Object Oriented Programming.

Class: [Entity represented as Some attribute & behaviour.]

Behaviour are actions. ↓ entity is written inside Java class Program.

Attributes, methods, Properties } → class.

attributes & Behaviours → Properties

```
Public class student {
    String address;
    String name;
    int age;
```

class:
is named group of Properties & functions.

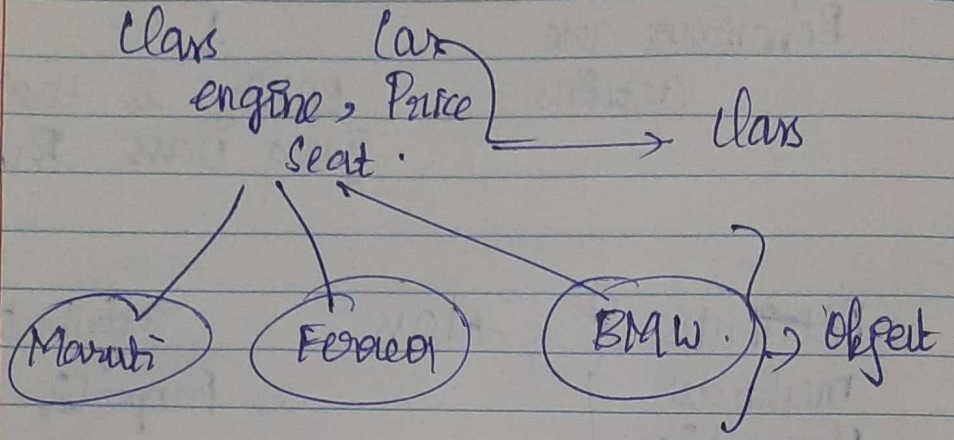
Create class:

```
class Student {
    int[] rno = new int[5];
    String[] name = new String[5];
    float[] marks = new float[5];
}
```

class Student { int rno.; String name; float mark; }

Student Ragavarshini; ↗ this if you call contains all 3 attributes.

class is a template of object.
object is a instance of class.



Print (Maruti.Price). → Ans (20000)

dot operator.

to create object

~~Student s = new S.~~

Car car1 = new Car();

dynamically allocate memory & return a reference variable to it.

Car Car1 = new Car();

Compile time → Bytecode & JVM
All during compilation

Run time. → Application Running that tym memory will be allocated → dynamic memory Allocation

By default null or zero.

```
Car car1 = new Car();
SOP(car1.price)   0.0
SOP(car1.Engine)  null   } -> default
SOP(car1.Seat)    0
```

Bytecode & JVM

↓

Bytecode is an intermediate representation of Java Program that run on any System with JVM.

JVM is responsible for interpreting & Executing the ~~Java code~~ bytecode instructions providing Platform Independent runtime for Java application

"write once, run anywhere"

new → Using for creating instance of class objects and it will dynamically allocate memory and return reference variable.

Constructor: If a class has 3 attributes like regno, name, salary for each object you cannot assign it so use

↙ Special function in class. Bind the arguments with object

```
Student Kunal = new Student(13, "Kush", 14.3);
```

Constructor will have same name as the class.

```
Student() {
    this.regno =
    this.name =
    this.salary =
}
```

this will replace the object.

helps to initialize variable.

```
void changename ( string Newname){
        name = Newname;
  };

  Kernal. changeName( "  ")
  }
```

Constructor overloading:

Has multiple constructors with different parameter list.

```
Public class rectangle {
      int width;
      int height; int width, int height
  Public rectangle () {
      this. width = wi
      this. height = height;
  }
  Public rectangle (int sidelength) {
        Width = Sidel;
        hie = Side;
  }
    Public int cal () {
        ret w*h;
    }
}
```

Wrapper class.

total keyword.

encapsulates
Primitive data types
and Provide utility
method for working.

Integer wraps int
Double " double
Character " char
Boolean " boolean.
Byte " byte.

```
Integer num1 = new Integer(10);
int value = num1.intValue();
```

Autoboxing - converting int to Integer
Unboxing - " Integers to int.

Garbage Collection.

Automatic garbage
collection will happen in Java.

```
Public class Garbage {
  Psvm ( ){
Person Person1 = new Per("Alice");
"  2 = "  (bob;
"  3 = "  (Charlie;
Per1 = P2;
P2 = null;
```

bob becomes garbage.

```
for(int i=0; i<10000; i++)
    new Per(" "+i)
```

after loop its Garbage collected

Final variables:
↓ cannot be change after it has been initialised.

Method: method cannot be overridden by Subclass

Class cannot Be Subclassed.

Video 2: follows hierarchial method which means folder inside folder.

Packages:

Package is generally a folder.
In one you cannot create classes with same name that is where Package comes into case.

Path. ← Package com. Raj. Packages. b;

Public class Greeting { → Here is where the file lies.

}

you can access a code written in Package b from Package A.

Package Greeting → import static com. Kunal. Pkags. b. Msege. ms;

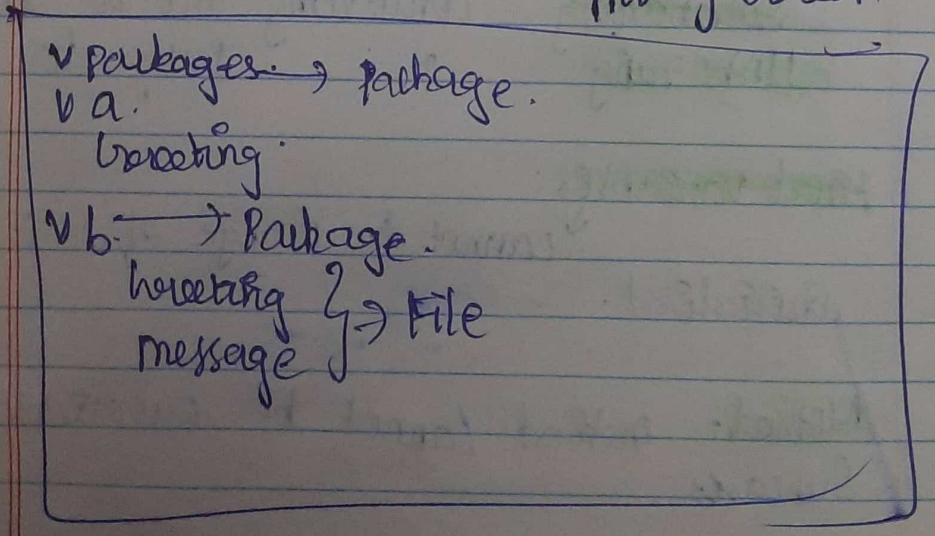
Package

Message Package → message written here.

class

Import Statement.

Intellij Screen.



import com. Raj. Packages. b. Message. message.

Static Element Example.

Human

```
Public class Human {
    int age;
    String name;
    int Salary;

Public Human (int age, String name, int Salary) {
    this.age = age;
    "    = ;
    "    = Salary;
}
}
```

Main

```
Public class main {
    PSVM {
        Human Rag = new Human (22, "Vansh", 10000);
    }
}
    SOP (Rag.age);
}
```

Package

What is Static Variables:

- Static variables are common for all the objects that cannot change for each objects.

(Eg: Population)

Static long Population;

April

S.O.Pln(Kunal.Population);
SOPln(Human.Population);

for Static variable don't use object name instead use class name.

Inside a static method we cannot use anything Unstatic which means.

```
PSVM()                          PSVM()
{                                {
}.greeting,                      greeting;
                                 }
Public Void greeting;            Void greeting;
```

```
Public class StaticBlock{.
Static int a=4;
Static int b;

Static{
    SOP(" ");
  b = a*5;
}.

Public Static Void main(){.
   StaticBlock Obj = new StaticBlock();
```

Inner class:
classes inside classes.

```
Public class Innerclass {
  Static class Test{
    String name;
Public Test ( String name){
    this.name = name;}
  }
}
  psvm ()
    Test a = new test ( );
    Test b =  "       ( );
```

If you dint mention Static for class inside Innerclass it will give error.

Singleton class:

class where you can create single object.

Public class Singleton {

Because it can be accessed without creating any instance

```
  -Private Static Singleton instance ;
  Private Singleton(){
  }
  Public Static Singleton getInstance(){
    if(instance ==null);{
        Inst = new Singleton(); }

  return instance;

}}
```

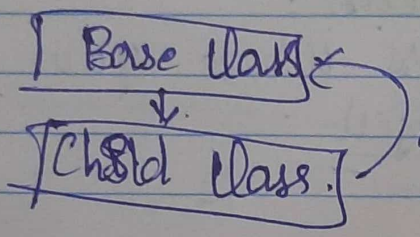
Video-3

4 Important Properties.

Inheritance
Polymorphisms.
Encapsulation.
Abstraction.

Inheritance:

Class that use Properties and Stuffs of other class.



Child class will inherit the Properties of Base class.

```
class child Extends Base {
        int weight;
    }
class child = new child ();
    child. length
    child. width;
    child. height;
```

Child class will inherit Properties from Base class also has some additional Properties

```
Public class Box {
    Double l;
      "   h;
      "   w;
Box() {
    this.h = -1;
    this.l = -1;
    this.w = -1;
  }
Box(double side) {
    this.w = side;
    this.l = side;
    this.h = side;
  }
Box(double l, double h, double w) {
        this.l = l.;
        this.w = w;
        this.h = h;
}.

Public class BoxWeight extends Box {
    double weight;

Public BoxWeight() {
        this.weight = -1;
  }
}

Public class Main {
    PSVM() {
      Box box1 = new Box(4.2,0.7,4.2);

      BoxWeight box3 = new boxweight();
```

```
Public BoxWeight
(l, w, h, weight).
    this.w = we
    Super(l, h, w)
```

Private "keyword":

If you created any variable for Private keyword then it will be accessed only within that file.

Super keyword:

Every class has object as a Superclass.

Super is a reference variable that is used to refer to the immediate Parent class object.

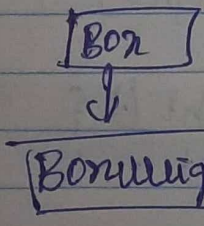
doesn't care about what child class contains.

Super(l, w, h) → use in child class that refers to variables/ objects in Parent class.

Types of Inheritance:

① Single Inheritance:

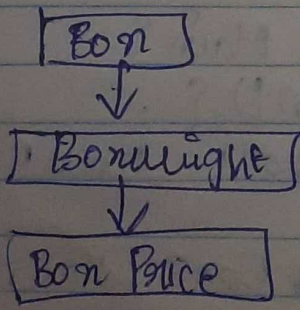
one class Extends another class.



② Multilevel Inheritance:

~~one class extends more than 1 class.~~

A child class can Be a Parent of Some other class.

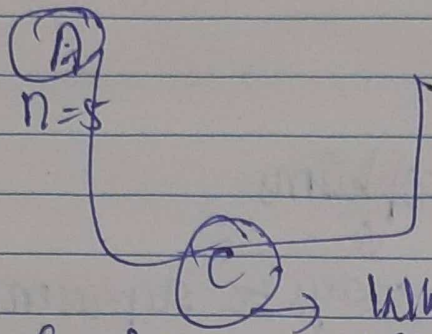


l, h, width weight.

Multilevel

Box extends BoxWeight.

BoxPrice extends BoxWeight {
    double cost;
}



→ What

C obj = new C();

c.n → this doesn't know what to print for n either 5 or 10 So.

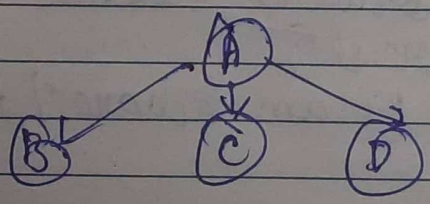
multiple inheritance is not supported in Java.

③ multiple inheritance:

One class Extends more than 1 class. Not allowed in Java.

④ Hierarchial inheritance:

One class is inherited by many classes.

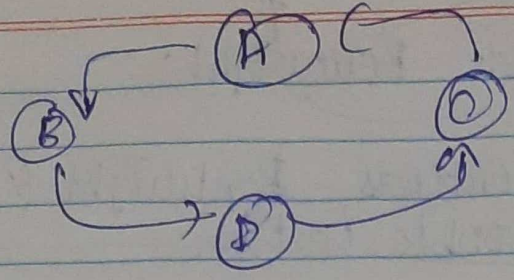


BoxPrice can inherit Box.

⑤ Hybrid inheritance.

Combination of Single & multiple inheritance.

(not Supported in Java)



Polymorphisms :

poly morphisms
↓ ↓
many ways to represent.

```
Public class Shapes {
    void area() {
        SOP("This is Shapes");
    }
}
Public class Circle Extends Shapes{
    void area() {
        SOP("Area of Circle is");
    }
}

Public class Square Extends Shapes{
    Void area() {
        SOP("I am Square");
    }
}

Public class main{
    PSVM ( ) {
        Circle. Circle = new Circle();
        Square. Square = new Square();
    } Shape. area();
}
```

Types of Polymorphism:

① Compile time / Static Polymorphism → achieved via method overloading.
② .

Same name but types, arguments return types ordering can be different.

Ex multiple constructors:

```
Public class Numbers{
    int Sum(int a, int b) {
        return a+b;
    }
    int Sum(int a, int b, int c){
        return a+b+c;
    }
}
```

→ Same name but diff Params arguments & return types.

② Runtime / Dynamic Polymorphism:→ method overriding.

The Shape has 2 classes one is Circle another Square in both area is there. this is called overriding.

@ override → will help check if it is overridden or not.
→ annotations → gives data abt the program But not part of Program itself.

Upcasting: Process of casting an object from Subclass to Superclass.

Encapsulation:

Wrapping up the implementation of data members & methods in class.

Abstraction:

Hiding unnecessary details & showing valuable details.

Access modifiers:- Private, Public & Protected modifiers:
default.

Private ⇒ accessable only within the class

Default (Package-Private).

↓

No specifier → void default() → don't have any specific modifiers

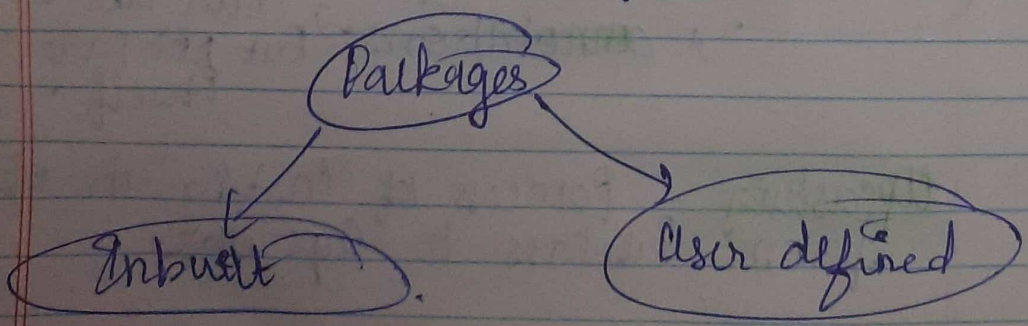
accessable only within same package

Protected:

Accessed by Subclasses, even if the Subclass are in different package.

Public:

Accessed from anywhere.



Inbuild:
lang → language
io → input output
util → utility class
data structure
Applet → Framework
awt → Graphical interface
net

hashcode is used in hashset, hashmap, hashtable.

Operator overloading:
Allows us to use familiar mathematical notation in code making it more Expressive & Intuitive.
Java doesn't support operator overloading in default