

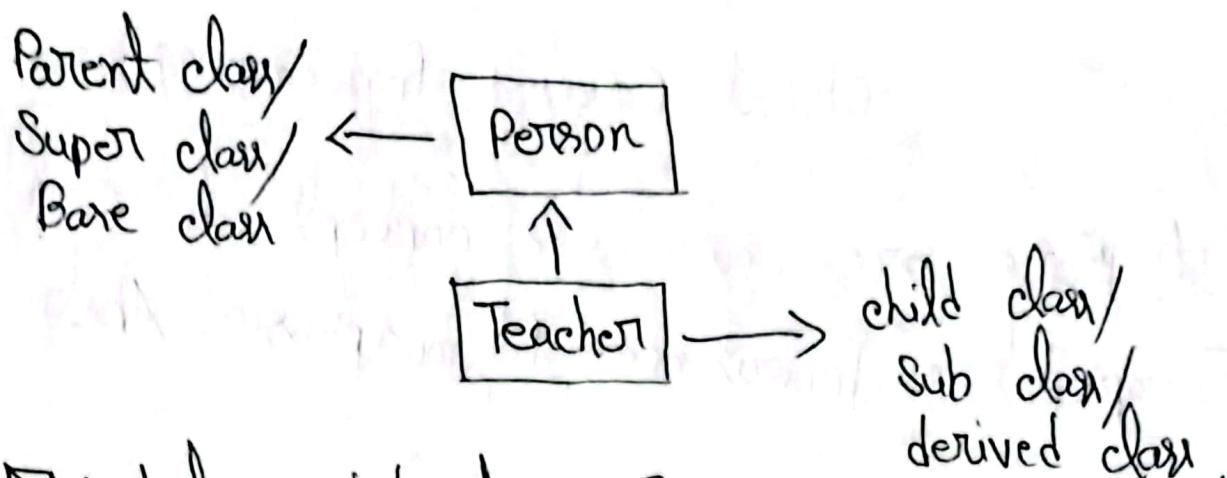
Object Oriented Programming

There are four core concept in OOP.
Encapsulation, Inheritance, Polymorphism, Abstraction.

Inheritance:

Some elements extend from another class.

```
public class person {
    String name;
    int age;
    void displayInformation1() {
        System.out.println("Name :" + name);
        System.out.println("Age :" + age);
    }
}
public class teacher extends person {
    String qualification;
    void displayInformation2() {
        displayInformation1();
        System.out.println("Qualification :" + qualification);
    }
}
```



What is inheritance ?

⇒ It can be define as the process where one class acquires the properties of another.

Why we need inheritance ?

⇒ i) For code Reusability .

ii) For method overriding .

iii) To implement parent - child relationship .

Practical :

* Write a class called Human with variables name & nationality. Create another class called Player & extends it from human. Add the variable position to Player & add a method to print all the variables in Player .

→

class Human {

String name;

String nationality;

public Human(String name, String nationality) {

this.name = name;

this.nationality = nationality;

}

public void displayInfo() {

System.out.println("Name : " + name);

System.out.println("Nationality : " + nationality);

.

}

class Player extends Human {

String position;

public Player(String name, String nationality, String position) {

super(name, nationality);

this.position = position;

}

```
public void displayInfo()
```

```
    super.displayInfo();
```

```
    System.out.println("Position :" + position);
```

```
}
```

```
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Player P = new Player("Jehan", "Bangladeshi", "Forward")
```

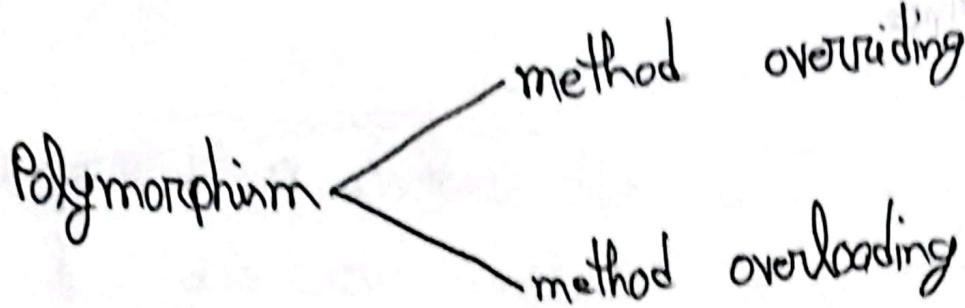
```
        P.displayInfo();
```

```
}
```

```
}
```

OOP :

16/10/23



```
public class Animal {  
    void eat(){  
        SOP("Eating");  
    }  
}
```

```
class Dog extends Animal {  
    void eat (int n){  
        SOP("Dog ate " + n + " times");  
    }  
}
```

public class Shaped

void calculateArea

Output:

Calculating

public class Shaped

void calculateArea()

SOP("Calculating...");

}

}

class Rectangle extends Shaped

double width, height;

(width * height) * 902

}

C

Kazi-min
Kazi-bhai

Illegible signature

Eco -
15.10.23

15.10.23

15.10.23

>Create a football class with method "play". Create two sub-classes called "forward" & "goalkeeper". override the play method & print "Forward is attacking" & "Goalkeeper is saving" sequentially.

public class Football

void play()

}
} OOPS

class Forward extends Football

void play()

SOP("forward is attacking.");

}

}



GDP

class goalkeeper extends football

void play()

SOP("Goalkeeper is saving.");

public class test

public static void main(String[] args){

football f = new football();

f.play();

goalkeeper g = new goalkeeper();

g.play();

}

}

810-211-888

OOP

18/10/23

Encapsulation:

→ Default → public, → private, → protected.

↓
Access modifier

```
public class Student {  
    private int age;  
    private String name;  
    public void setAge(int age){  
        this.age = age;  
    }  
    public int getAge(){  
        return age;  
    }  
}
```

```
public class Test {
```

```
    public static void main(String[] args) {
```

```
        Student s = new Student();
```

```
        s.setAge(25);
```

```
        System.out.println("Age is: " + s.getAge());
```

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

```
public class Student {
    private int age;
    public void setAge(int age) {
        if (age >= 18) {
            this.age = age;
        } else {
            System.out.println("Not accepted");
        }
    }
    public int getAge() {
        return age;
    }
}
public class Test {
    public static void main(String[] args) {
        Student s = new Student();
        s.setAge(7);
        System.out.println("Age is :" + s.getAge());
    }
}
```

Thread:

Thread allows to operate more efficiently by doing multiple things at a same time.

- It works in the background without interrupting the main program.

Example:

```
public class Main extends Thread{  
    //code  
}
```

```
import java.util.*;  
public class Test {  
    public static void main (String [] args) {  
        Scanner sc = new Scanner (System.in);  
        try {  
            int a = sc.nextInt();  
            int d = sc.nextInt();  
            int result = a/d ;  
            System.out.println ("Division result: " + result);  
        }  
        catch (Exception e) {  
            System.out.println ("Error found" + e.getMessage());  
        }  
    }  
}
```

```
public class Test {
    public static void main(String[] args) {
        try {
            catch (Exception e) {
                System.out.println("Error found: " + e.getMessage());
                System.out.println("Enter integer value");
            }
        }
    }
}
```

Exception is used so that if there is any errors in any line then it won't effect the other lines and the code will run.

```
public test{  
    public static void main (String [] args){  
        Scanner sc = new Scanner (System.in);  
        try {  
            int a = sc.nextInt();  
            if (a < 0){  
                throw a new Exception ("Input positive is not valid no.");  
            }  
        } catch (Exception e){  
            System.out.println ("Error" + e.getMessage());  
            a = sc.nextInt();  
            System.out.println (a);  
        }  
        try {  
            int b = sc.nextInt();  
            if (b  
            while (true)
```