OOPS (Object-Oriented programming Systems)

→ OOPS is the core of Java which is used for designing a program using classes and objects.

→ In this approach, coe can define the data type of a data structure and the operations that are applied

→ OOPs is used in Java, C++, C#, python, JS, Perl, Ruby etc..

OOPS IN JAVA :-

→ Objects
→ classes
→ Abstraction
→ Inheritance
→ Polymorphism
→ encapsulation.

Ex :-

```
class Car {
    int price;
    String colour;
    int topspeed;
}
```

→ Car is the class and price, colour, speed are objects.

Languages that use OOP :

→ C++
→ Java
→ PHP
→ Python
→ C#

Benefits

→ Modularity
→ Reusability
→ Productivity
→ Security
→ Flexibility.

Alternative methods to OOP

→ Imperative programming : focuses on functions rather than models

→ Declarative programing : It involves statements on what the task is but not how to achieve it.

CLASS :-

→ The collection / The group of certain Properties, function and behaviour is known as class.

→ It is also known as template of an object which doesn't consume any space.

→ It has fields, Methods & constructors.

→ A class contain
- → class name
- → Modifiers
- → super class
- → Body
- → Interface..

Ex :-

```
static class Nums {
    int a;
    int b;
    void PrintSum() {
        SOP
        return a+b;
    }
    Void PrintMul() {
        SOP
        return a*b;
    }
}
Main {
    nums num = new nums()
    num.a = 5
    num.b = 6;
    num.PrintMul(); }
```

» output
» a+b
» a*b

» 30

OBJECTS :-

→ Objects are always called as instances of a class. They have states & behaviour.

→ For Example :-

when we treat animals as objects.

it has states like color, name, breed etc.

and behaviours such as eating, sleeping etc.

Example

```
static class Mybook {
    int x = 10;
P.S.V.M (String args[]) {
    Mybook obj = new Mybook();
        S.O.P (obj.x);
    }
}
```

>> 10

→ Mybook is the datatype which is created by the user using a class.

→ obj is the reference variable.

→ new keyword allocated the memory in stack

→ Mybook() is the constructor / function in class

Example :-

```
class student {
    int age;
    String name;
    int roll;
}

class Main {
    P.S.V.M (String [] args) {
        Student s1 = new Student();
            s1.age = 18;
            s1.name = "John";
            s1.roll = 623;
        S.O.P (s1.age);
        S.O.P (s1.name);
        S.O.P (s1.roll);
    }
}
```

>> 18
>> John
>> 623

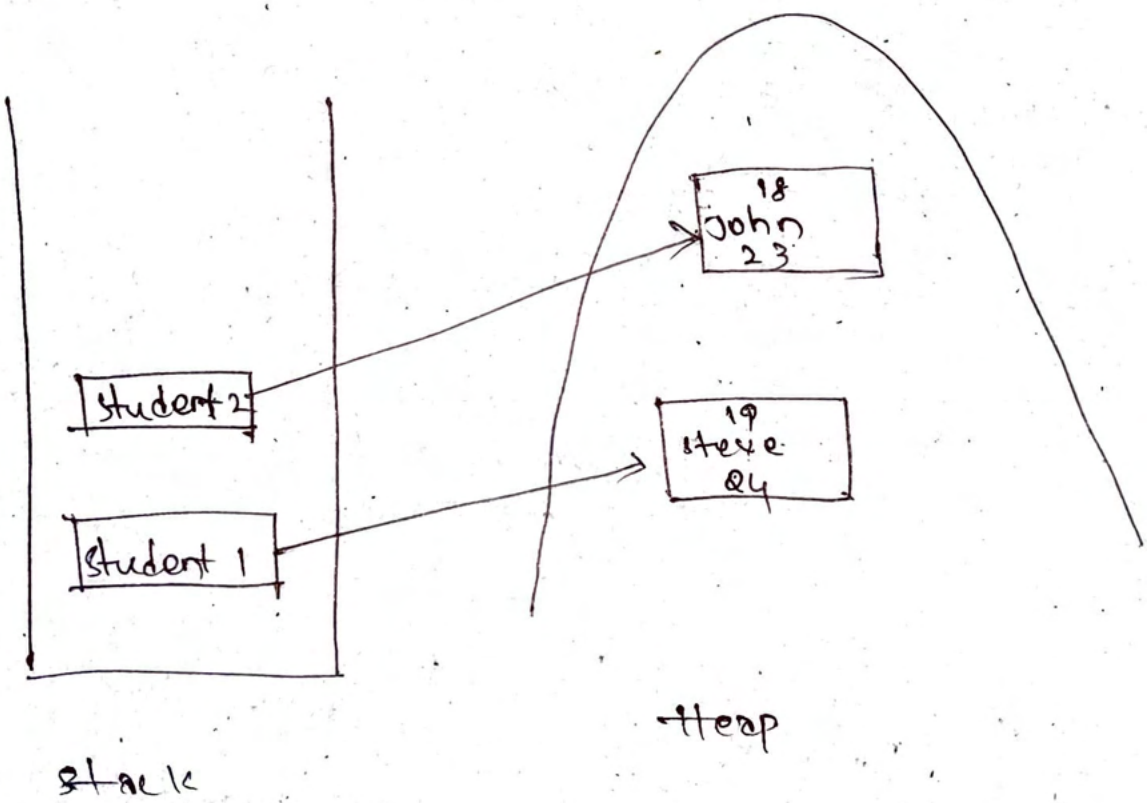
DYNAMIC MEMORY ALLOCATION.

```
Student s1 = new student();
```

compile time — Runtime.

→ The Memory allocation while the Runtime of the program is known as dynamic memory allocation.

Memory



CONSTRUCTORS :

→ A constructor is a block of code similar to the method

→ It is called when an instance of a class is created.

→ It is a special type of method used to initialize the object.

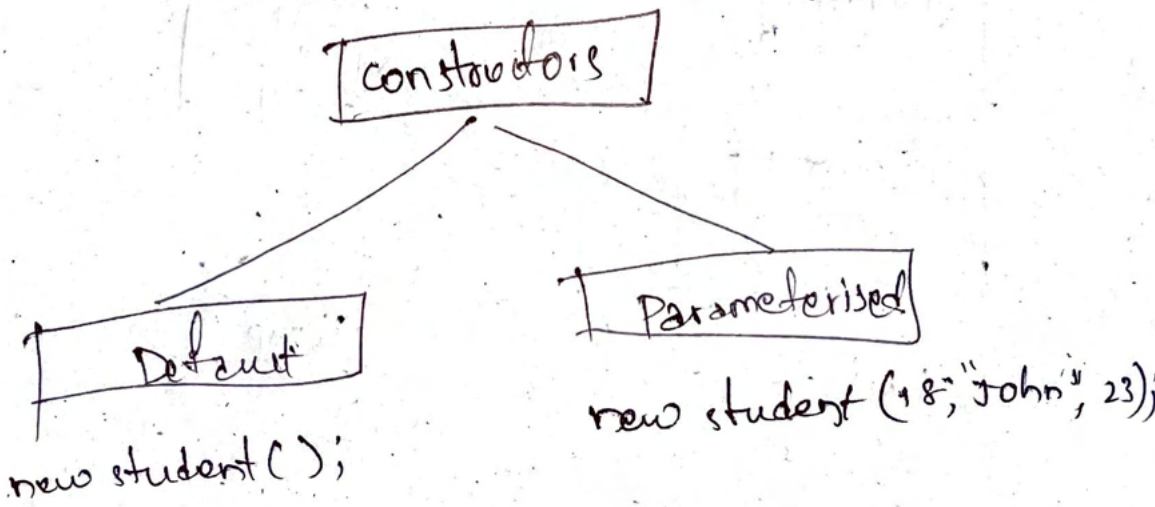
→ constructor binds the objects with args with object

Rules :

→ Name of class = Name of constructor

→ No return type.

→ it cannot be abstract, static, final

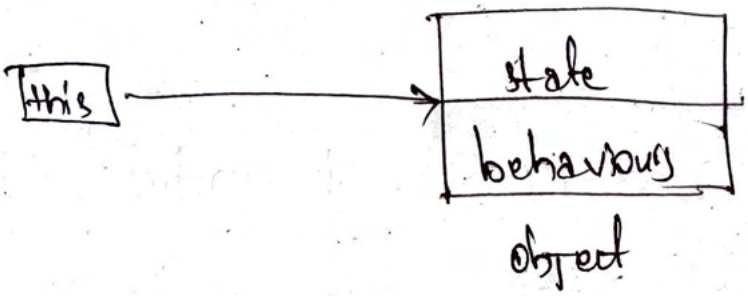


Example

```
static class Rectangle {
    int length;
    int breadth;
    int height;

  Rectangle() {   // Default constructor.
  }

  Rectangle (int length, int breadth, int height) {
      this.length = length;
      this.breadth = breadth;
      this.height = height;
  }
  void Printarea() {
      S.O.P(l * b * h);
  }
}.

Main {
  Rectangle r1 = new Rectangle (3, 4, 5);
      r1.Printarea();
```

This keyword :

→ This is a reference variable that refers to the current object.



→ Used to refer current class

→ used to pass as an arguement.

→ It is used to access an object inside the current class

CONSTRUCTOR OVERLOADING :

constructor overloading is a technique of having more than one constructor with different parameter list. Each constructor performs various task.

WRAPPER CLASS :

It is a type of class in which a primitive datatype is converted into an object to perform various operations

Ex:-

```
Main {
    int a=45; // Primitive datatype.
    Integer num = new Integer(45); // object
    S.O.P(a);
    S.O.P(num);
  }
}
```

>> 45
>> 45

FINAL KEYWORD

Final keyword is a non access modified. used for classes, attributes and methods, which make them non-changeable.

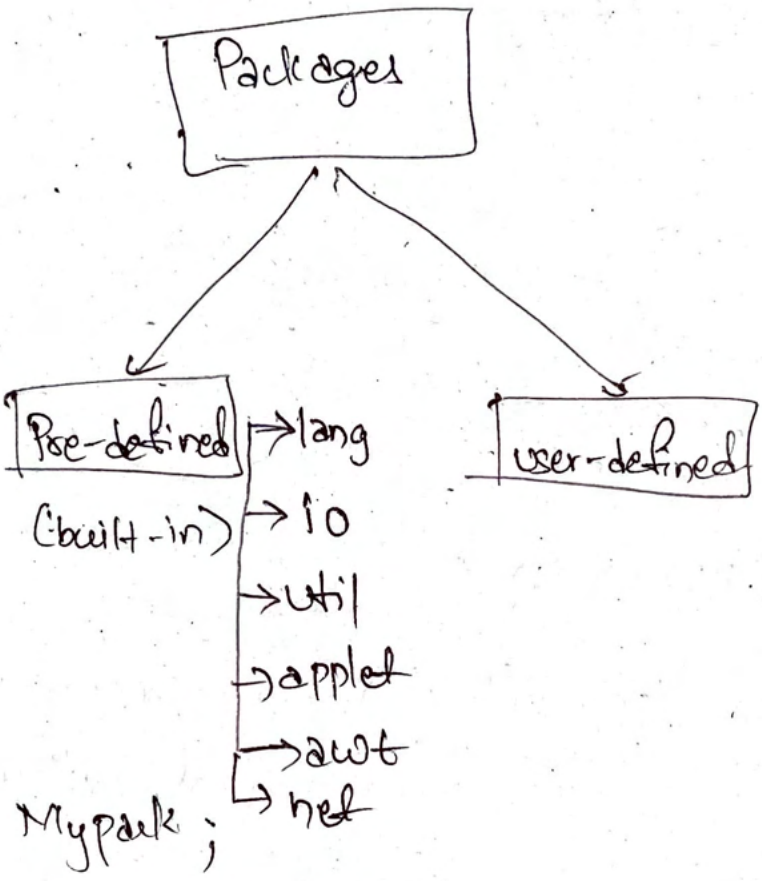
⇒ It is impossible to inherit or overide.

Ex :- PI = 3.14159...

⇒ The final keyword is called as modifier.

PACKAGES :

→ A package in java is said to be a group of similar types of classes, interfaces and sub packages.



By :

```
Package Mypack;
Public class Example{
    P.S.V.M(String []args){
        S.O.P("Hello");
    }
}.
```

Syntax for importing package

import package.*; [for all classes]

import package.classname; [for a specified class].

Ex:-

```
Package A;
class PackA {
    P.S.V.M{
    void msg(){
        S.O.P("Hello")
    }
```

```
Package B;
import pack
import A.*;
class PackB {
    P.S.V.M(String
        [] args
    A obj = new A();
    obj.msg()
    }
}
```

>> Hello.

IMPORT :-

The import keyword is used to import the package, class or an interface in the current class.

STATIC :-

→ We use static for Variable, Method, Block, Nested which is independent of class and commonly used in all class.

Ex

```
class Student{
    String name;
    int roll;
static String college = "SMEC";
    Student (String name, int roll){
        this.name = name;
        this.roll = roll;
    }
    void display(){
        S.O.P (name + roll + college);
    }
```

PUBLIC STATIC VOID MAIN()

Main is declared as static because we can use main method without creating any objects in the class.

→ We cannot use any non-static method inside the static method

→ static method can only access static data.

→ A non static is dependent on objects
→ A static is dependent on instance.
→ But a static Method can be used inside non static.

SYSTEM . OUT . PRINT ("");
↓ ↓ ↓
class. variable Method

SCANNER SC = NEW SCANNER (SYSTEM.IN
↓ ↓ ↓ ↓
class variable create Takes the
Memory. input.

SINGLETON CLASS :-

A singleton class is a type of class in which we can create only one instance/object.

→ What happens if the main keyword is replaced with another word

Answer:

The Java virtual Machine only looks for main and not any custom names as the starting point.

String args[]

This feature allows the user to pass an arbitary number of value of the declared datatype to the method as parameters.

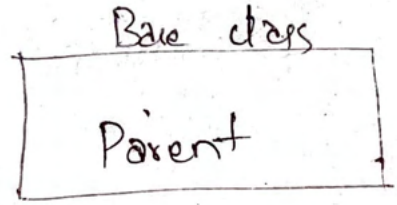
→ If is a command line arguement located in ~~the~~ (any) package.

INHERITANCE :-

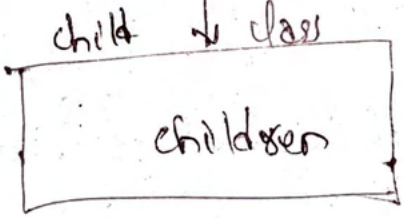
Inheritance in Java is a mechanism in which an object acquires all the properties and behaviour of parent object.

→ We use Inheritance for Method overriding

→ We use extends keyword Inorder to Access a parent class.



(Money, Property traits)



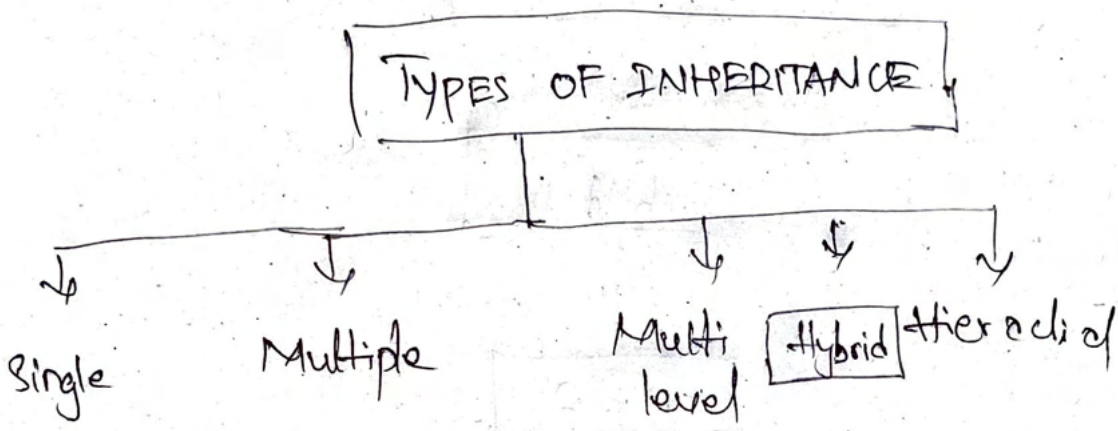
=> children can use the properties of parent class ~~including~~ and also its own properties of children.

Ex 1:

```
static class box1 {
    int length;
    int breadth;
    int height;
    box1 (int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }        // constructor.
}

static class box2 extend box1 {
    int weight;
    box2 (int length, int breadth, int height,
          int weight) {
        super (length, breadth, height);
        this.weight = weight;
    }
}
```

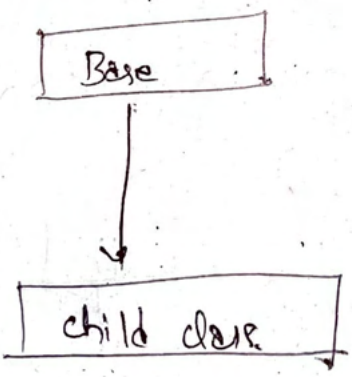
SUPER KEYWORD :

When ever the subclass / child class needs to access the constructors in parent class or superclass from which it is defined we use super keyword.



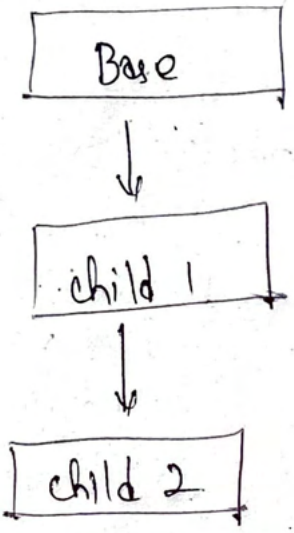
(1) Single Inheritance :

One class extends another class



(2) Multi level :-

The parent class derives a child class and the child class derives another class.

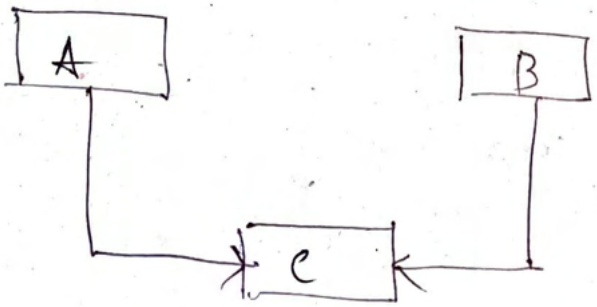


⇒ for child 2 the parent class is child 1

→ for child 1 the parent class is base class.

(3) MULTIPLE INHERITANCE :-

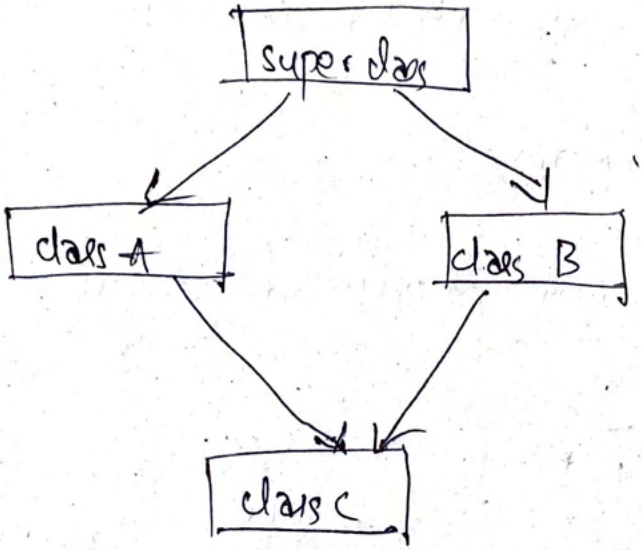
One class extends more than one class.



→ Multiple Inheritance isn't supported in Java because if two or more classes have the same property, then the child class gets confused.

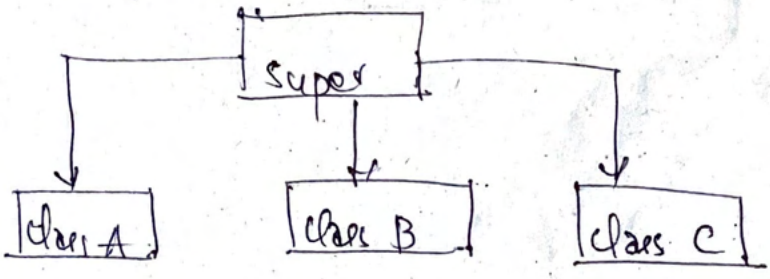
→ It leads to diamond problem

diamond problem in Java :-



(4) Hierarchical Inheritance :-

One class is inherited by many classes



Ambiguity :-

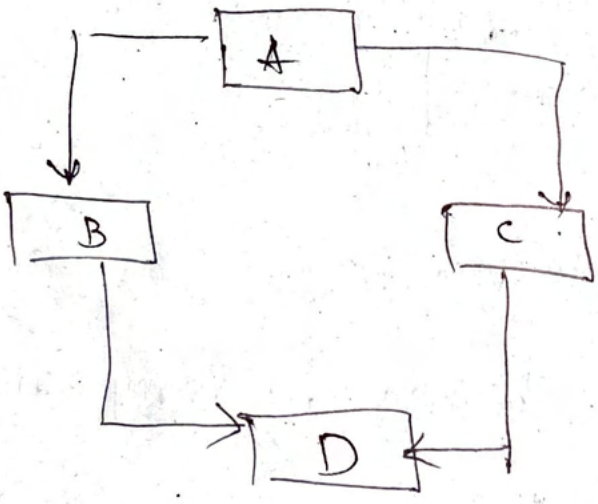
Issues / bugs that are not specified perfectly in Java.

→ It occurs in Multiple Inheritance.

(5) Hybrid Inheritance :-

→ The combination of Single and Multiple inheritance.

→ It is not supported in Java



POLYMORPHISM :-

Poly Morphism

Many ways to represent

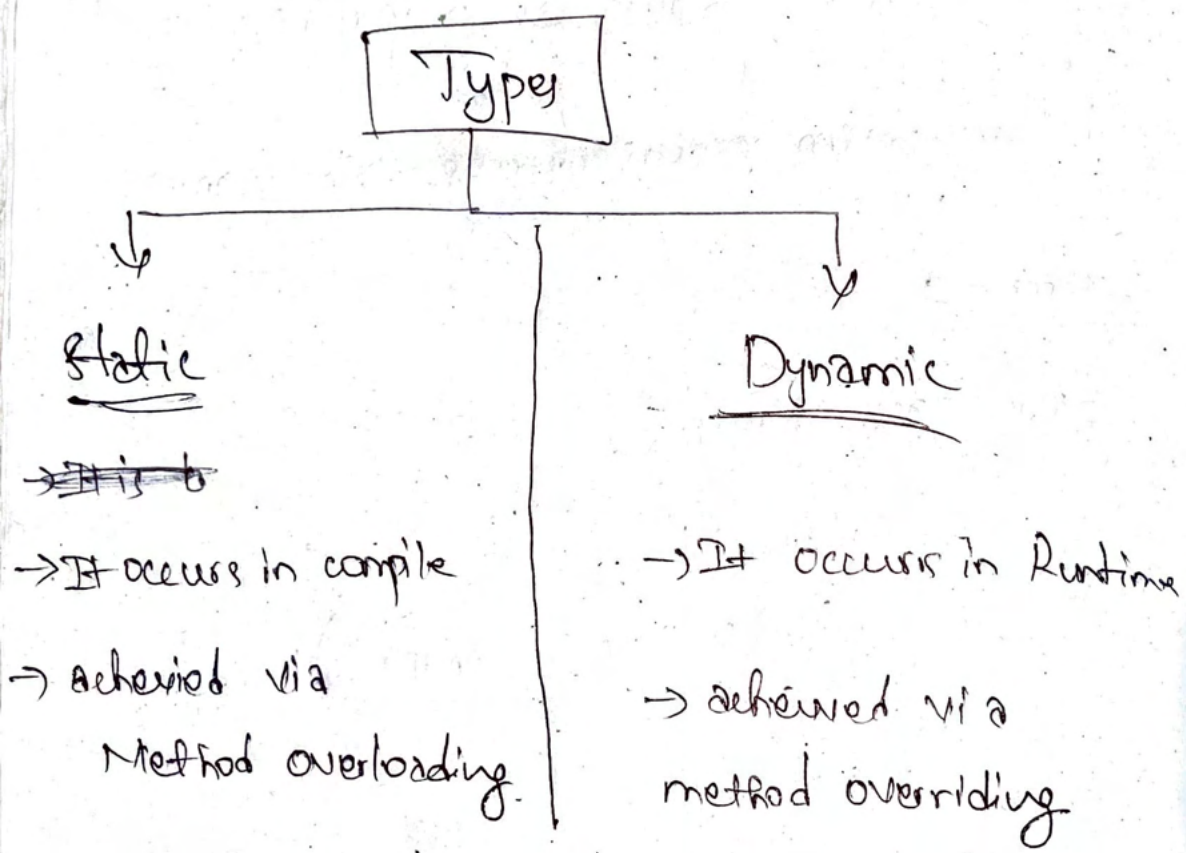
→ Many ways to represent single entity or item is known as polymorphism.

→ Polymorphism occurs during Inheritance.

Example :-

```
Static class shapes {
    Void Area(){
        S.O.P("am in shape");
    }
}
Static class Circle extends shapes {
    Void Area(){
        S.O.P("Pi * r * r");
    }
}
```

```
Static class Square extends Shapes {
    Void Area() {
        S.O.P ("side * side");
    }
}
}
```



Method overloading :- Methods that have same name but different parameters and different return types

Method Overiding :-

When the Method name in the child class is same as the method in the superclass it is known as Method overiding.

→ we use @override annotation in order to check whether a method is overridden or not.

FINAL KEYWORD :-

→ we use final keyword in ~~total~~ which the istance, method or variable is unchangleble.

→ we use final inorder to prevent method overriding.

ENCAPSULATION :-

Wrapping up the implementation of data methods and data members inside a class is known as encapsulation.

Abstraction :

Hiding the unnecessary detail and showing the necessary details is known as abstraction.

Encapsulation Vs Abstraction

Abstract datatypes :

ADT is an abstraction of a datastructure in which the interface doesn't give any specific details.

Ex: ArrayList, StringBuilder etc..

Getters and setters

→ These are used in encapsulation.

→ Inorder to get the return a value we use get() method AKA Getter.

→ Inorder to change the value we use set() method AKA setter.

(*)

→ Getters & setters are used in ENCAPSULATION

→ ENCAPSULATION is used in ABSTRACTION.

PUBLIC :-

The public keyword is an access modifier used for classes, Methods and attributes, Making them accessible by any other class.

PRIVATE

The Private keyword is an access modifier used for classes, methods and attributes making them only accessible within declared class

→ We can access the private attributes using getters and setters.

PROTECTED

There are used to restrict the scope within which the variable, method, constructors can be accessed

Ex for Private :

```
static class Name {

    Private string name = "vignesh";

    Public string Getname() {  //Getter
        return name;
    }

    Public void Setname(string naam){
        this.name = naam;  //setter
    }
```

(*)

→ Public → Everywhere in world

→ Private → Only in specific file but can be accessed through getters and setter

→ No access modifier → can be used only in specific package

	class	Package	Subclass same Pkg	Subclass diff Pkg	world
Public	✓	✓	✓	✓	✓
Protected	✓	✓	✓	✓	✗
No modifier	✓	✓	✓	✗	✗
Private.	✓	✗	✗	✗	✗

✓ → can be accessed

✗ → cannot be accessed.

hash code :-

A number representation of an object is known as hashcode.

ABSTRACT CLASSES :-

An abstract class is a class that is declared as abstract. It may or maynot have abstract methods. They cannot be instantiated. but they can be subclassed

Instantiation :- creating new instances of objects to be used in program.

Syntax :-

abstract returntype name (Parameters);

(*) If a class contains morethan one abstract methods, it should be declared abstract

Generics

The Java generics programming is introduced in J2SE5 to deal with type-safety objects. It makes the code stable by detecting bugs at compile time.

Advantages

→ Type Safety
→ Type casting not required
→ compile time checking

If is used in ArrayList

ArrayList<Integer> list = new ArrayList<>();

↓

Generic

LAMDA FUNCTIONS

These are in line functions which are written in a single line of code.

Syntax

(Par1, Par2) → {code block}.

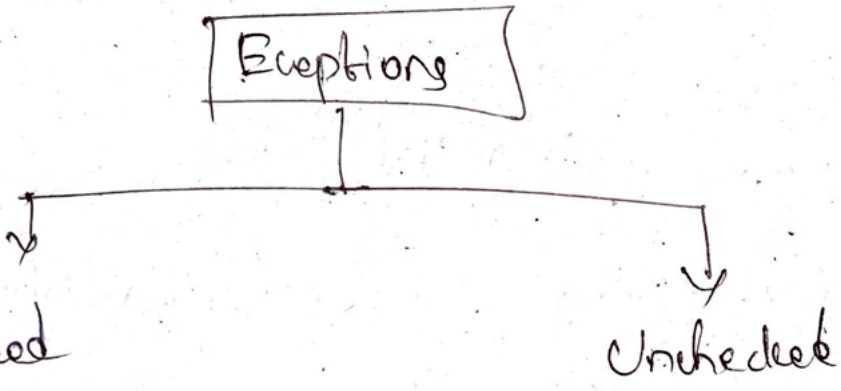
Exception handling:

It is a type of mechanism used to handle the runtime errors so the flow of program is maintaing

Exception:

→ Exception is an abnormal condition

→ Exception handling is used to handle runtime errors



→ Compile time → Runtime.

Keywords:

→ try
→ catch
→ finally
→ throws → declare exception
→ throw → throw exception.

Collections

→ The collection in Java is a framework that provides an architecture to store and manipulate the objects.

→ Ex:- Set, List, Queue, Deque etc...

Framework :-

→ Framework provides readymade architecture.

→ It represents a set of classes & interfaces

→ It is optional.

collection framework:-

It represents a unified architecture to storing and manipulating a ~~set~~ group ~~of~~ objects.