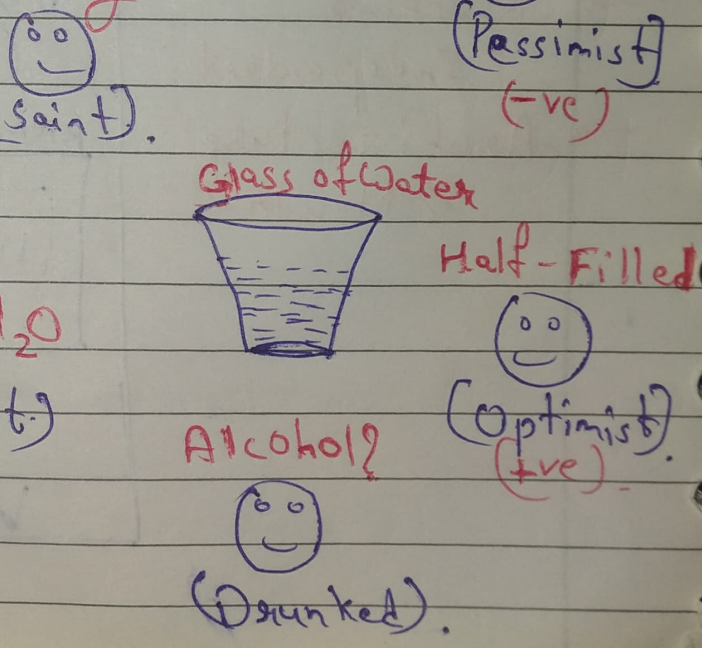
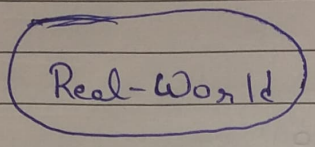
#Object Orientation.

⇒ Object Orientation refers to the Perspective of looking at this world as a collection of object.

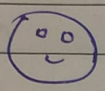
#Orientation:-
→ Point-of-View (POV).
→ Perspective
→ The way of looking at something.

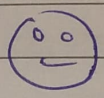


Page No:
Date:



"Collection of Object"



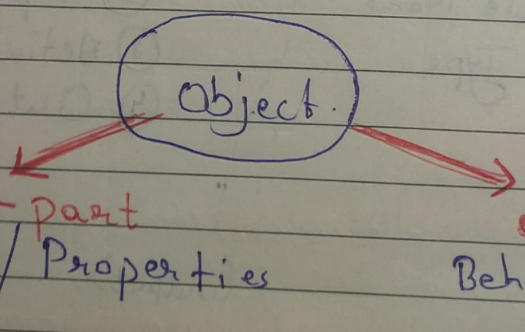


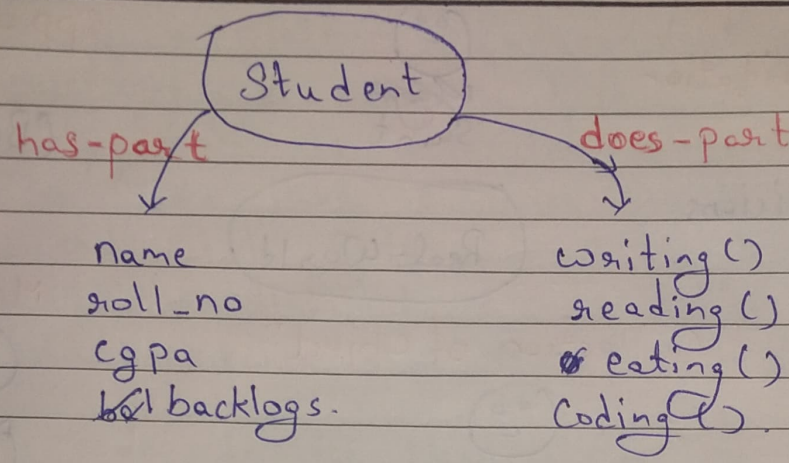
OBJECT-ORIENTATION

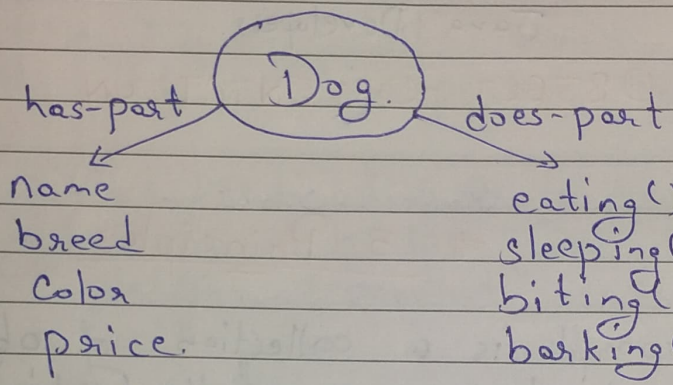
#5 Principles

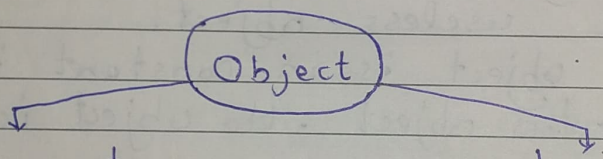
(1) The world is a collection of objects.
(2) Every object is a usefull object. No object is the useless object.
(3) Every object is in constant interaction with some other object. No object is in isolation.

(4) Every object belongs to a certain "type". That type is technically called as a "Class". CLASSES don't exist in reality, Only OBJECTS exist in reality.
(5) Every object has 2 parts.









has-part
State / Properties
Fields / Variables.

does-part.
Behavior / Activity
Methods / Functions.

#General syntax:-

① Variable Name.
② Data Type.

① Method Name.
② Input [Parameters].
③ Activity [Body].
④ Output [Return Type]

```
data_type variable_name;

(output)                    (input).
return_type method_name(Parameters)
                            *Optional
{
    (Activity / Task).
    // Body of the method.
}
```

Page No:
Date:

Printing Output :-

C :- printf("Hello, World!");

C++ :- cout << "Hello, World!";

C# :- Console.WriteLine("Hello, World!");

JavaScript :- console.log("Hello, World!");

Python :- print("Hello, World!");

Java :- System.out.println("Hello, World!");
System.out.print("Hello, World!");
System.out.printf("Hello, World!");

Scanning Input :-

C :- scanf("%s", a);

C++ :- cin >> a;

C# :- a = Console.Readline();

JavaScript :- a = prompt();

Python :- a = input();

Java :- Scanner scan = new Scanner(System.in);
a = scan.next();

```
Class Student.
{
    String name;
    int roll_no;
    float cgpa;
    boolean backlogs;

    void eating()
    {
        System.out.println("Student is eating...");
    }
    void sleeping()
    {
        System.out.println("Student is sleping...");
    }
    void writing()
    {
        System.out.println("Student is writing...");
    }
    void reading()
    {
        System.out.println("Student is reading...");
    }
}
```

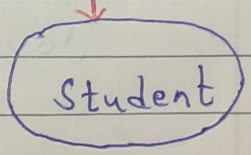
[HAS-PART]

[DOSE-PART]

reference/handle

s1

JVM

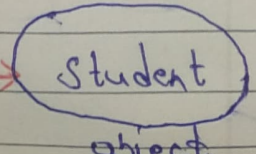


object.

```
Student s1 = new Student();
Student s2 = new Student();

s1.eating();
s2.sleeping();
```

s2

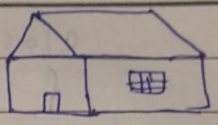


object.

Page No: .....
Date: .....

Blue print of a house.

Melon



Real-Life House.

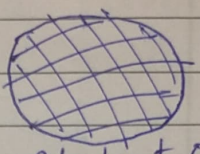
"A class is a blueprint of an object"

```
Class Student
{
}.
```

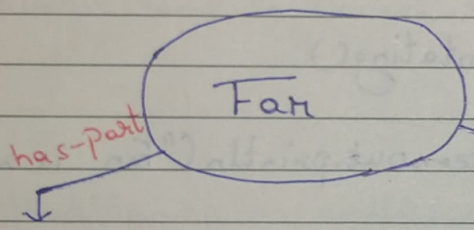
JVM

"new"

"An object is a real life entity"



Student Object.



brand.
color
no_of_blades.
price.
Fields

starting()
rotating()
blowingAir()
stopping().
Methods.

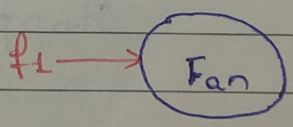
f_1
starting()
stopping()

f_2
starting()
rotating()

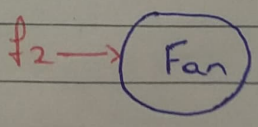
f_3
starting()
rotating()
blowingAir()
stopping().

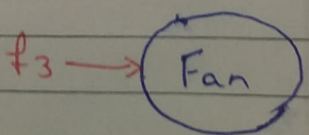
```
class Fan
{
    String brand;
    String color;
    int no_of_blades;
    float price;

    void starting()
    {
        System.out.println("Fan is starting...");
    }
    void rotating()
    {
        System.out.println("Fan is rotating...");
    }
    void blowingAir()
    {
        System.out.println("Fan is blowing air...");
    }
    void stopping()
    {
        System.out.println("Fan is stopping...");
    }
}
```




```
Fan f1 = new Fan();
Fan f2 = new Fan();
Fan f3 = new Fan();
```






```
f1.starting();
f1.stopping();

f2.starting();
f2.rotating();
```

Java is a Case-sensitive Programming Language.

```
f3.starting();
f3.rotating();
f3.blowingAir();
f3.stopping();
```

Naming Conventions:-

1. PascalCase Convention.
2. camelCase Convention.
3. kebab-case Convention.
4. snake_case Convention.

Naming Conventions In Java:-

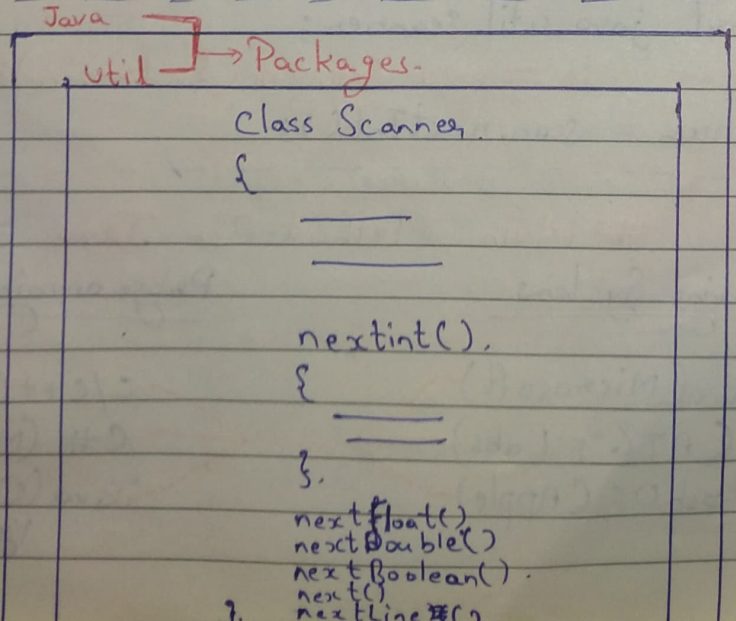
1. ClassName :- PascalCase Convention - Noun.
   Fan
   ElectricFan

2. variableName :- camelCase Convention - Noun.
   brand
   numberOfBlades

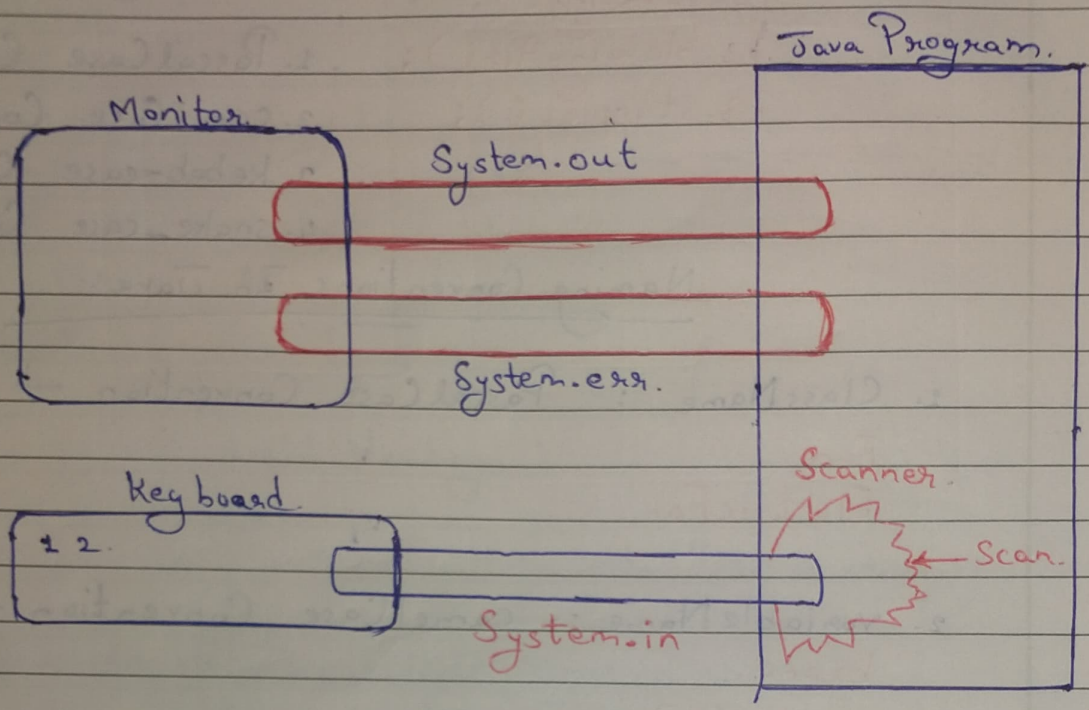
3. methodName :- camelCase Convention - Verb.
   starting()
   blowingAir()
   blowingColdAir()

   (....able).

4. InterfaceName :- PascalCase Convention - Adjective
   Runnable
   Collable
   Calculatable



Input - Output [I/O] in Java [STREAMS]



* Output :-

To display output messages :-

```
System.out.println("Welcome to PW!");
```

To display error message :-

```
System.err.println("Invalid Input!");
```

* Input :-

```
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
int num = scan.nextInt();
```

Main Method In Java.

Operating Systems

- Windows (Microsoft)
- Linux (AT&T Labs)
- MacintoshOS (Apple)

→ main() ←

Programming Languages.

- C/C++ (AT&T Labs)
- C# (Microsoft)
- Java (Oracle) / Python (PSF).

Date: .....

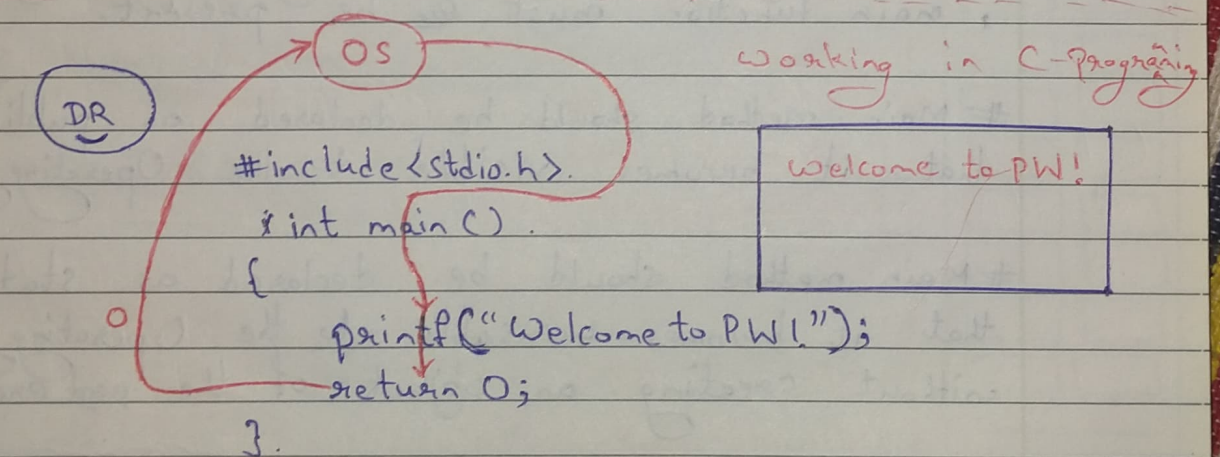
Main method is the entry point or starting point of execution with in a program.

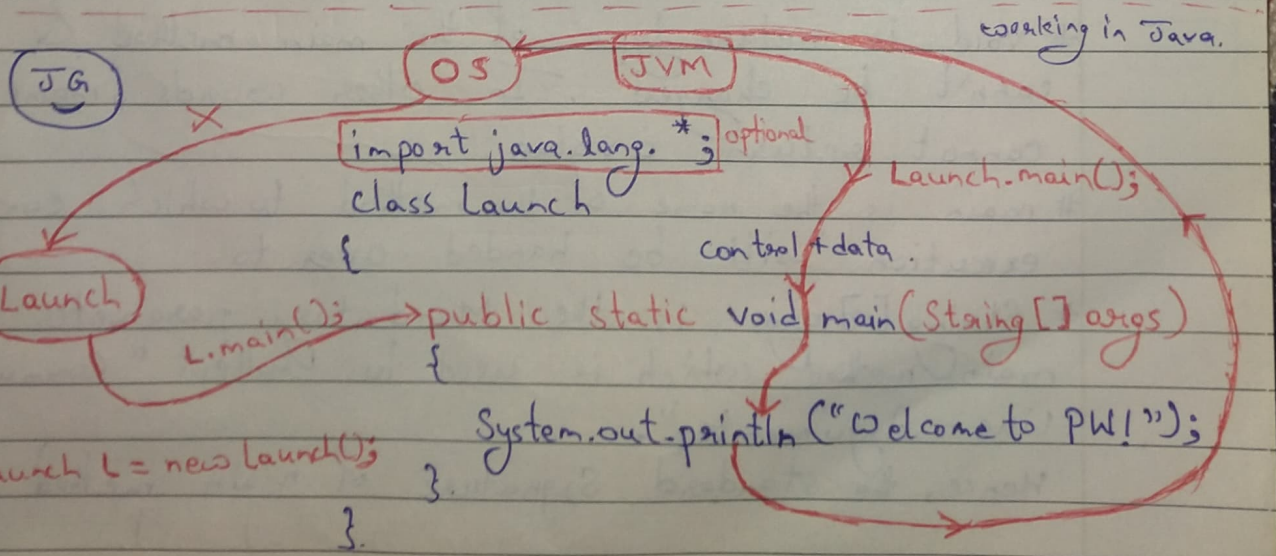
Every method of function must have the following 4 Component.

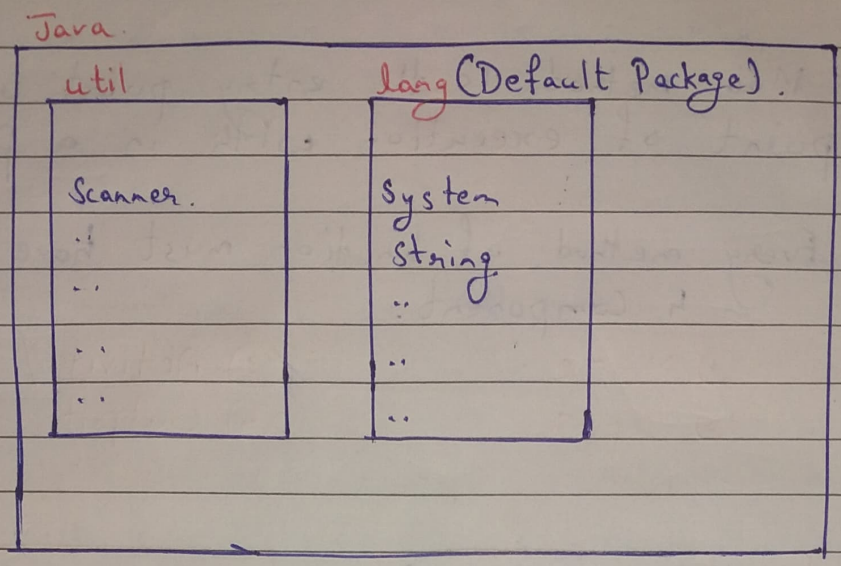
1. Name
2. Input
3. Activity / Task.
4. Output

```
output name(input)
{
    // Activity/Task.
}.
```

```
return_type method_name(parameters)
{
    // Body of the method.
}.
```







The most important software in a computer is the Operating System (OS). As the name suggests it makes the system operational

⇒ For any OS to handover the control of execution to a program written in any programming language, main function must be present.

Main method should be declared as public so that it becomes visible to the Operating System [JVM].

Main method should be declared as static so that it becomes accesable to the Operating System without creating an object of the enclosing class.

Void is return type of the main method & it cann't be changed. In other words, main() cannot return a value.

main is the name of the method to which control of execution will be handed over to.

String [] args is the Mandatory parameter of the main method which is used to collect "Command Line Arguments".

Hence, the standard Signature of main method is.

Public static void main(String[] args)