OOP's

(Object Oriented Programming)

- OOP tries to map code instructions with real world making the code short and easier to understand.
- The core concept of OOP is to break complex problems into smaller objects.
- An object is any entity that has a state and behavior. For e.g., a bicycle is an object. It has
  - States : idle, first gear, etc.
  - Behaviors : braking, accelerating, etc.

- DRY stands for don't repeat yourself. this concept focusses on code reusability.

★ Class : A class is a blueprint for creating obj.

Car ⇒ Ford made a car ⇒ Ford Aspire

Class ⇒ Object Instantiation ⇒ object

↳ Contains valid info to create an valid object.

Like : we all are objects of human class.

! Always use capital letter to start class name. This is an Pascal's notation.

★★ Creation of class

```
class Human {
    int age;          // attribute 1
    String name;      // attribute 2
}
```

▶ Object : An object is an instance of a class.

! Memory will be allocated only after object instantiation.

Ex. We all are the object / different objects of human class.
We all are same but different.
As all have same prop. like have hands, eyes etc. but all have differet names, nature, different identity etc.

* Creation of object.

```
Class A {
    int age;

    A object-name = new A(); // object will be created
}
```

The new keyword is used to create an object.

▶ Constructor :- It has same name as class also but it dosen't have any return type.
It is an type of method having class name and non return type.

Types :> Parameterised
e.g. A(int a){ sout(a); }

2> Non-parameterised / default constructor
A(){ ..... }

* Calling constructor

```
A a0 = new A();   } These both will call
new A();          } constructor.
```

! Constructor is called after creation of obj.

* Instance block :-

```
class A {

    {
        sout("This is an Instance block");
    }

    psvm(String args[]){
        A a0 = new A(); // This will call
                        instance block and constructor.
    }
}
```

! The object must be created to call instance block.

! Because object is an instance of the class.

! As object get created first instance block will be called then constructor.

! If there are more than one instance blocks then they are called by sequence.

* Static block.

```
class A {
    static {
        sout("This is a static block");
    }
```

```
psvm(String args[]){
    // static block will be called as class get
       load.
    }
}
```

! As class loads the static block is called.
! It has highest preferance than instance block and constructor.
! It only run once when class is loaded for first-time. or when first obj created.

Preference

Static block // called when ~~obj~~ class load
↓
Instance block // called when obj created
↓
Constructor. // called after creation of object.

* Methods : A method is a block of code that performs specific task.

▷ Let's take an example if we are calculating avg. of number pair 5 times, we can use method to avoid writing code for five times to perform same task.
▷ Code reusablity increases due to use of methods.
▷ A method is a function written inside the class.
▷ As java is an OOP language we need to write methods inside some class.

Types of Methods
1> User defined methods :- we can create as per req.
2> Standard library methods :- Built-in methods in java.

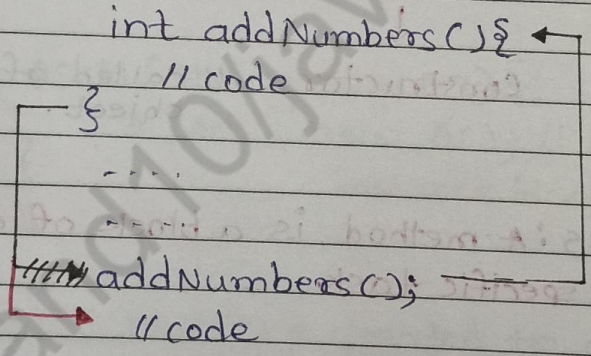
Declearing Java Method's

```
returnType methodName( ){
    //method body
}
```

! If method does not return any value then its return type is void

Calling method :-

```
addNumbers( ); //calls the method.
```



! A variable declared inside the method is called as local variable.

Static keyword in Java

static keyword is used when we want to access class members without creating an instance of the class / variable of the class / object of the class.

▷ static methods

```
class A{
    public static void sum(){...} // static method
}

class Main{
    A.sum(); //Invoking the static method
             directly using '.' (dot operator) by
```

▷ Process of method invocation in Java.

```
// consider method sum
int sum(int a, int b)
 {
    return a+b;
 }
// The method is called like this
   A obj = new A();
   obj.sum(2,3);
```

⇒ The values 2 and 3 are copied to a and b and then a+b returned which is integer value.

! Note: In the case of arrays the reference is passed same is the case for the object passing to methods.

▷ Method overloading.

Two or more methods having same name but different parameters, there methods are called as overloaded methods. And process is

called as method overloading.

```
sum();
sum(int a);
sum(int a, int b);
```
} overloaded methods for sum method.

▶ This is same for constructors called as constructor overloading.

! Note : Method overloading cannot be performed by changing the return type of the methods.

▶ Variable arguments (varargs)
A method in which we can pass values as much we want.

```
public static void sum(int ... arr){
// arr is avalible here as int arr[]
int sum=0; for(int a : arr){
        sum = sum+a;
    }
    sout(sum);
}
```

▶ sum can be called with zero or more arguments.
sum(1,2), sum(1,2,3), sum(2,3,8,10), etc

▶ Packages : are used to arranging.
com.anand.main class A{...}
This means there is com folder in that there is anand folder in that folder there is main folder

in that folder there is file A or class A.

Static

Static get called as the class loads.

Static do not need any object to call it or anything that is static is not depend on objects / instances.

- Static variables :- These are not dependant on objects / instance.
- Anything static is an part of class not of ~~m~~object.
- we can declare variables and methods as static. and they belongs to/ from the class.
- The main method (public static void main(s.....);) is a static method bec it is very first thing that JVM calls and also it should get called without creating an object.

- static is same for all in the class.
e.g. population is same for all the humans on the earth.

- for calling static methods/variables we directly use class name and we can access then because they are the part of the class rather than the object.

Like

```
class A{
    static name = "Anand";
}
```

We can access it like A.name; we don't need to create an object for that call class to access that variable.

⊖ Something which is non-static belongs to an object / instance.

⊖ we can't use non static data in static method [directly without refferencing its instances.] bec non static need instance or obj to access them.

e.g. void greeting (){....}

```
static void main(){
    greeting(); // not allowed to use
                   directly without giving reff.
Main obj = new Main();
 obj.greeting(); // This is allowed be
                    we are giving reff of its.
```

⊖ we can't use this keyword in static method be caus of this points an object.

Inner classes or nested class.

⊖ class in another class

```
class A {          // outer class.
      static class B {....} // inner class
}
```

! outer class should not be static.

Singleton Class. (It is a concept not a keyword)

A class whose only one instance can be created. More than one instance creation for that class is not allowed.

/instances.

```
class Abc{
    static Abc obj = new Abc(); //Static instance of this class.

    private Abc(){ // making default constructor private, bec using that def const. object should not get created.
    }

    public static Abc getInstance(){ // creating a public method which will return an instance which is created. This will allow to create only one instance of class.
        return obj;
    }
}

class singleton{
    public static void main(String[] args){

        Abc obj1 = Abc.getInstance();
// as we can say new Abc(); to create instance bec default constructor is private.
So calling getInstance(); method which will retrn one instance.
due to this only one instance is created. If we create may reff var obj but they will be pointing to the same instance/obj.
        Abc obj2 = Abc.getInstance();
```

new reff var — Pointing to same instance.

obj1 → getInstance();
obj2 →