

OOPS

⇒ OOPS

Class → is a named group of properties and functions

→ It is template of an object
ex: cars, houses.

Class → logical construct

Object → physical Reality // occupies space in memory.

Object is an instance of a class

Class Student {

int roll; } instance variables
String name;
float marks;

Student Sharath = new Student();

↙ by default constructor
if dynamically allocates memory at runtime

→ constructor is a special function, that runs when you create an object and assign some variables

→ this key word used to substitute by reference variable.

Student Sharath = new Student(13, "Sharath")

Student (int rono, String naam) {

 this.name = naam;
 this.rollno = rono;

It is substituted by Sharath.

→ constructor overloading.

⇒ final keyword

→ for primitives like int if we use final keyword we cannot modify the value of variable

ex: final int a = 10;

This 'a' cannot be modified.
But final variable should always initialized.

→ for objects we can modify the data
but we cannot reassign it

final Student A = new Student("something")

A.name = "nothing" } This works with
final keyword
but not reassigns
other object

→ in java for primitives it's only pass by
value. not pass by reference.
but for objects it pass by reference.

⇒ class
objects
new
constructor
final
Garbage collector

⇒ oops 2

→ packages are nothing but folders

→ When a property is common to everyone (class) we static.

```
class Human {  
    int age;  
    int salary;  
    static long population  
}
```

Here population is common property of the class.

→ When you are access static variable, always access with class name, you can also access through the reference variable of object but better way is use class name.

ex: "human.population"
is right way than
"this.population"

→ Why Main is Static?

1) Main is static because it is first method to run & when java program runs.

2) if it is not static it belongs to objects/instance.

3) Since Main is static, JVM starts the Java application, it needs a way to execute the code without requiring an instance of class. Declaring 'main' as static ~~then~~ that allow JVM to call directly using the class name without creating object of class.

→ Inner Class and Singleton class

⇒ static vs non static → non
 Static block → ~~non~~
 Singleton class
 inner class

Inner class

⇒ OOPS

Inheritance.

Class Box {

double l
double w
double h.

}

child class

can inherit from
parent class

Class BoxWeight extends Box {

double weight;

}

Super key word calls the constructor
of parent class

→ BoxWeight Box = new Box(...);

this is not possible because,
parent class doesn't know what
are below classes

→ Super keyword explicitly refers to super constructors or variables.

```
Box {
    double weight;
    double l;
}
```

```
Boxweight extends Box {
    double l;
    double w;
}
```

If you want to print or get 'l' from Boxweight class use this.l if you want to explicitly refer to Box class it use Super.l.

→ 'final keyword' is used to avoid inheritance and overloading.

→ Static methods doesn't override.

polymorphism → setter and getter

→ Data hiding focuses on data security
Encapsulation focuses on hiding complexity

```

=> Box {
    }

```

```

Boxweight extends Box {
    }

```

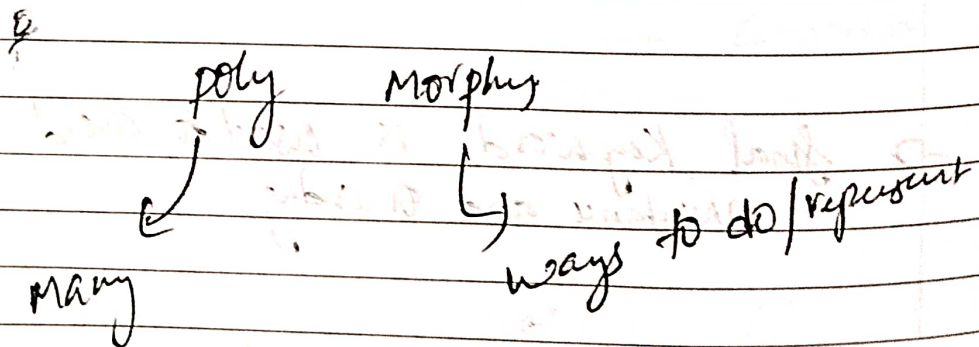
```

-> Boxweight bw = new Box();
-> Box bw = Boxweight();

```

And the difference between them

1st one is not possible mostly but it works after casting



Static/runtime polymorphism → Dynamic method dispatch (method overloading)

Dynamic/compiletime polymorphism → method overloading

→ OOPS4

→ Access modifier table

→ Imp rule for protected.

↳ It can be accessed in subclass
in other packages.

→ OOPS5

Abstract class

↳ methods should override.

Interface

↳ supports multiple inheritance.

→ OOPS6

Shallow copy → for primitive it creates
new one but non primitives it points
to same objects

Object comparison

Lambda function

Exception handling

Object cloning