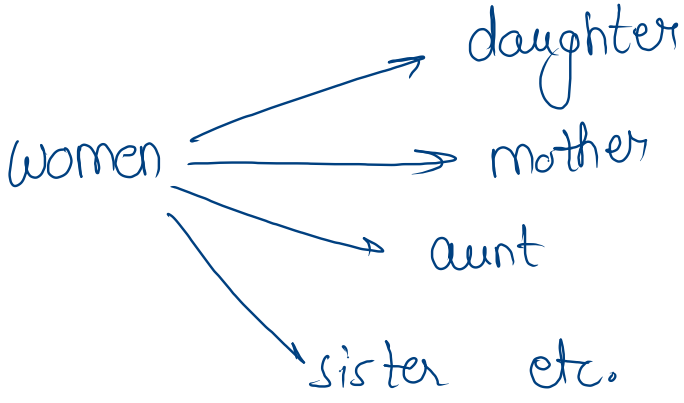
⇒ Polymorphism (many forms)

many forms

define

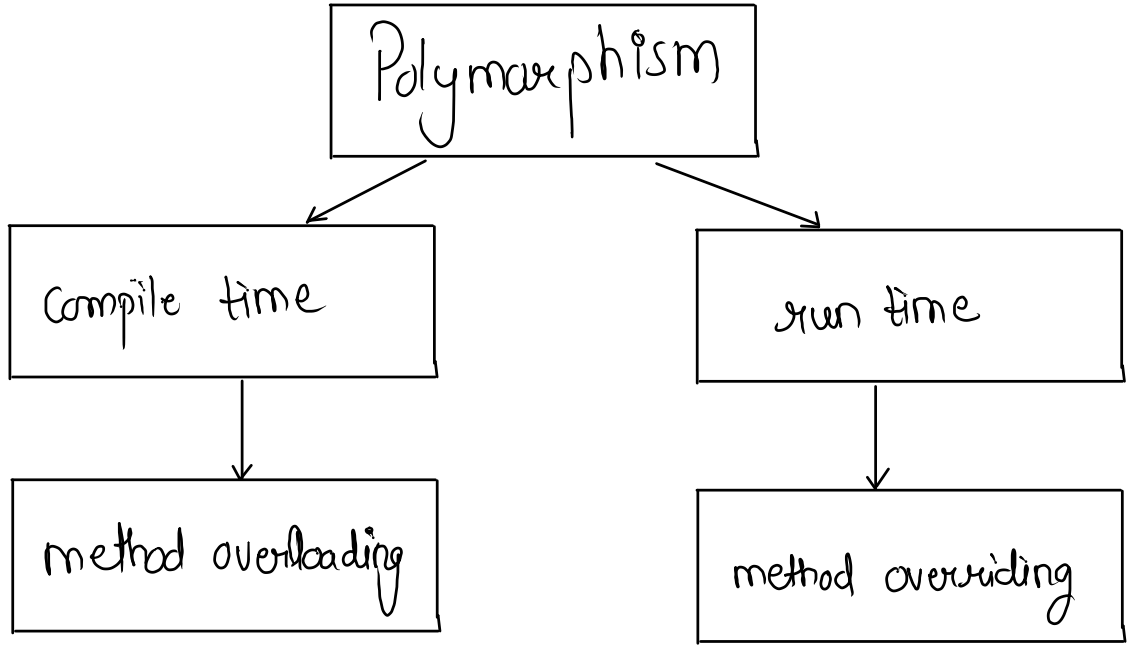
Polymorphism allows you to define one interface and have multiple implementations.



⇒ Types of polymorphism

1) Comple time polymorphism

2) Run time polymorphism



⇒ Method Overloading

↳ method overloading in java refers to the ability to define multiple methods in same class with same name but different no. of parameters and different type of parameters.

Note :- This is also known as static poly. and it identifies at compile time

Code

```
class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public int add(int a, int b, int c) {
        return a + b + c;
    }
    public double add(double a, double b) {
        return a + b;
    }
    public String add(String a, String b) {
        return a + b;
    }
    public static void main(String[] args) {
        Calculator obj = new Calculator();

        System.out.println( obj.add(5, 6) );
        System.out.println( obj.add(5, 6, 7) );
        System.out.println( obj.add(5.0, 6.2) );
        System.out.println( obj.add("hello", "world") );
    }
}
```

output

```
Finished in 56 ms
11
18
11.2
helloworld
```

⇒ Method Overriding

↳ Method overriding in java occurs when subclass provides a specific implementation of a method that is already defined in its superclass.

Note:- This is also known as Dynamic method dispatch.
and it identifies at runtime.

code

```
static class Animal {
    public void makeSound() {
        System.out.println("Animal is making sound");
    }
}
static class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog is barking");
    }
}
static class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat is meowing");
    }
}
public static void main(String[] args) {
    Animal animal1 = new Animal();
    animal1.makeSound();

    Animal animal2 = new Dog();
    animal2.makeSound();

    Animal animal3 = new Cat();
    animal3.makeSound();
}
```

output

```
Finished in 62 ms
Animal is making sound
Dog is barking
Cat is meowing
```

⇒ Advantages

- Code reusability
- Flexibility
- Simplifaction code
- Enhanced maintanance
- Support for interface

etc.

⇒ Advantages

- Code reusability
- Flexibility
- Simplifaction code
- Enhanced maintanance
- Support for interface

etc.

⟹ Abstraction (Data Hiding)

↳ abstraction in java refers to hiding the implementation details of a code and exposing only the necessary information to the user.

Types of abstraction

- Abstract classes
- Interface

→ What is an abstract class

↳ An abstract class is a class that cannot be instantiated on its own and typically serves as a blueprint to other class.

Note:- to define an abstract class, we uses the keyword abstract

Imp:- When even a single method in a class is abstract then that class has to be abstract.

Code

```
abstract class Shape {
    public abstract void area();
    public void display() {
        System.out.println("This is a shape");
    }
}
class Rectangle extends Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public void area() {
        System.out.println( "Area of rectange is " + (length * width) );
    }
}
class Circle extends Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public void area() {
        System.out.println("Area of Circle is " + (Math.PI * radius * radius));
    }
}
class Main {
    public static void main(String[] args) {
        Shape rect = new Rectangle(4.0, 5.0);
        rect.display();
        rect.area();

        Shape circle = new Circle(4.0);
        circle.display();
        circle.area();
    }
}
```

output

```
Finished in 59 ms
This is a shape
Area of rectange is 20.0
This is a shape
Area of Circle is 50.26548245743669
```

Note :-

Abstract class provides 0 to 100% abstract

Interface provides 100% abstract

169. Majority Element

0	1	2	3	4	5	6	7	8	9	10
2	2	3	2	4	4	4	2	2	2	2

arr = [2, 2, 3, 2, 4, 4, 4, 2, 2, 2, 2]

approch 1 :- find freq of each S.C = O(n) ✗

approch 2 :- sort the array T.C = O(n log(n)) ✗

T.C = O(n) , S.C = O(1)