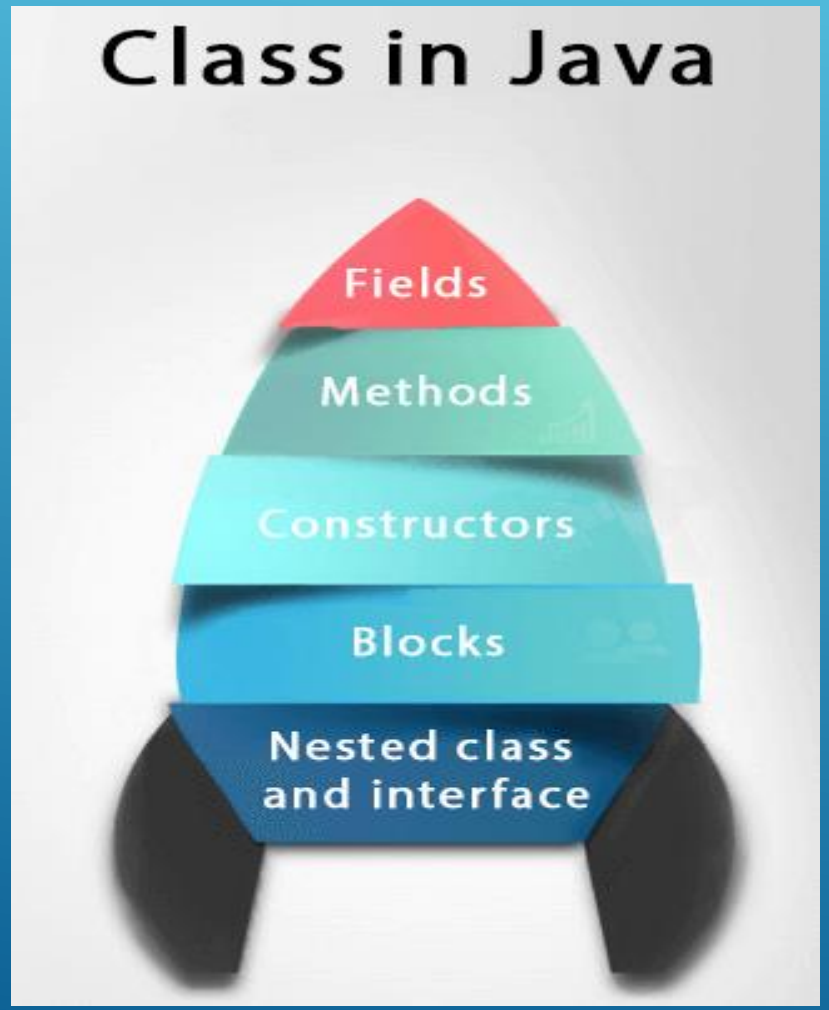
TODAY'S AGENDA

- What is Class and Object?
- What is a Method?
- Method Overloading
- What is Constructor?
- Static and Non-Static
- OOPs (Object Orientated Programming)
  - Inheritance
  - Polymorphism (Method Overriding)

WHAT IS A CLASS IN JAVA?

- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

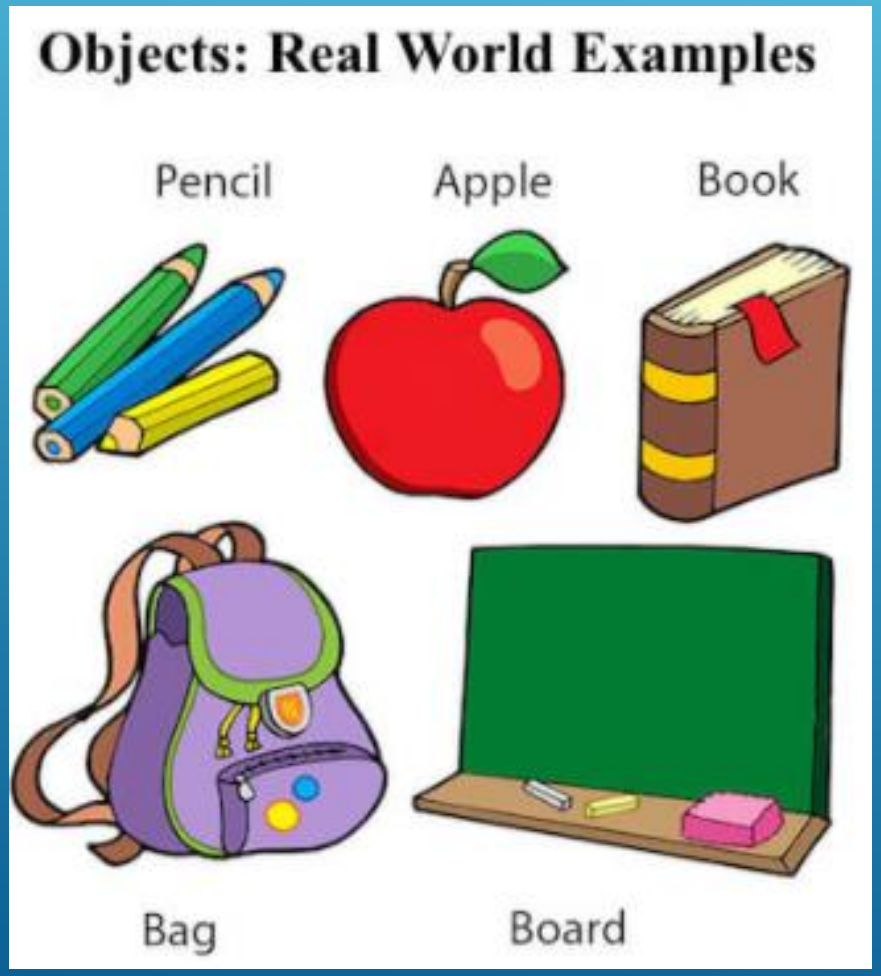


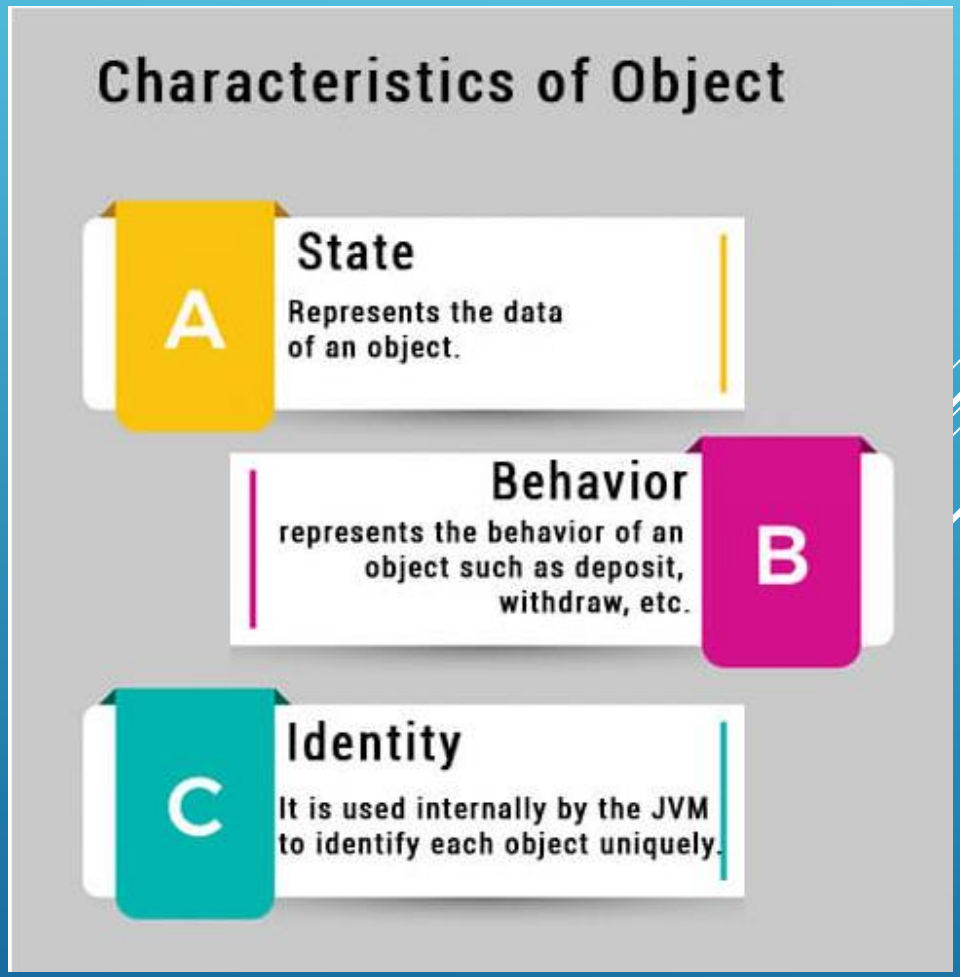
Syntax to declare a class:

```
class <class_name>{
    field;
    method;
}
```

WHAT IS AN OBJECT IN JAVA?

- An entity that has state and behaviour is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical.





Example of class:

```
public class Main {
        int x = 5;
}
```

Example of object:

```
public class Main {
        int x = 5;
public static void main(String[] args) {
        Main myObj = new Main();
        System.out.println(myObj.x);
        }
}
```

WHAT IS METHODS ?

- A method is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- Methods are used to perform certain actions, and they are also known as functions.
- Why use methods? To reuse code: define the code once, and use it many times.
- A method must be declared within a class. It is defined with the name of the method, followed by parentheses ().
- Java provides some pre-defined methods, such as System.out.println(), but you can also create your own methods to perform certain actions:
- Example:

```
public class Main {
        static void myMethod() {
         // code to be executed
        }
}
```

HOW TO CALL METHODS ?

- To call a method in Java, write the method's name followed by two parentheses () and a semicolon;
- Example :

```
public class Main {
        static void myMethod() {
        System.out.println("I just got executed!");
        }
public static void main(String[] args) {
        myMethod();
        }
}
```

METHOD PARAMETERS

(PARAMETERS AND ARGUMENTS)

- Information can be passed to methods as parameter. Parameters act as variables inside the method.
- Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.
- Example:

```
public class Main {
        static void myMethod(String fname) {
        System.out.println(fname + " Tester");
        }
public static void main(String[] args) {
        myMethod(“Test1");
        myMethod("Test2");
        myMethod("Test3");
        }
}
Output:// Test1 Tester // Test2 Tester // Test3 Tester
```

METHOD OVERLOADING

- In method overloading, multiple methods can have the same name with different parameters
- Multiple methods can have the same name as long as the number and/or type of parameters are different.

Example:

```
int myMethod(int x)
float myMethod(float x)
double myMethod(double x, double y)
```

Example:

```
static int plusMethodInt(int x, int y) {
 return x + y;
}
static double plusMethodDouble(double x, double
y) {
 return x + y;
}
public static void main(String[] args) {
 int myNum1 = plusMethodInt(8, 5);
 double myNum2 = plusMethodDouble(4.3, 6.26);
System.out.println("int: " + myNum1);
System.out.println("double: " + myNum2);
}
```

CONSTRUCTORS

- A constructor in Java is a special method that is used to initialize objects.
- The constructor is called when an object of a class is created.
- It can be used to set initial values for object attributes
- Constructor name must match the class name, and it cannot have a return type (like void).
- Also the constructor is called when the object is created.
- All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you.

Example:

```
// Create a Main class
public class Main {
        int A; // Create a class attribute

// Create a class constructor for the Main class
public Main() {
        a = 5; // Set the initial value for the class attribute a
}

public static void main(String[] args) {
        Main myObj = new Main(); // Create an object of class Main (This will call
        the constructor)
        System.out.println(myObj.a); // Print the value of a
        }
}
```

CONSTRUCTOR PARAMETERS

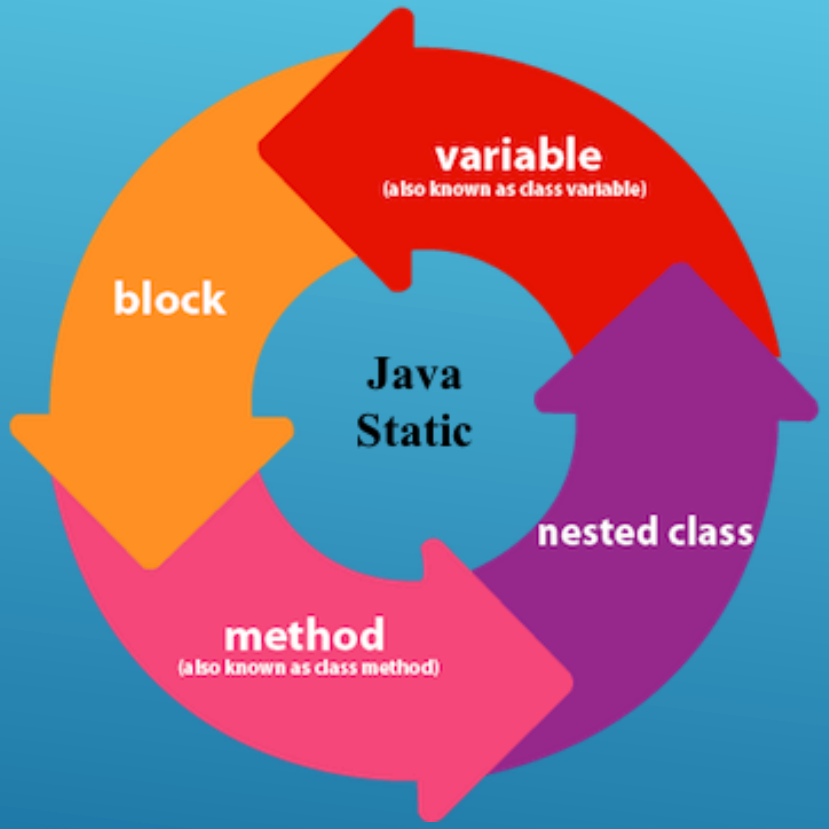
- Constructors can also take parameters, which is used to initialize attributes.

Example:

```
public class Main {
        int x;
public Main(int y) {
        x = y;
}
public static void main(String[] args) {
        Main myObj = new Main(5);
        System.out.println(myObj.x);
        }
}
```

STATIC AND NON-STATIC

‘Static’ is a Java keyword, it is able to improve “Shareability”.



A static member can be accessed directly by the class name and doesn’t need any object.

A non static member cannot be accessed directly and an object is created to access.

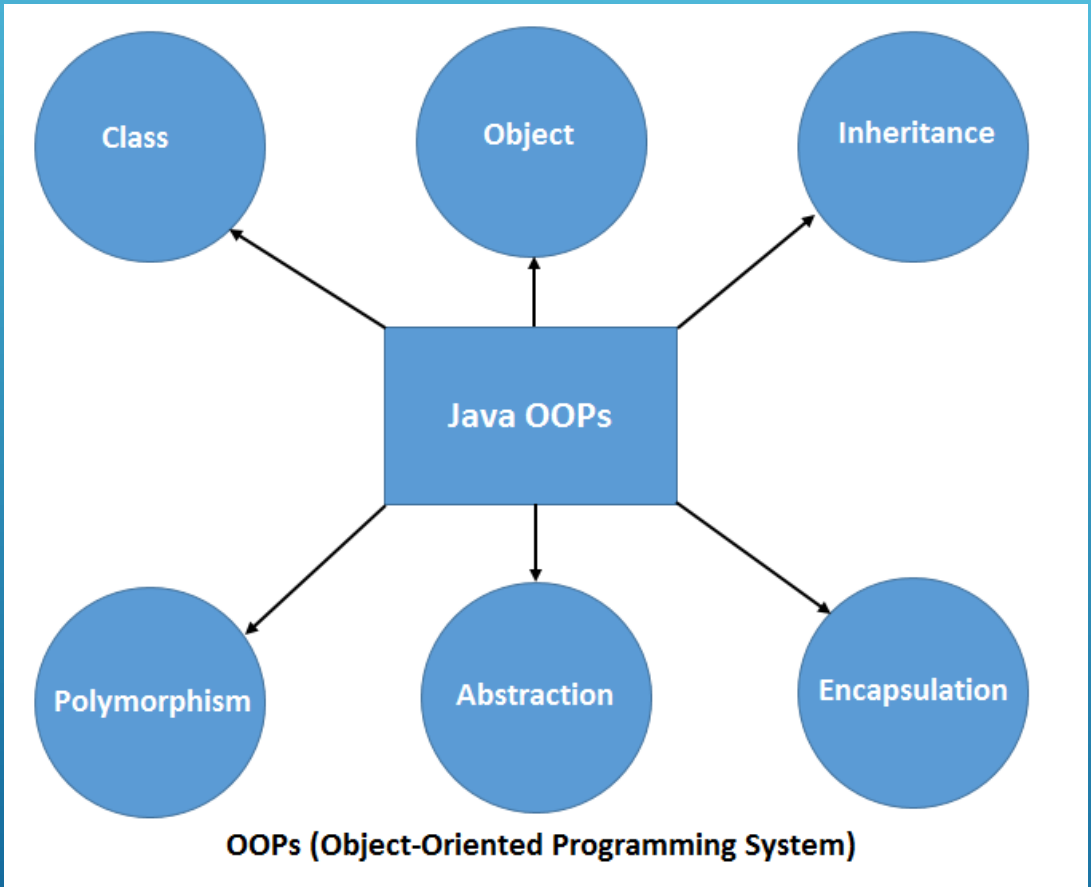
```
public class StaticAndNonStatic {

    String name;
    static int salary;

    public void getName(){
    System.out.println(“non-static method");
}

public static void getSalary(){
    System.out.println(“static method");
}
    public static void main(String[] args) {

        salary = 2500;
        System.out.println(salary);
        getName();

        //1. call by classname:
        StaticAndNonStatic.salary=25000;
        System.out.println(StaticAndNonStatic.salary);
        StaticAndNonStatic.getName();

        //2. How to call non static members- create the
object:
        StaticAndNonStatic obj = new
StaticAndNonStatic();
        obj.getName();
        obj.name = "Tom";
        System.out.println(obj.name);
```

OOPS CONCEPT (OBJECT ORIENTATED PROGRAMMING)

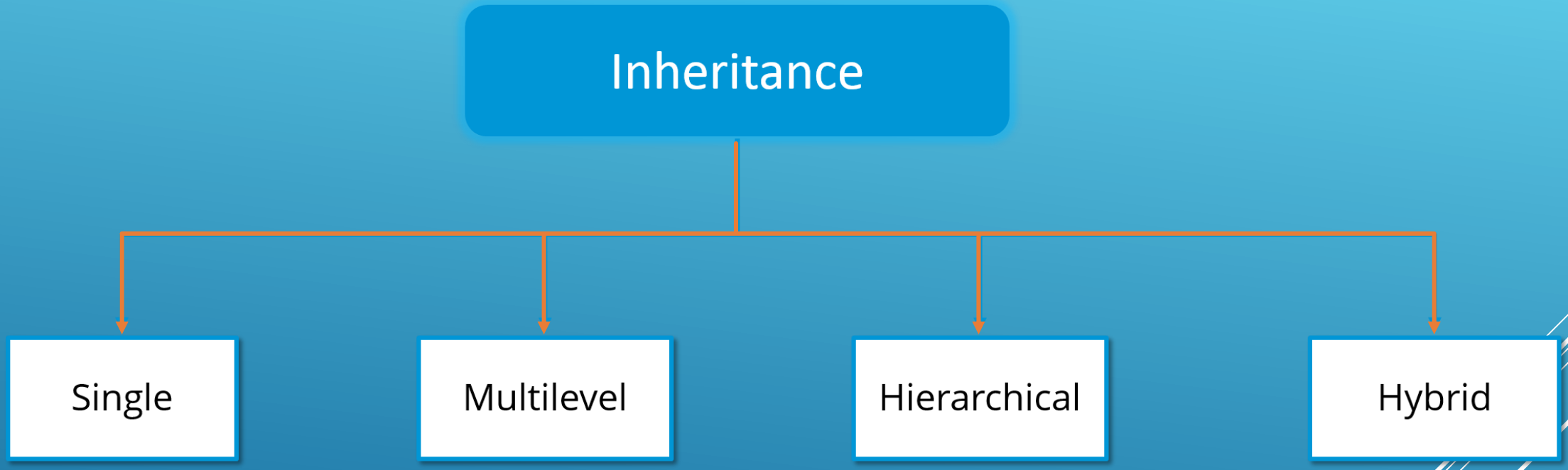
- Object Oriented programming is a programming style which is associated with the concepts like Class, Object, Inheritance, Encapsulation, Abstraction, Polymorphism.
- Object Orientated PL are providing a simplified approach to represent all real world entities in the form of Coding part. Any new method can be added or deleted easily in an object/a class just by adding/deleting it.
- Abstraction is good in OOPL's (Process of hiding data from outside)
- Security is very good in OOPL's
- Shareability is very good in OOPL's (E.g: Sharing Library to multiple programs at a time)
- Reusability is very good in OOPL's

OOP'S CONCEPT (OBJECT ORIENTATED PROGRAMMING)

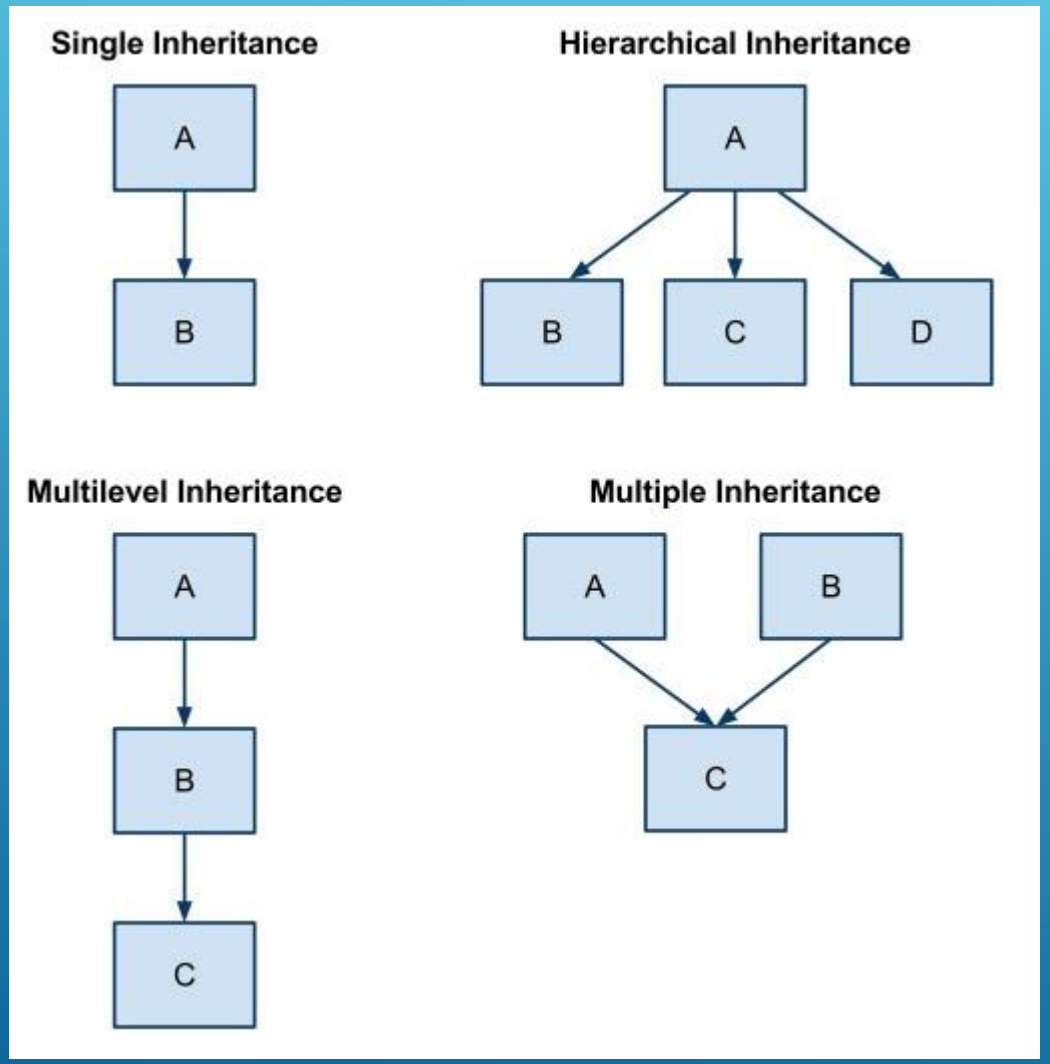
The Building Blocks of Object Orientated Programming are:



TYPES OF INHERITANCE

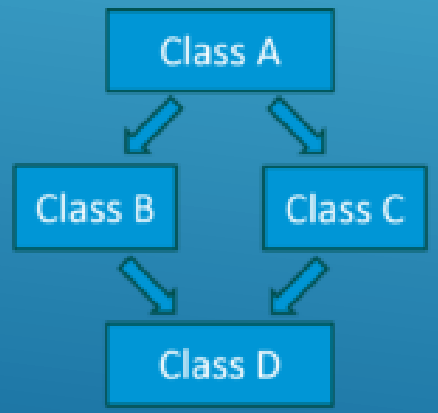


TYPES OF INHERITANCE




```
1  Class A{
2  ---
3  }
4  Class B extends A{
5  ---
6  }
7  Class C extends A{
8  ---
9  }
```

```
1  Class A{
2  ---
3  }
4  Class B extends A{
5  ---
6  }
7  Class C extends B{
8  ---
9  }
```

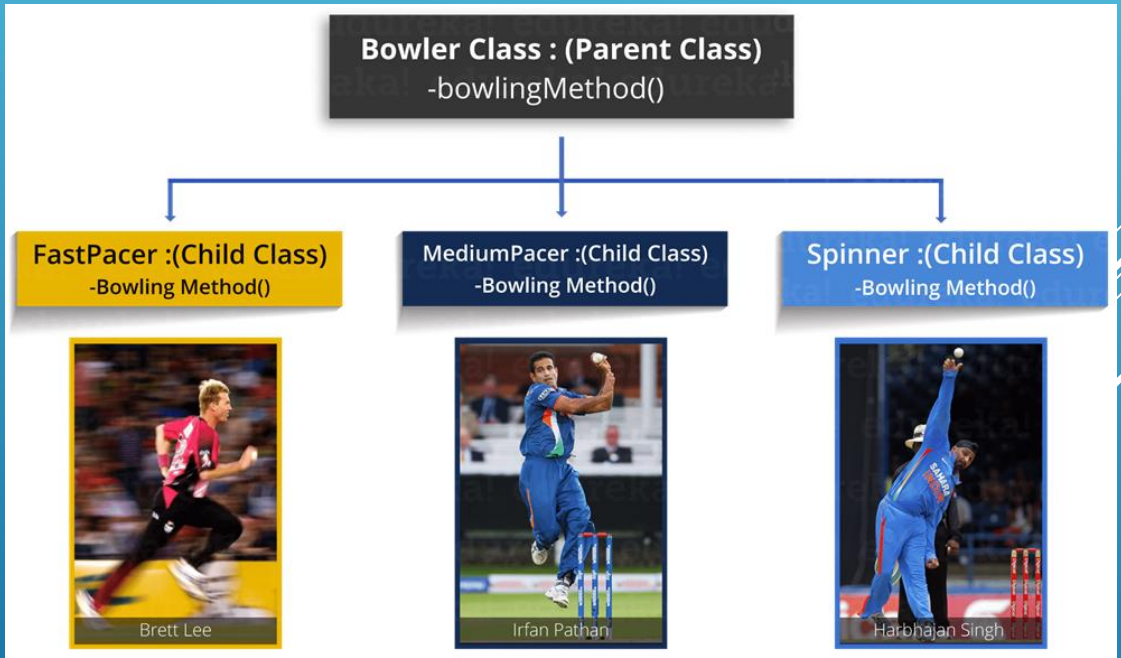


Hybrid

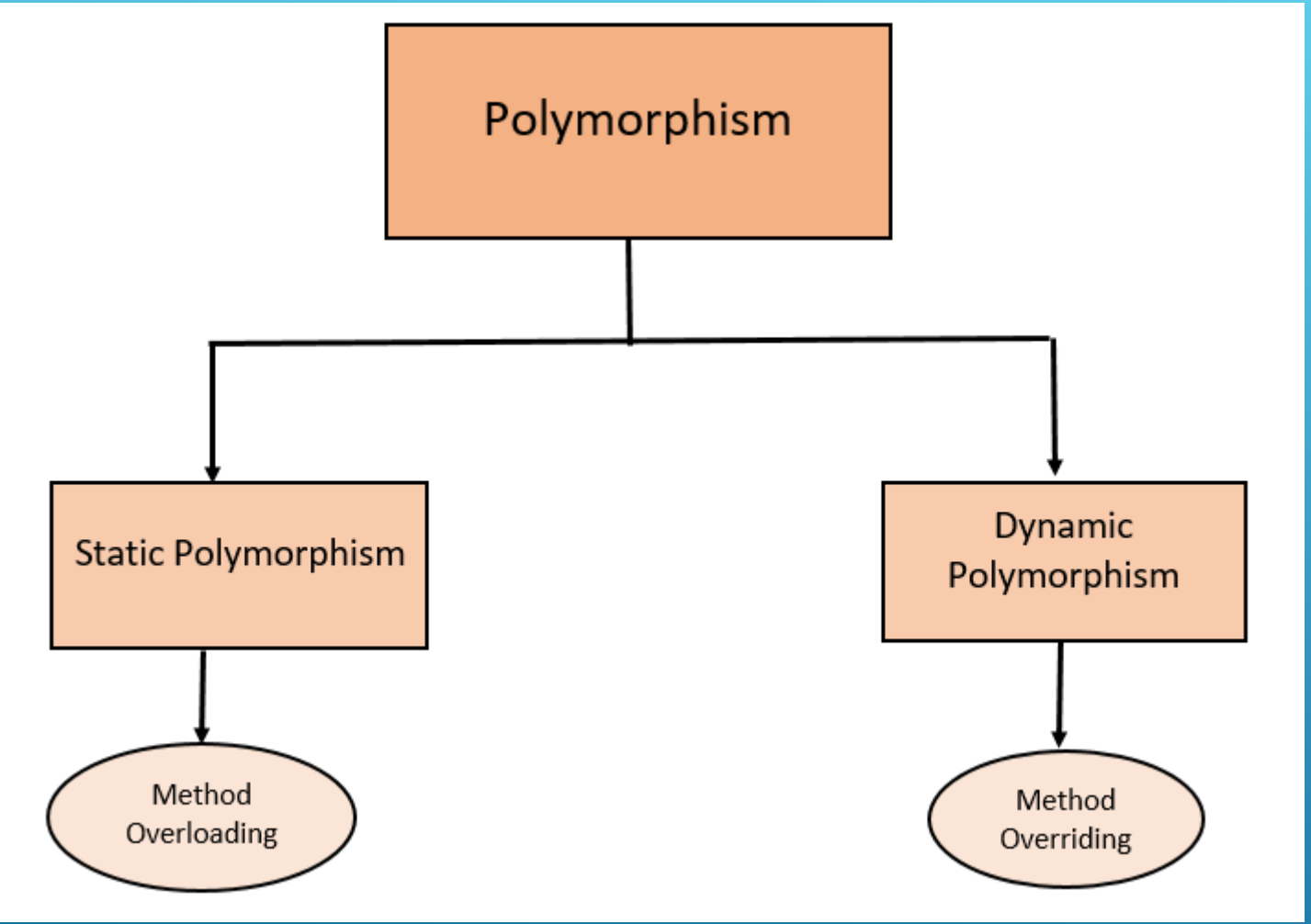
```
1  Class A
2  {
3  ---
4  }
5  Class B extends A {
6  ---
7  }
```

POLYMORPHISM

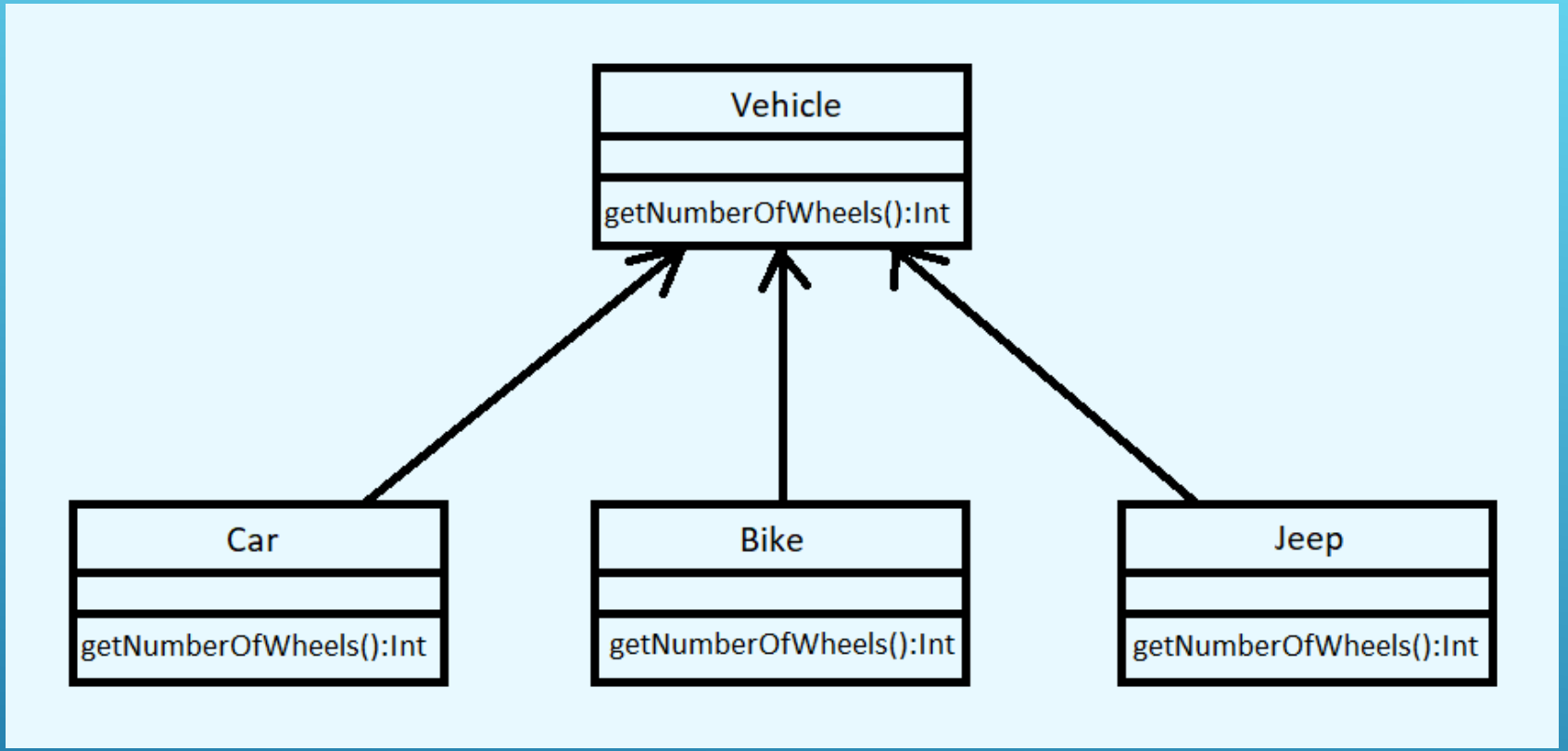
- Polymorphism means taking many forms, where ‘poly’ means many and ‘morph’ means forms. It is the ability of a variable, method or object to take on multiple forms. In other words, polymorphism allows you define one interface or method and have multiple implementations.



TYPES OF POLYMORPHISM



METHOD OVERRIDING




```java
public class Vehicle {

    public void numberOfWheels() {
        System.out.println("Vehicle --- 4 wheels");
    }
}
```

```java
public class Bike extends Vehicle {

    public void numberOfWheels() {
        System.out.println("Bike --- 2 wheels");
    }
}
```

```java
public class RunTest{

    public static void main(String[] args) {

    Bike bike = new Bike();
    bike.numberOfWheels();
    }
}
```

HOMEWORK

OOPs:

- Write a program to implement below OOPs concept:
  - Class
  - Object
  - Inheritance
  - Polymorphism

Deadline: Wednesday Midnight Latest