Write a Program where you inherit method from parent class and show method Overridden Concept?

```
class Vehicle
{
void run()
{
System.out.println("Vehicle is running");
}
 }
class class Bike2 extends Vehicle{
void run()
{System.out.println("Bike is running safely");
} public static void main(String args[])
{ Bike2 obj = new Bike2();
obj.run();
} }
```

Write a program to show run time polymorphism in java?

```
class Bike{
 void run()
{
System.out.println("running");
}
 }
 class Splendor extends Bike
```

```
{
 void run()
{System.out.println("running safely with 60km");
}
public static void main(String args[])
{ Bike b = new Splendor();
b.run();
}
}
```

Achieve loose coupling in java by using OOPs concept?

To achieve loose coupling, one should use abstract classes or interface while performing inheritance and another class extends or implement the interface by using the inherit concept of Object Oriented Programming Language.

Write a program to show Compile time polymorphism in java?

```
public class MethodOverloading
{
Void show
{ System.out.println("number 1 : " + num1);
 }
 void show(int num1, int num2)
{ System.out.println("number 1 : " + num1 + "  number 2 : " + num2);
 }
 public static void main(String[] args)
{
```

```
 MethodOverloading obj = new MethodOverloading();
obj.show(3);
obj.show(4, 5);
} }
```

What is the benefit of encapsulation in java?

Encapsulation prevents access to data members and data methods by any external classes.

1.Flexible Programs

2.Easy debugging & testing

3.Reusability

4.Hiding data

Is java a t 100% Object oriented Programming language? If no why ?

Java is not a fully object-oriented language as it supports primitive data types like int, byte, long, short, etc., which are not objects.

But The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.

What are the advantages of abstraction in java?

1. Reduces Complexity
2. Increase Security
3. Increase reusability
4. Avoid code duplication
5. Loose coupling

What is an abstraction explained with an Example?

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Abstract class in Java

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

```
abstract class Bike{
  abstract void run();
}
class HeroHonda extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new HeroHonda();
 obj.run();
}
}
```

Interface

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface

interfaces can have abstract methods and variables. It cannot have a method body.

There are mainly three reasons to use interface. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of **multiple** inheritance.

It can be used to achieve loose coupling.

```
interface Animal {

  public void animalSound();

}
```

```
class Dog implements Animal {

  public void animalSound() {

   System.out.println("Dog braking");

}

}
```

What is the final class in Java?

The final class is a class that is declared with the final keyword. Subclasses can't inherit a final class or a final class cannot be inherited by any subclass. So, we can restrict class inheritance by making use of a final class.