What is OOPs in java:

Pillars of OOPs:

1. **Encapsulation.**
2. **Inheritance.**
3. **Polymorphism.**
4. **Abstraction.**

Encapsulation:

- **What is encapsulation?**

1. Data Hiding.
2. Security to the data member.
3. Protecting the field.
4. Binding.

- **how to implement encapsulation?**

1. It can be achieved by Using the "Private" key word Setter /Getter/Constructor and their shortcut.
2. **This** keyword in Encapsulation.

- **What are the advantages of Encapsulation?**

1. Not giving direct access to your properties.
2. protecting your data by using encapsulation.
3. Controlling the Assigning

Example Problem:

```
class Person{
        private String name= "Tilak";
        private int age;
        private String address;
```

```
	private Boolean married;

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public Boolean isMarried() {
		return married;
	}
	public void getMarried(Boolean married) {
		this.married = married;
	}



}
public class Demo1 {

	public static void main(String[] args) {
	Person P = new Person();
	P.setAge(20);
	P.setName("Vikram");
	P.setAddress("12/24 mint street");



	System.out.println(P.getAddress());
	System.out.println(P.getAge());
	System.out.println(P.getName());
	}
```

}

Inheritance:

- **What is Inheritance in Java?**
  1. Inheriting once class functions in another class .
  2. Making a relationship of parent and child.

- **How to achieve inheritance**
  1. By using extends key word.
  2. Usage of **This()** and **Super()**;
  3. One Program to show this() and Super() difference
     Example:

```
class Data2{
        int age;
        String name;

        public Data2() {
                this(7,"nhn" );
                System.out.println("i am in parent Class Constructor ");
        }
        public Data2(int age,String name) {
                System.out.println(age +" " + name);
        }
}
class Data extends Data2{
        int age;
        String name;

public Data() {
        System.out.println("Vikram is here in no para constructor");
}
public  Data(int age,String name) {
      super();
        this.age= age;
        this.name=name;
        System.out.println(age  + " " + name);
}
}

public class Students {

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

```
		Data D = new Data(23,"vikram");

	}

}
```

One Example to show Inheritance achieving loose coupling and downcasting.

Example:

```
class Parent{
	public void m1() {
		System.out.println("i am a parent class m1 method");
	}
	public void m2() {
		System.out.println("i am a parent class m2 method");
	}
public void m3() {
	System.out.println("i am a parent class m3 method");
	}
public void m4() {
	System.out.println("i am a parent class m4 method");
}
}

class Child extends Parent{
	@Override
	public void m2() {
		System.out.println("i am a child class m2 method");
	}

	//Specialized methods
	public void m5() {
		System.out.println("i am a child class m5 method");
	}
}
public class Demo3 {

	public static void main(String[] args) {
	Parent P = new Child();//loose Coupling
	((Child) P).m5();//  ( specialized method)// down casting
	P.m3();//Parent Class method
	P.m1();
	P.m2();//Override method

	}
```

```
        }
```

Example1:

```
class Student{
        private  String name="vikram";//instance variable //fields
/properties
        private int age =30;
        private int rollnumber;

        public  Student() {
                System.out.println("am i a Constructor");
        }
        public Student(String name, int age) {
                this.age=age;
                this.name=name;
        }
        public Student(int age, String name) {
                this.age=age;
                this.name=name;;
        }




        public String getName() {
                return name;
        }

        public int getAge() {
```

```
            return age;
        }

        public int getRollnumber() {
            return rollnumber;
        }









}

public class Demo1 {

        public static void main(String[] args) {

        Student S = new Student();
        Student S1= new Student("Tilak",28);


   System.out.println(S.getName());
   System.out.println(S.getAge());
//   System.out.println(S.getRollnumber());


   System.out.println(S1.getName());
   System.out.println(S1.getAge());
//   System.out.println(S1.getRollnumber());

        }

}
```

Example2:

```
class Car{
	public void color() {
		System.out.println("my car color is red");
	}
public void speed() {
	System.out.println("my car speed is 200km/hr");
	}


}

class Tata extends Car {

	@Override
	public void speed() {
		System.out.println("my car speed is 320km/hr");
		}

	//specialized method

	public void cost() {
		System.out.println("my car cost is 2500000rs");
	}
}

public class Demo2 {

	public static void main(String[] args) {
		Car C = new Tata();// loose coupling
		 C.color();

		 ((Tata) C).cost();// down casting

	}

}
```

Example3:

```
class Animal{
        String name;
        int age;

    public Animal() {
        this(24,"jack");// it will search for a constructor in the same
class which take string type of data and has one para
        System.out.println(" hy i am in no para animal method");
    }

    public Animal(String name) {

        this.name=name;
        System.out.println(name);
    }
    public Animal(int age,String name) {
        this.name=name;
        this.age=age;
        System.out.println(name  +" " + age );
    }


}

class Dog extends Animal {
        int speed ;
        int number;

        public Dog() {
        this(0,1234);
                System.out.println("i am in no para of Dog method");
        }

        public Dog(int speed ) {
```

```
            this();
            this.speed=speed;
            System.out.println(speed);
        }
        public Dog(int speed ,int number) {

            this.speed=speed;
            System.out.println(speed);
        }



}
public class Demo3 {

        public static void main(String[] args) {
    Dog D = new Dog(300);


        }

}
```