JAVA ⟶ Platform (JRE)

↓ ↘ OOP

Programming language

★ Types of JAVA application

Mobile App, Enterprise, Web,

Standalone apps

JVM → Java Virtual Machine

⇒ Virtual Application

↳ Doesn't physically exist

↳ Specification that provides a runtime environment in which bytecode can be executed

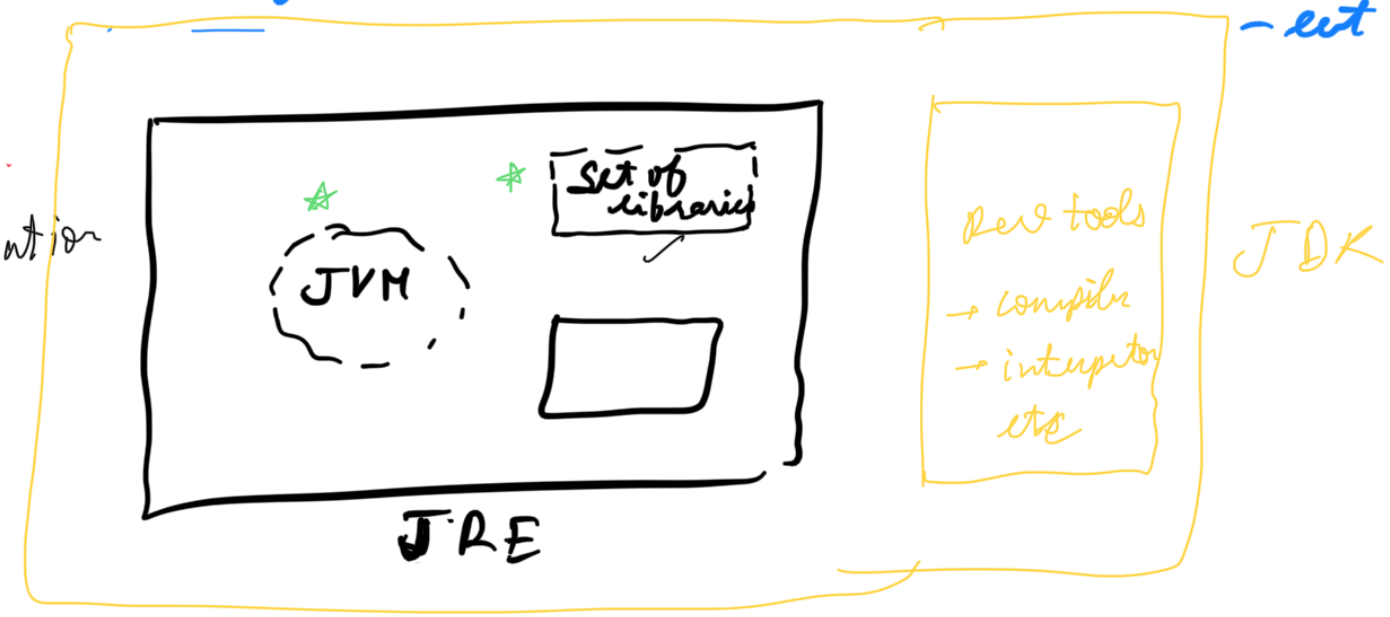
→ Loads Code → Executes Code

→ Verifies the code → Provides Runtime environment

JRE →

↓

Implementation of JVM



JDK JAVA Development Kit (JRE + dev tools)

OOP →

Object Oriented Programming Language

1. Class → blueprint for an object
2. Object → Instance of class
3. Inheritence → Reuse of code
4. Polymorphism
5. Abstraction
6. Encapsulation

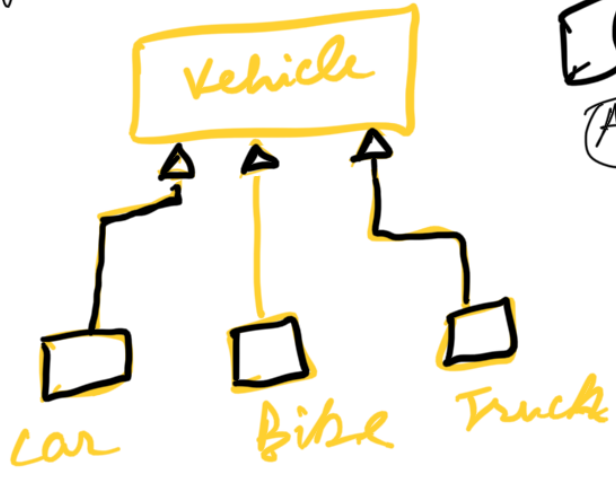
1. Coupling
2. Cohesion
3. Association
4. Composition
5. Aggregation

⇒

```
Car car1 = new Car();
```

```
class Car {
    int regNum;      } variables / properties
    String color;
    → accelerate(); } functions / functionality
}
```

③ Inheritance ⇒



1. Car →
2. Motorbike →
3. Truck →

④ Polymorphism →

→ add(int a, int b) } → ① Function/ Method overloadi

→ add(int a, int b, int c) → overriding

(5) Encapsulation — capsule

↓

wrapping code and data together into single unit.

Combination of medicines packed togeth

JAVA BEAN

(6) Abstraction

Hiding the complexity and showing the functionali -ty

Ex- A phone call

JAVA? → Abstract class
→ Interface

Coupling → Couple — Dependency on other class

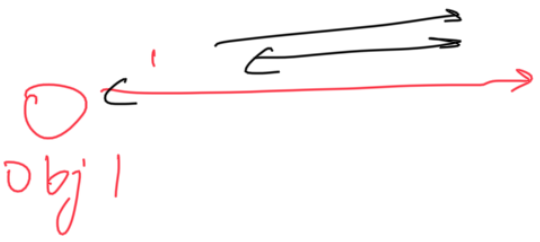
Weak coupling > Strong Coupling

⇒ class has information about other class
→ strong coupling

Interface is a way

cohesion → Weak Coupling → Seperates task into diff parts.

Association → Relationship b/w objects



One to one

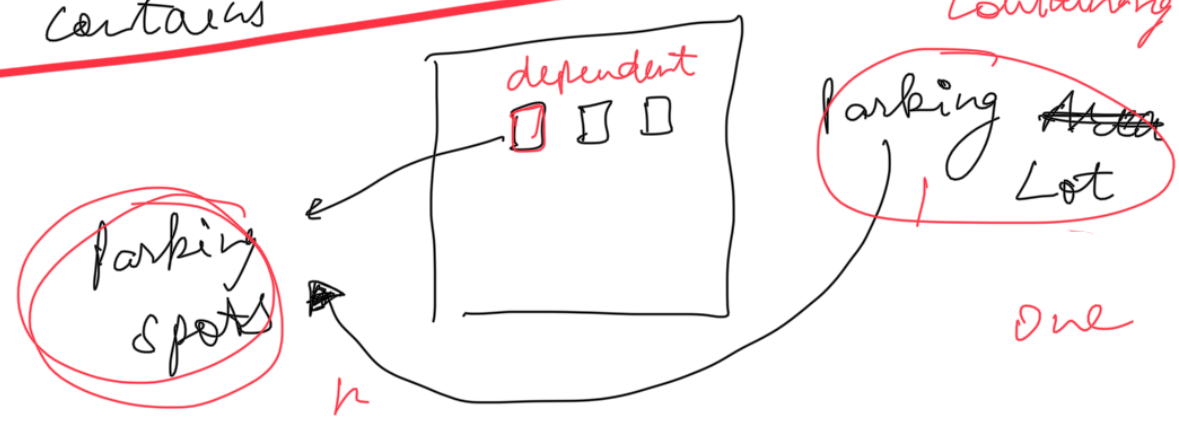
One to many

Many to one

Many to many

way to achieve

Composition Represents the relationship where one obj contains



① Parking Lot {

ⓝ List < Parking Spots >

}

Aggregation → has-a

weak relationship b/w objects

Advantages of OOPs -

1. Reuse of code
2. Development and maintenance easier
3. Data hiding

4. provides the

Constructor → Initialise → JAVA → default constructor

- Method
- gets called when obj is created
- At the time it is called, memory is allocated to obj

2 types of constructor

- Default → No arg

```
Car(){
    a = 10;
}
```

- Parametrized

```
Car(int a){
}
```

Polymorphism → Method

1. By changing no. of args
2. By changing data type

```
add(int a, int b)
add(2.5, 2.3)
```

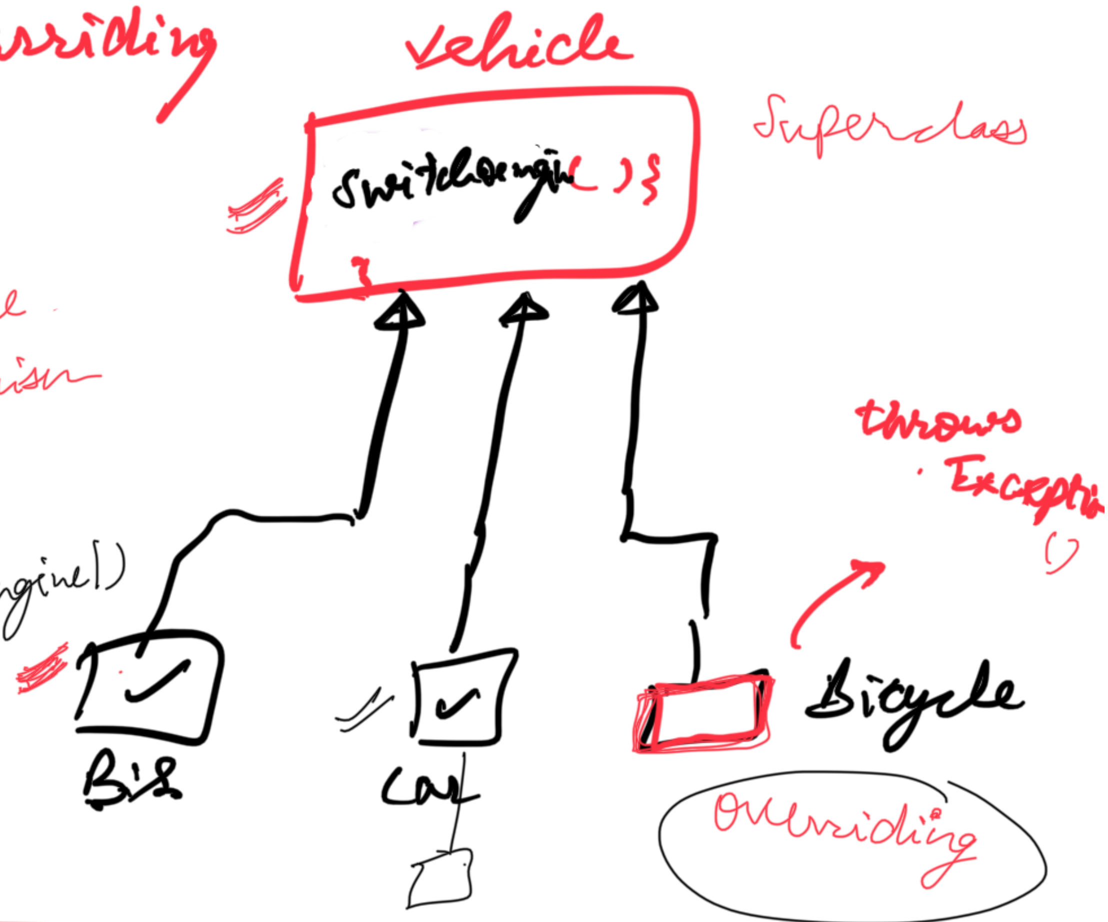
```
int    add(int a, int b)
double add(int a, int b)
```

→ Method overloading

Compile Time error

Function overriding

Used for runtime polymorphism



Bicycle
b.switchEngine()

f- e(){
Super(..);

Super (—, —)

Super()

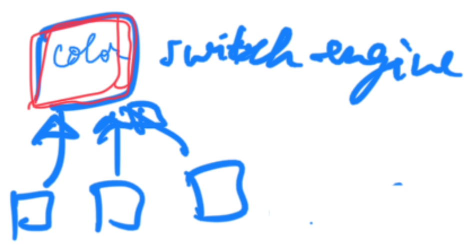
}

Super-

Refer to parent

super.function

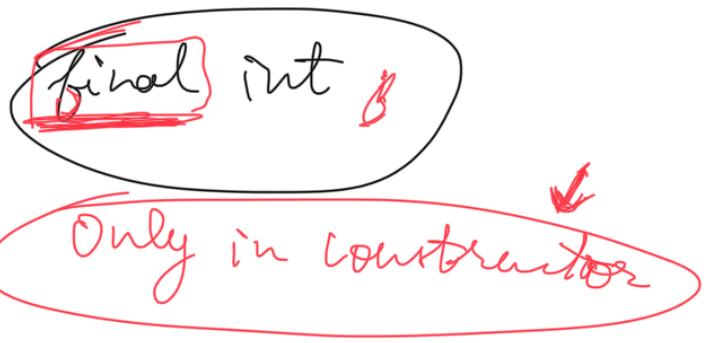
super.color



1. Method
2. Variable
3. Constructor → super() → in the child constructor

Final Keyword →

1. Variable → can't change
2. Method → Override ✗
3. Final Class → You can't extend it
   Inheritence → ✗

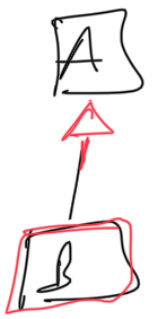


Runtime Polymorphism →

↳ process in which a call to override method is resolved at runtime.

Upcasting

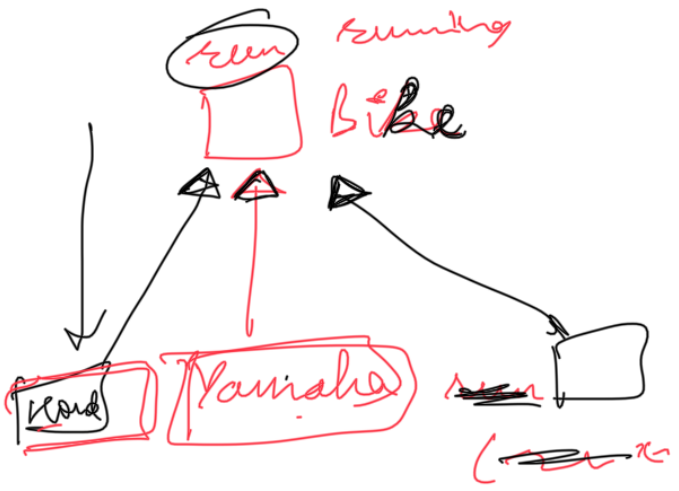
Reference variable of Parent class refers to child class object

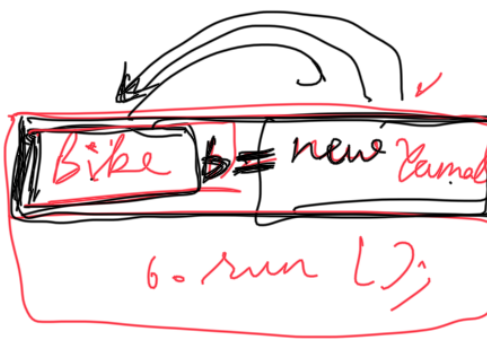


Class B extends A

A a = new B();

a.





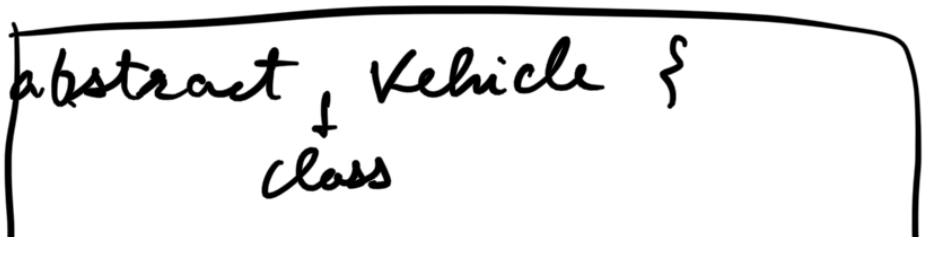
---

1. Abstract class →

abstract class xyz {

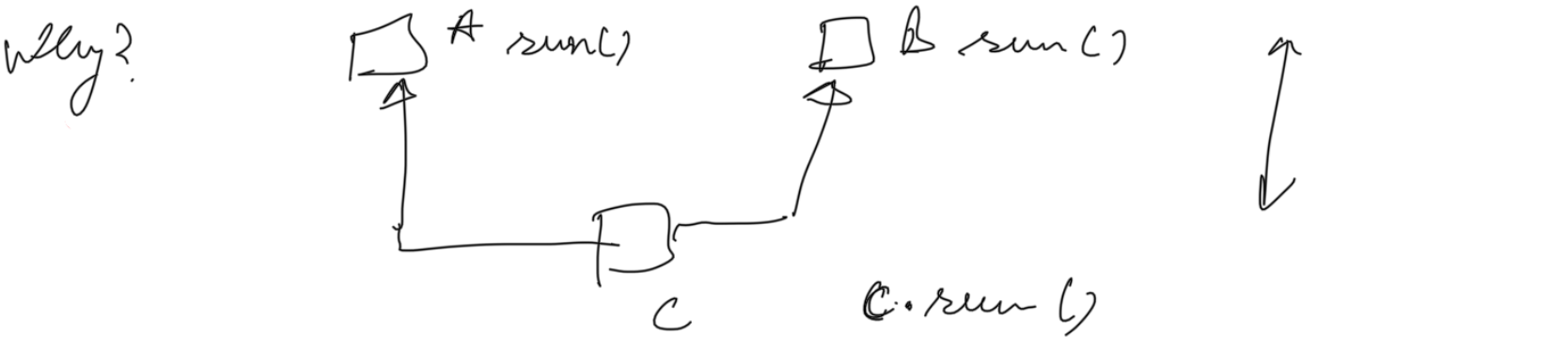
// abstract methods
// non abstract me

extend + method implementation }



abstract void run();

class Honda extend Bike{}



Abstract Class	Interface
1. Abstract methods & Non abstract methods	1. Abstract methods
2. Multiple Inh ✗	2. ✓
3. abstract	3. interface

4. 'extend'
5. private, protected

4. implement
5. public by default

Encapsulation →

① **Packages**

Group of similar type of classes, ↑ ——

→ Built in package
→ User defined

removing naming collision

package ——

(i)

package pack

public class A {

}

import

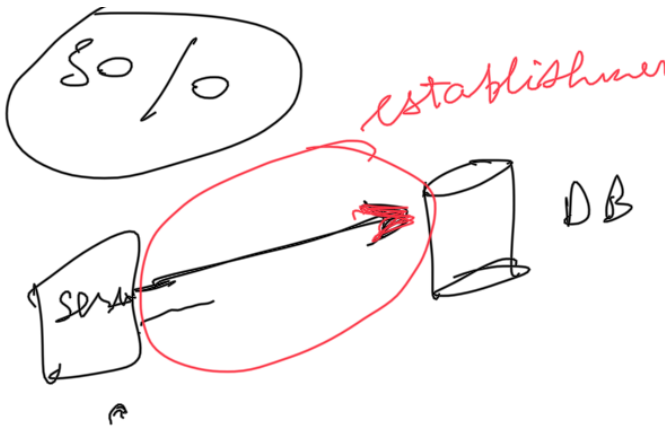
Exception Handling ↓ → handle runtime errors

→ Event that disrupts the flow of program

→ Object thrown at runtime

Exception →

50/0 → 8hr



run () throws Exc

Types of JAVA Exception

1. Checked [SQL Exceptions]
2. Unchecked (Error) [ArrayIndexOutOfBoundsException]

String s = null;
s.length() // NullPointerExcep

SQL

class A extends B

B

try {
try }

Nested Exception Handling