Introduction to OOP

→ Object oriented programming tries to map code instructions with real world, making the code short & easier to understand.

→ With the help of OOPs, we try to implement real world ~~problems~~ entities such as object, inheritance & abstraction etc.

→ OOPs helps us to follow the DRY (Don't repeat yourself) approach of programming, which in turn increases the reusability of the code.

Two most important aspects of OOP- Classes & Objects

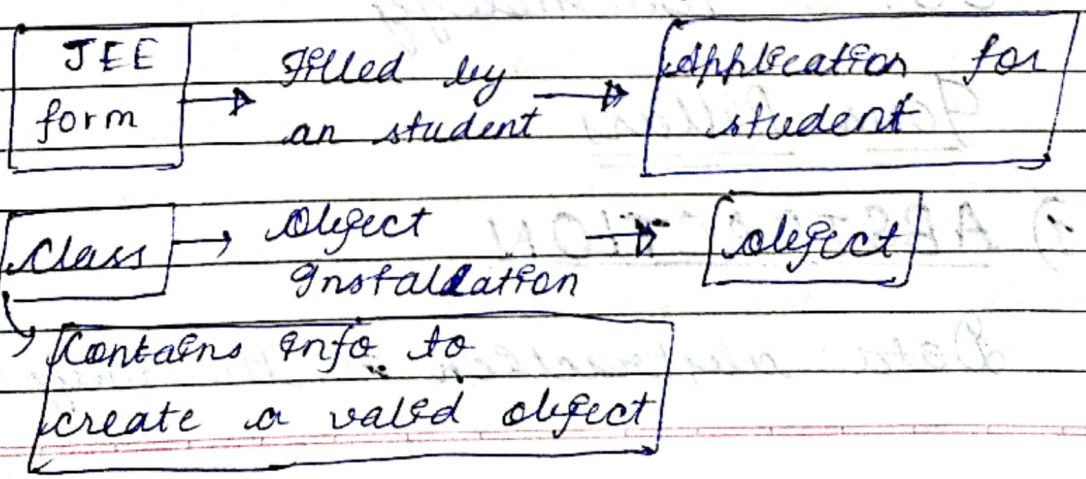
Class:

→ A class is a blue print for creating objects

→ Classes don't consume any space in memory.

→ Objects inherit methods & variables from the class.

→ It is a logical component.



Objects :

→ An object is a instantiation of a class. When a class is defined, a template info is defined.
→ Every object has some address, & it occupies some space in the memory.
→ It is a physical quantity.

For e.g.

Class:	Objects
Fruits	1) Mango
	2) Guava
	3) Grapes

How to model a problem in OOP

We identify the following:

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verb → Methods → getSalary(), increment()

OOP's Terminology

Four Pillars

1) ABSTRACTION

Data abstraction is the way through

which only the essential info is shown to the user, & all internal details remain hidden from the user.

E.g. We use the phone without bothering how it was made.

2) ENCAPSULATION

→ The act of putting various components together (in a capsule).
→ In Java, the variables & methods are the components that are wrapped inside a class.
→ All methods & variables of a class remain hidden from any other class.

For e.g. Chocolates are wrapped inside a chocolate vending machine

3) INHERITANCE

→ The act of deriving new things from existing things.
→ In Java, one class can acquire all the properties & behaviour of other some classes.
→ The class which inherits some other class is called as child class or sub-class.
→ The class which is inherited is known as parent class or super class.

→ Increases reusability of code.

E.g. Rickshaw → E-Rickshaw
Phone → Smartphone

4) POLYMORPHISM

→ One entity, many forms.
→ poly means many, morphs means forms
→ In OOPs, polymorphism helps to perform ~~one~~ single task in many ways.
→ For e.g.
A woman can be a mother, sister, wife, daughter etc.
A smartphone can work like a camera, calculator etc.

Custom Class

Syntax of Custom class

Syntax

```
class <class_name> {
    field;
    method;
}
```

E.g.

```
public class Employee {
    int id;          // Attribute - 1
    String name;     // ---- " ---- - 2
}
```

Note: The first letter of a class should always be capital

→ Any real world object = Prop. + Behaviours

→ An Object in OOPs = Attributes + Methods

A class with methods

E.g.

```
package com.company;

class Employee {
    int id;
    int salary;
    String name;

    public void printDetails() {

        System.out.println("My id is " + id);
        System.out.println("& my name is " + name);
    }

    public int getSalary() {
        return salary;
    }
}
```

```
public class custom{

    p s v m (String [] args)
    {
        Sout t  // Instantiating a new emp.
    Employee John = new Employee();

// setting Attributes
    John.id = 17;
    John.salary = 12;
    John.name = John Khandelwal;

// Printing Attributes

    John.printDetails();
    Int salary = John.getSalary();
    Sout.ln(salary);

    }
}
```

O/P:

My id is 17.
& my name is John Khandelwal
12

Practice questions on OOPs

Q1 Create a class Employee with the foll. properties & methods

→ salary (property) (int)
→ getSalary (method returning int)
→ name (property) (String)
→ getName (method returning String)
→ setName (method changing name)

Q.2 Create a class square with a method to initialize its side, cal. area, perimeter etc.

```
package com.company

class Employee{
    int salary;
    String name;

    public int getSalary(){
        return salary;
    }
    public String getName(){
        return name;
    }
    public void setName(String n){
        name = n;
    }
}
```

```
class Square {
    int side;

public int area() {
        return side * side;
    }

public int perimeter() {
        return 4 * side;
    }

}

public class OOP {

    p s v m (String [] args) {

Employee harry = new Employee();
harry.setName("CWH");
harry.salary = 233;
Sout(harry.getSalary());
Sout(harry.getName());

Square sq = new Square();
sq.side = 3;
Sout(sq.area());
Sout(sq.perimeter());

    }
}
```

o/p

238

CWH

9.

12