstNumber = numbers[0];
String firstName = names[0];
double firstValue = values[0];
```

Access Data in variables:

```
int firstNumber = numbers[0];
String firstName = names[0];
double firstValue = values[0];
```

LOOP THROUGH AN ARRAY

```
for (int i = 0; i < numbers.length; i++) {
    System.out.println("Element at index " + i + ": " + numbers[i]);
}
```

```
for (String name : names) {
    System.out.println("Name: " + name);
}
```

Enhanced For Loop

```
int length = numbers.length;  // Get the length of the array
System.out.println("Length of numbers array: " + length);
```

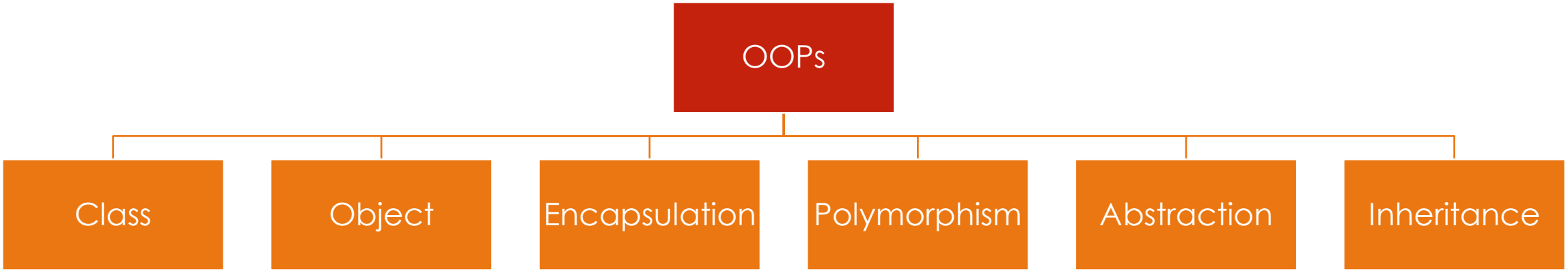
2D ARRAY

```
public class TwoDArray {

    public static void main(String[] args) {

        //int arr[][]=new int[3][3];
        int arr[][]= {{1,2,3},{2,3},{6,7,9,10}};

        System.out.println("Print Array");
        for(int row=0;row<arr.length;row++) {
            for(int col=0;col<arr[row].length;col++) {
                System.out.print(arr[row][col]+"\t");
            }
            System.out.println();
        }
        System.out.println("Print using for each loop");
        for(int row[]:arr) {
            for(int col:row) {
                System.out.print(col+"\t");
            }
            System.out.println();
        }
    }
}
```

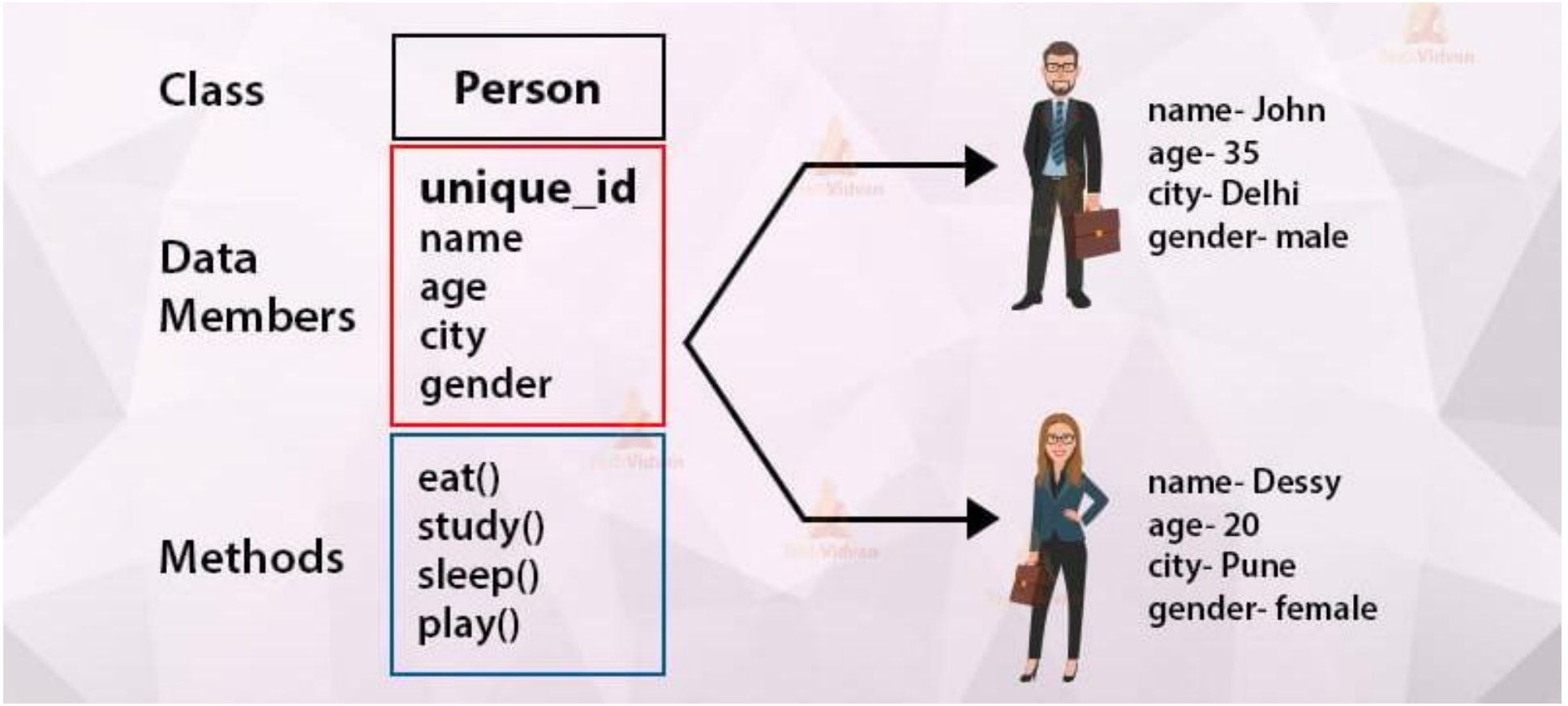
ACTIVITY: USE SCANNER FOR ALL

- Write a program to display table of 10.
- Write a program to display Fibonacci series of 10 numbers (0,1,1,2,3,5,8,13,21).
- Write a program to check whether a given number is odd or even.
- Write a program to find largest of two numbers.
- Write a java program to create an array of numbers and display the original array in sorted order.
- Write a java program to create an array of numbers and display the original array in reverse order.

OOPS


OOPs
Class
Object
Encapsulation
Polymorphism
Abstraction
Inheritance

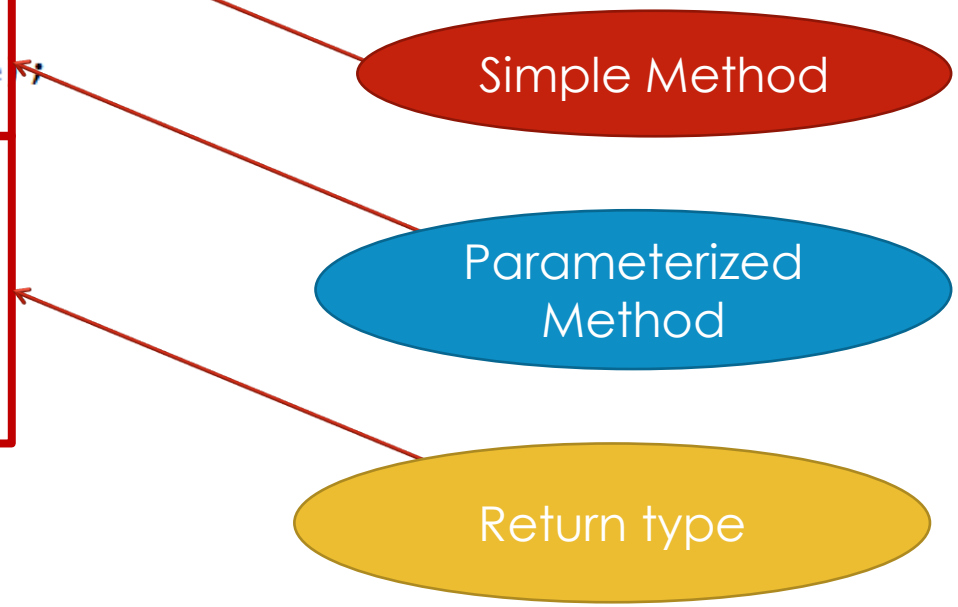
CLASS & OBJECT



```java
public class Student {

    // State
    int id;
    String name;
    String email;
    String address;
    // Behavior
    public void setData(int i, String n, String e, String a) {
        id = i;
        name = n;
        email = e;
        address = a;
    }
    public void speak() {
        System.out.println(name + " is speaking");
    }
    public void eat() {
        System.out.println(name + " is eating");
    }
}
```

METHODS

```
public class Methods1 {

    public void print() {
        System.out.println("Print Method Called");
    }
    public void fullName(String fname,String lname) {
        System.out.println("Welcome "+fname+" "+lname);
    }
    public int cube(int n) {
        return n*n*n;
    }
    public boolean checkValidity(int age) {
        return age>=18;
    }

    public static void main(String[] args) {
        Methods1 obj= new Methods1();
        obj.print();
        obj.fullName("Sonam", "soni");
        System.out.println("Cube of: "+obj.cube(4));
        System.out.println("User Validity "+obj.checkValidity(34));
    }
}
```



CONSTRUCTOR

- Constructors are used to initialize the object's state. Like methods, a constructor also contains **collection of statements** that are executed at time of Object creation.
- Constructors are used to assign values to the instance variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor).
- Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the same class.
- There are two type of constructor in Java:
  - No-argument constructor : A constructor is called "Default Constructor" when it doesn't have any parameter.
  - Parametrized constructor : A constructor which has a specific number of parameters is called a parameterized constructor.

DEFAULT & PARAMETRIZED CONSTRUCTOR

```java
class Employee {
    /* instance varibale */
    int id;
    String name;
    float salary;
    void display() {
        System.out.println(this.id + " " + this.name + " " + this.salary);
    }
}
public class Test {
    public static void main(String args[]) {
        //Object creation using default constructor
        Employee emp1= new Employee();
        emp1.display();

    }
}
```

```java
class Employee {
    /* instance varibale */
    int id;
    String name;
    float salary;
    // Parametrized constructor
    Employee(int id, String name, float salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    void display() {
        System.out.println(this.id + " " + this.name + " " + this.salary);
    }
}
public class Test {
    public static void main(String args[]) {
        //Object creation
        Employee emp1 = new Employee(101,"John",20000);
        emp1.display();
        Employee emp2 = new Employee(102,"Jack",10000);
        emp2.display();
    }
```

CONSTRUCTOR VS METHOD

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.
Constructor overloading is supported.	Method overloading is supported.

ACTIVITY: STUDENT GRADES MANAGEMENT SYSTEM

- create a Student class with the following attributes:
  - name (String)
  - grades (an array of integers representing the grades of the student)
- Implement the following methods:
  - A constructor that initializes the student's name and grades.
  - calculateAverage() method that returns the average grade of the student.
  - displayInfo() method that prints the student's name and their average grade.

THIS KEYWORD

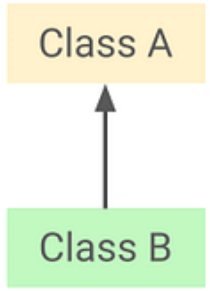
1. this can be used to refer current class instance variable.
2. this() can be used to invoke current class constructor.
3. this can be used to invoke current class method (implicitly)

```
// Parametrized constructor
Employee(int id, String name) {
    this.id = id;
    this.name = name;
}

// Parametrized constructor
Employee(int id, String name, float salary) {
    this(id,name);
    this.salary = salary;
}

void first() {
    System.out.println(this.id + " " + this.name);
    this.second();
}

void second() {
    System.out.println(this.salary);
}
```

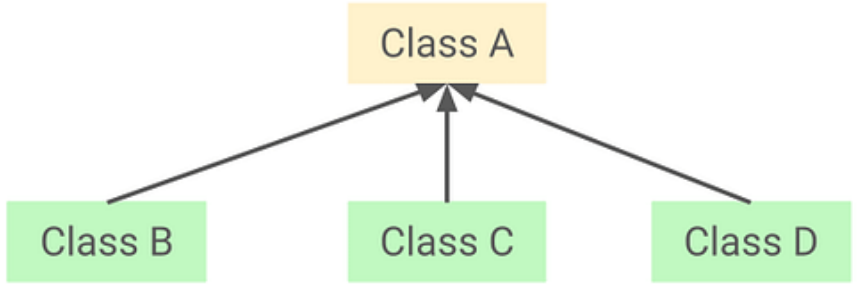
STATIC KEYWORD

- It is used to initialize variable once.
- Also used to call method directly without creating an object of class.
- Actually static variable is creating class level variables and methods.

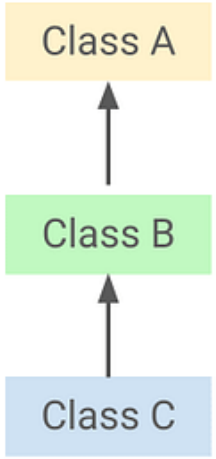
INHERITANCE



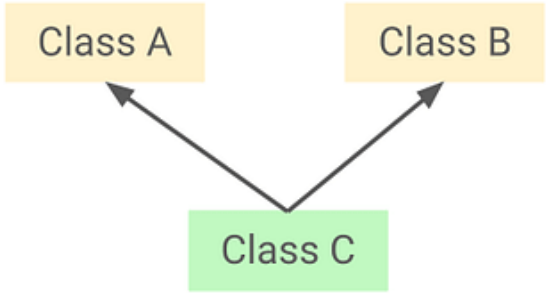
Single Inheritance



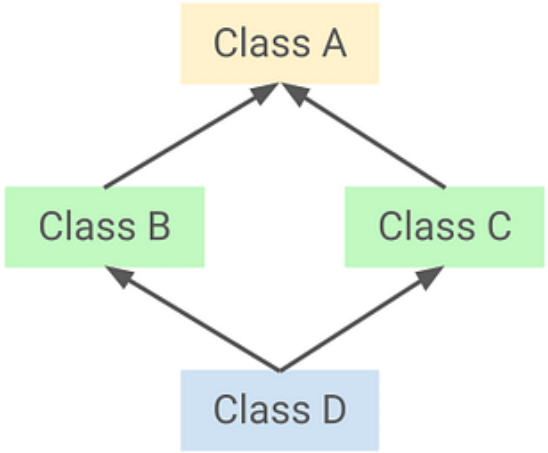
Hierarchical inheritance



Multilevel Inheritance



Multiple Inheritance

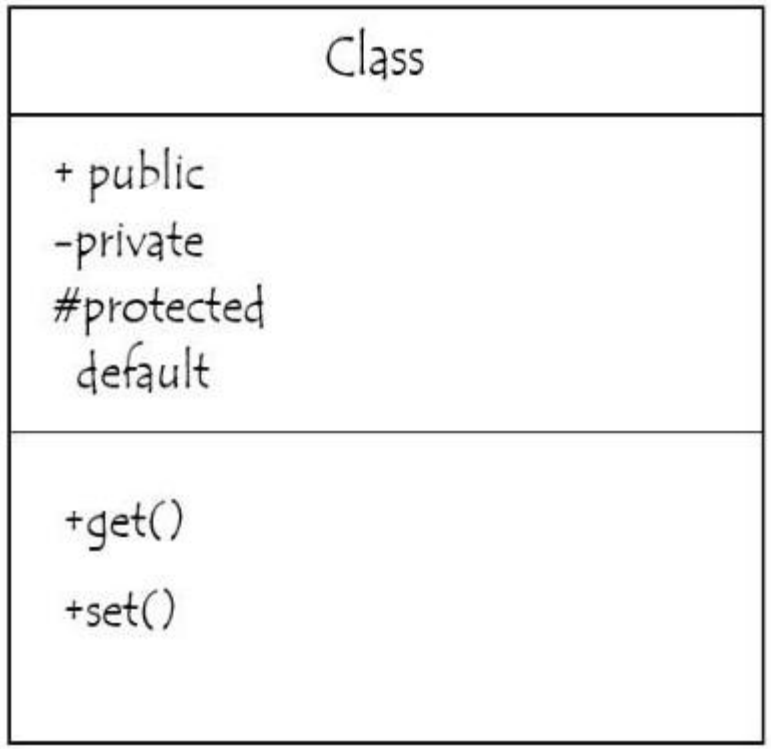


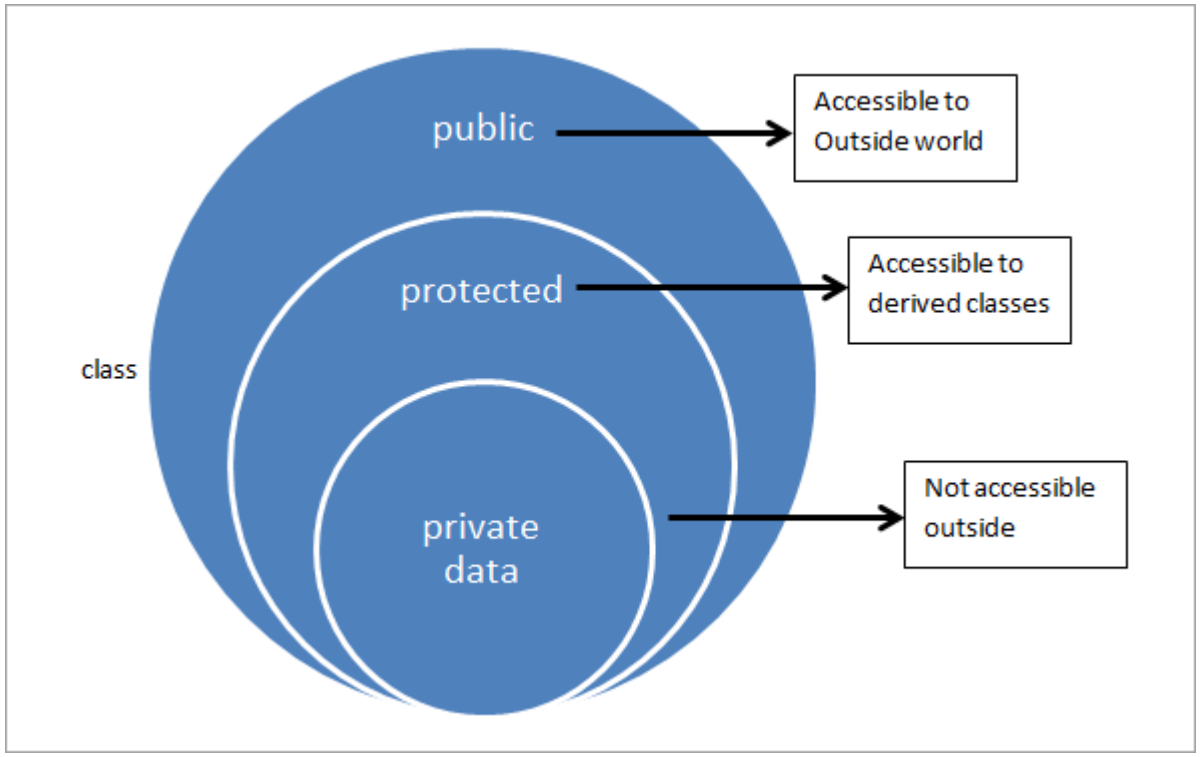
Hybrid Inheritance

SUPER KEYWORD

- It is used to call parent class constructor
- Used to pass data from parent to child.
- Used to call methods from child to parent.
  - Super.parentclassmethod()

ENCAPSULATION





LETS IMPLEMENT IT

- Make all the customer fields as private. Write getter and setters for customer class for all fields.
- Create customer objects using setter methods.
- Display the array of customer objects.

ABSTRACTION

- An abstract class is a class that is declared with abstract keyword.
- An abstract class can never be instantiated. If a class is declared as abstract then the sole purpose is for the class to be extended.
- A class cannot be both abstract and final. (since a final class cannot be extended).
- An abstract method is a method that is declared without an implementation.

DATA ABSTRACTION

- Implemented in 2 ways
  - Abstract class & Abstract methods
  - Interface

POLYMORPHISM