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
→ Verifies the code
→ Executes Code
→ Provides Runtime environment

JRE →

↓

Implementation of JVM

Dev tools
→ compiler
→ interpreter
etc

JDK

JDK JAVA Development Kit

JRE + dev

OOP →

Object Oriented Programming Language

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

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

```
class Car {
    int regNum ;
}
```

variables

Car car1 = new Car();

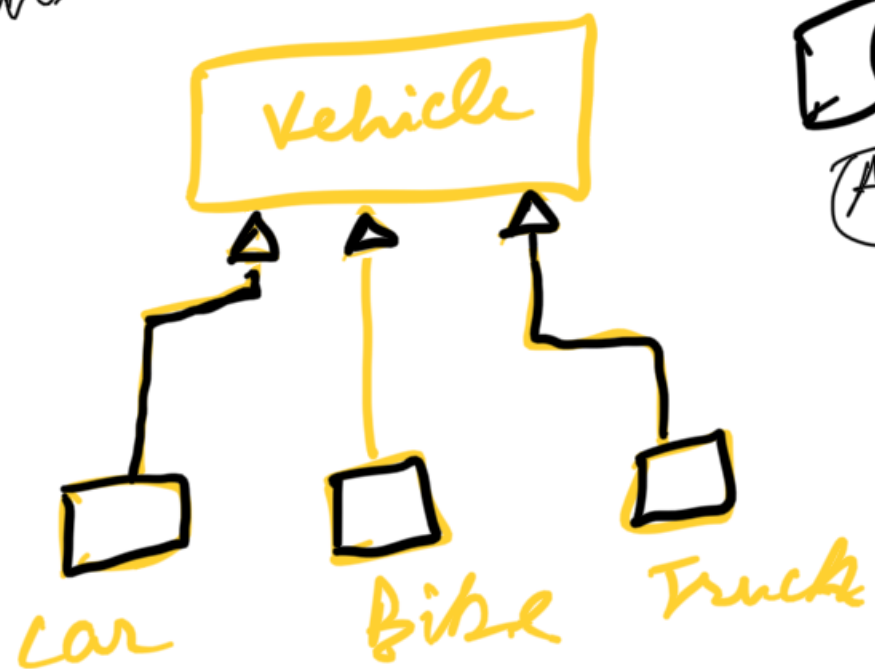
String color;

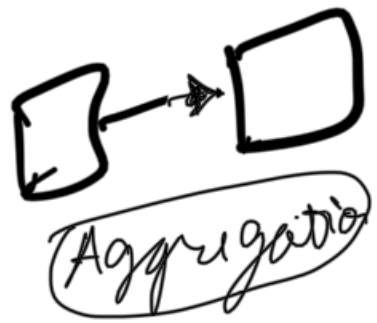
→ accelerate();

/ properties

} functions / functionality

③ Inheritance





1. Car →
2. Motorbike →
3. Truck →

④ Polymorphism →

→ add (int a, int b)

→ add (int a, int b)

} → ① Function / Method overloading

→ overriding

, int c)

(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

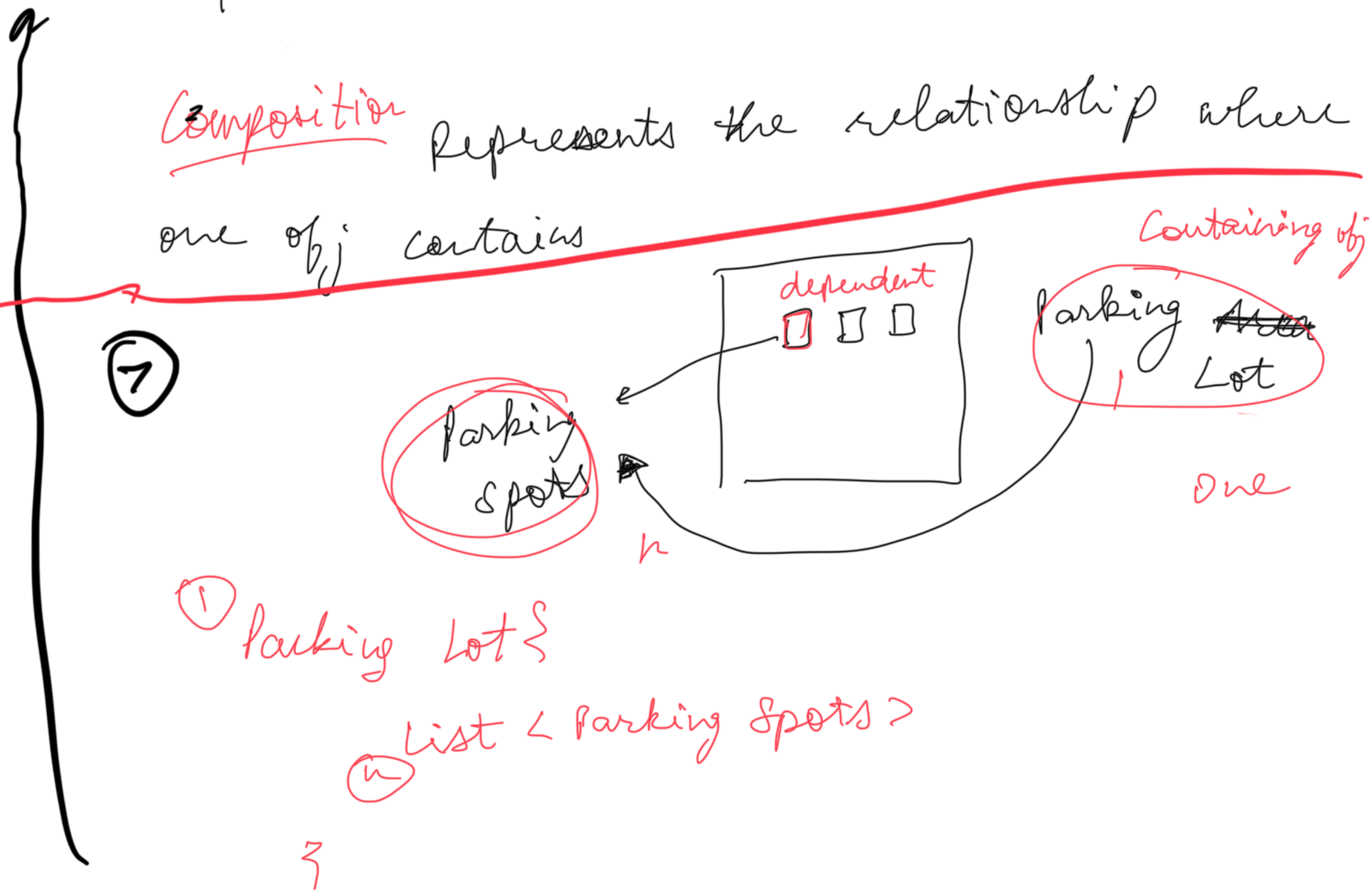
Obj 1

Obj 2

Many to one
one to many

way to achieve

Composition Represents the relationship where one obj contains



① Parking Lot {

② 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

Initialise JAVA

Constructor →

→ 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 →

1. By changing no. of args

Ref

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

vehicle

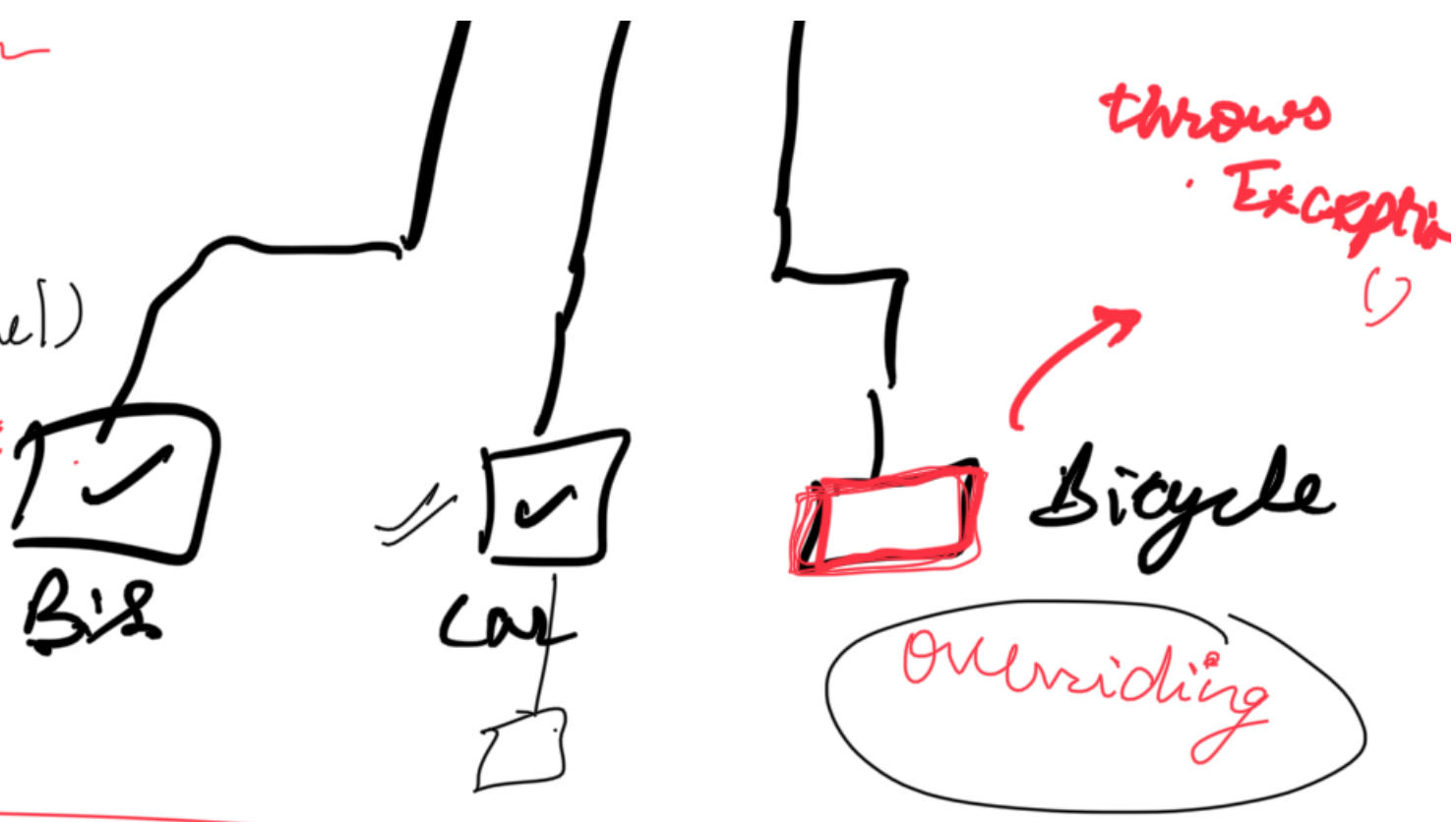
Superclass

used for runtime

polymorphism

throws Exception

Bicycle
6. switchEngine()



R. e ( ){

super();

super(—,—)

super()

}

Super -

1. Method

Refer to parent

super. fuetion

super. color

2. Variable

3. Constructor → super() → in the child constructor -or

Final Keyword →

1. Variable → can't change

2. Method → Override ✗

3. Final Class → You can't extend it

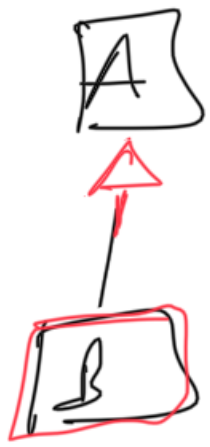
Inheritence → ✗

Runtime Polymorphism

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

Upcasting

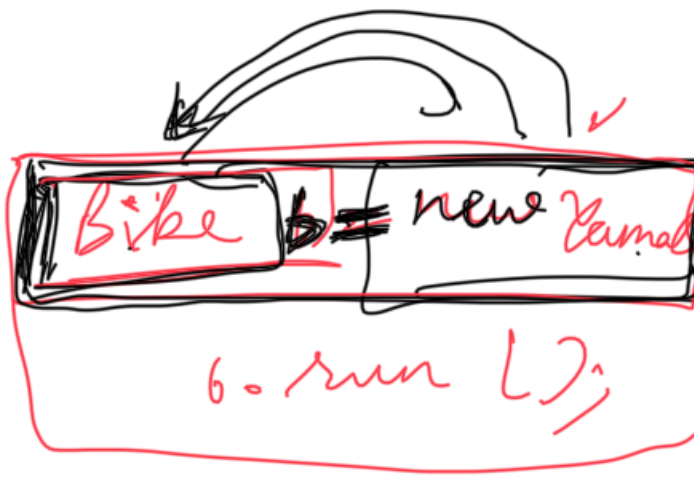
Reference variable of parent class refers to child class object

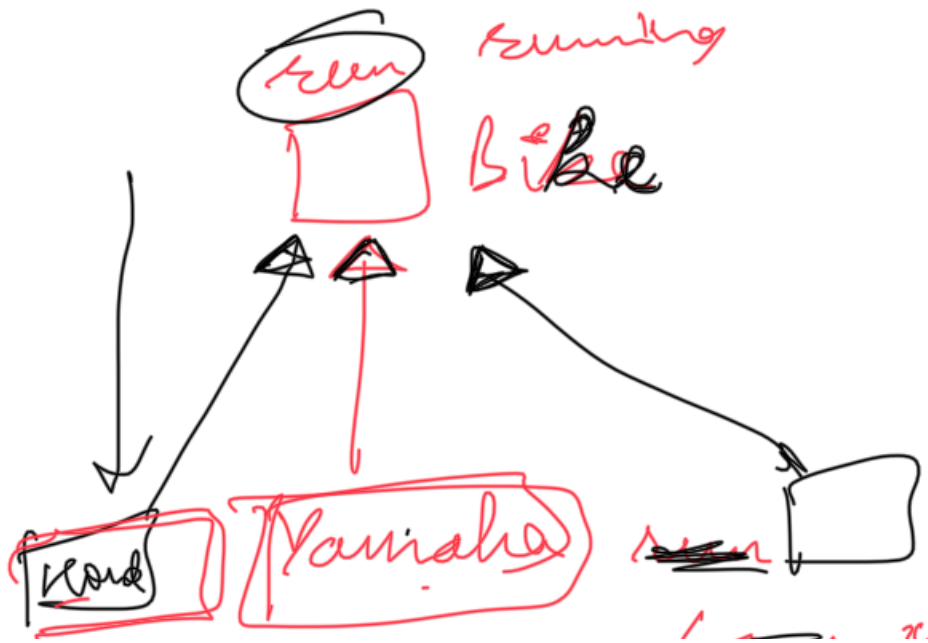


Class B extends A

A a = new B();

a.





1. Abstract class →

abstract class xyz {

// abstract methods

// non abstract me

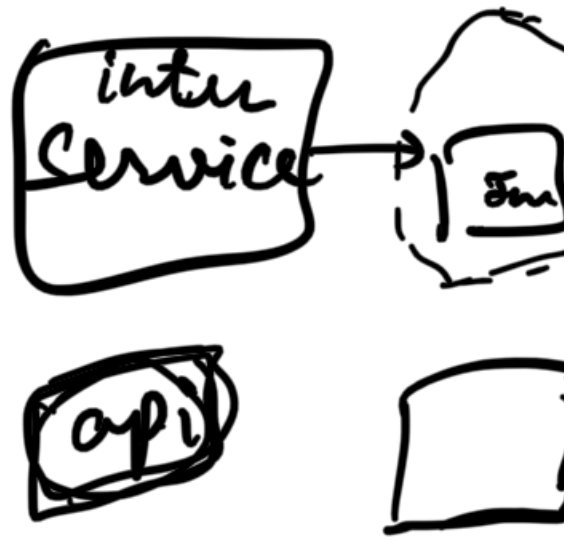
extend + method implementation }

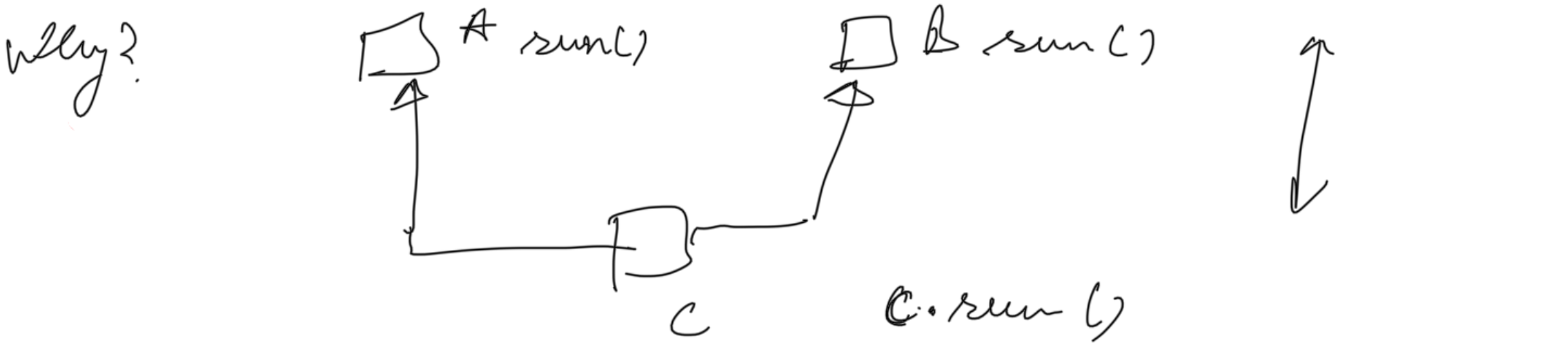
abstract class Vehicle {

class

abstract void run();

class Honda extend Bike {




why?
A
sum()
B
sum()
C
C.sum()

Interface
A
sum()
B
sum()
Abstract
methods
sum()}

Abstract Class	Interface
1. Abstract methods & Non abstract methods	1. Abstract methods
2. Multiple Inheritance ✗	2. ✓
3. abstract	3. interface
4. 'extend'	4. implement
5. Private, protected	5. Public by default

Encapsulation →

① Packages

Group of similar type of classes, i ——

Built in package

User defined

package ——

removing naming collision

① B

package pack

public class A {

}

import

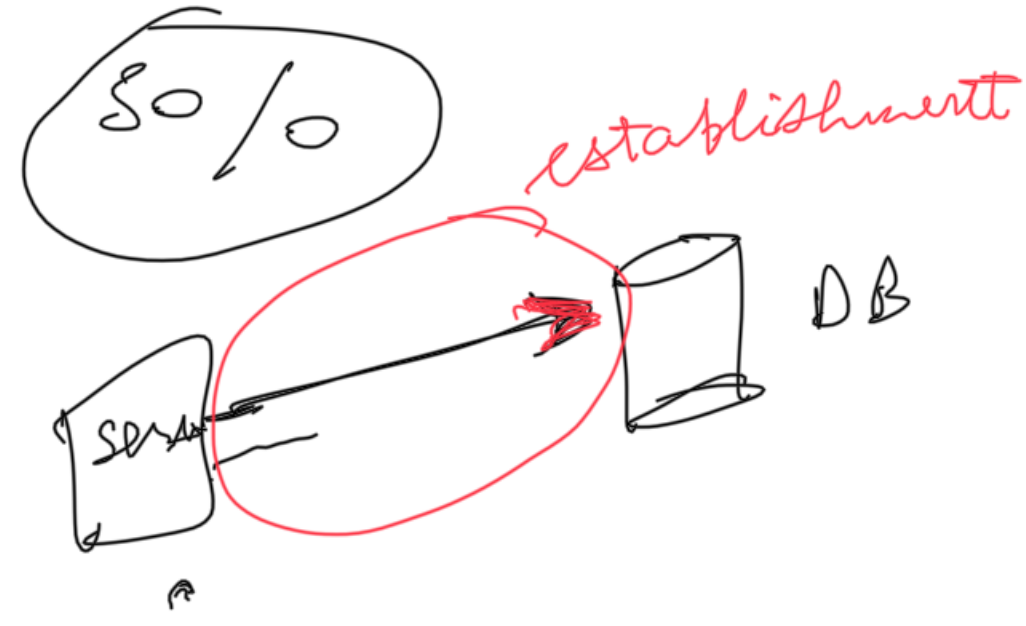
Exception Handling

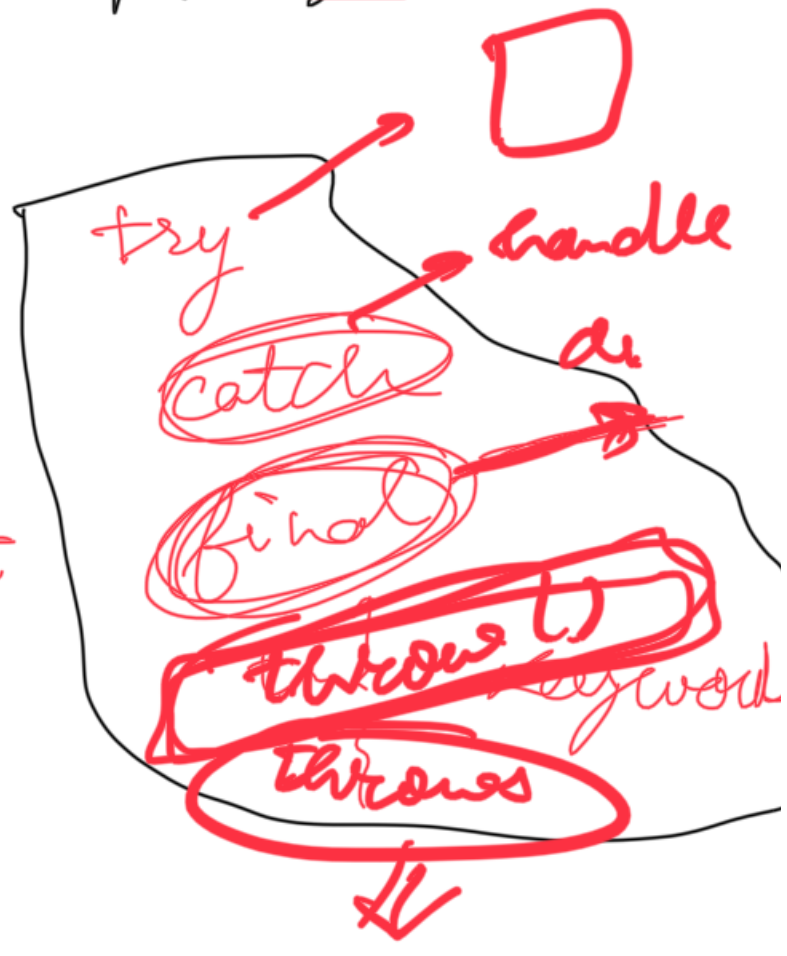
errors

→ Event that disrupts the flow of program

→ Object thrown at runtime

Exception →





Types of JAVA Exception

[SQL Exceptions]

1. Checked [SQL Exception]
2. Unchecked (Error) [Array Index Out Of Bounds Exception]

String s = null;
s.length() // Null pointer Excep

SQL

Class A extends B

B

try {
try }

Nested Exception Handling