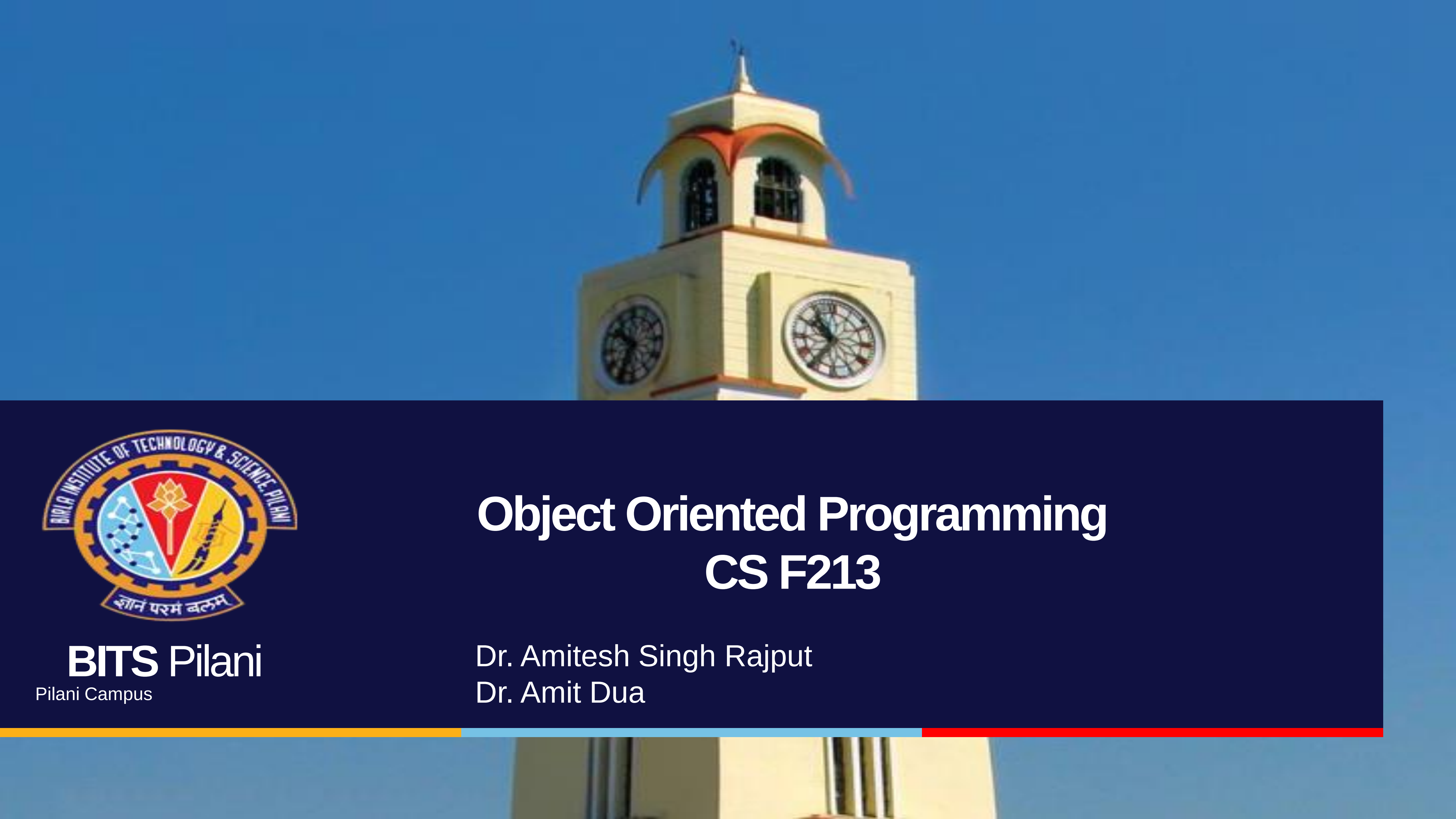


BITS Pilani
Pilani Campus

Object Oriented Programming
CS F213

Dr. Amitesh Singh Rajput
Dr. Amit Dua

Exercise 1

```
class Base {
   private void fun() {
       System.out.println("Base fun");
   }
}

class Derived extends Base {
   private void fun() {
       System.out.println("Derived fun");
   }
}
```

```
Class Main {
public static void main(String[ ] args){
        Base obj = new Derived();
        obj.fun();
   }
}
```

error: fun() has private access in Base

Exercise 2

```
class Base {
  static void fun() {
     System.out.println("Base fun");
  }
}

class Derived extends Base {
  static void fun() {
     System.out.println("Derived fun");
  }
}
```

```
Class Main {
public static void main(String[ ] args){
      Base obj = new Derived();
      obj.fun();
  }
}
```

Base fun

Exercise 3

```
class zero{
 int i = 10;
 float j = 20;
 void show(){
  System.out.println(i+" "+j);
 }
}

class first extends zero  {
 int i = 30;
 float j = 40;
 void show(){
  System.out.println(i+" "+j);
 }
}
```

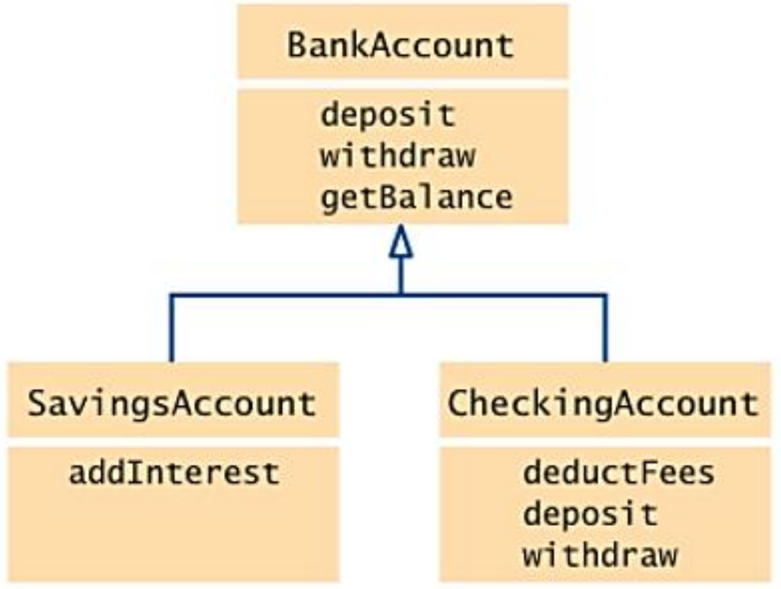
```
class second extends first{
 int i = 50;
 float j = 60;
 void show(){
  System.out.println(i+" "+j);
 }
 public static void main(String args[]){
  zero a = new first();
  a.show();
  first s = new second();
  s.show();
 }
}
```

Output:

```
30 40.0
50 60.0
```

What about
`zero a = new second();`

Bank - Example



```
class TestAccount{
 public static void main(String[] args) {
   Scanner sr = new Scanner(System.in);
   System.out.println("1 for new customers, 0 for others");
   int yr = sr.nextInt();
   BankAccount ba;
   if (yr == 1)
       ba = new BankAccount(111,"Ankit",5000);
   else
       ba = new CheckingAccount(111,"Ankit",5000);
   System.out.println("Initial: " + ba.getBalance());
   ba.deposit(1000);
   ba.withdraw(2000);
   ba.deposit(6000);
   System.out.println("After three Transactions: " + ba.getBalance());
   ba.deductFee();    //ERROR
   System.out.println(ba.getBalance());
   sr.close();
   }}
```

Solution 1

- Create an empty method in the Bank Account class

```
void deductFee()
{

}
```

- Meaningless, Isn't it?

Solution 2 – Abstract Class

```java
abstract class BankAccount{
 private int acc;
 private String name;
 private float amount;

BankAccount(int acc,String name,float amt){
 this.acc = acc;
 this.name = name;
 this.amount = amt;
}

void setAcc(int acc){
 this.acc = acc;
}

void setName(String name){
 this.name = name;
}
float getBalance(){
   return amount;
}
void deposit(float amount){
   this.amount = this.amount+amount;
}
void withdraw(float amount){
   if (this.amount < amount)
    System.out.println("Insufficient Funds");
   else
    this.amount=this.amount-amount;
}
abstract void deductFee();
}
```

Abstract class

- A restricted class that cannot be used to create objects. We can have references of abstract class type.

- Abstract method: Can only be used in an abstract class, and it does not have a body.

- An abstract class can contain constructors.

- We can have an abstract class without any abstract method.

- Abstract classes can also have final methods.

Example abstract class

```
abstract class Base {
 final void fun(){
  System.out.println("fun()
  called");
 }
}
class Derived extends Base { }
```

```
class Main {
public static void main(String args[])
 {
     Base b = new Derived();
     b.fun();
 }
}
```

```
fun() called
```

Questions

- Is it possible to create abstract and final class in Java?
- Is it possible to have an abstract method in a class?
- Is it possible to have an abstract method in a final class?
- Is it possible to inherit from multiple abstract classes in Java?

innovate achieve lead

BITS Pilani
Pilani Campus

Interfaces

Interface

- Interface is a blueprint of a class containing static constants and abstract methods.

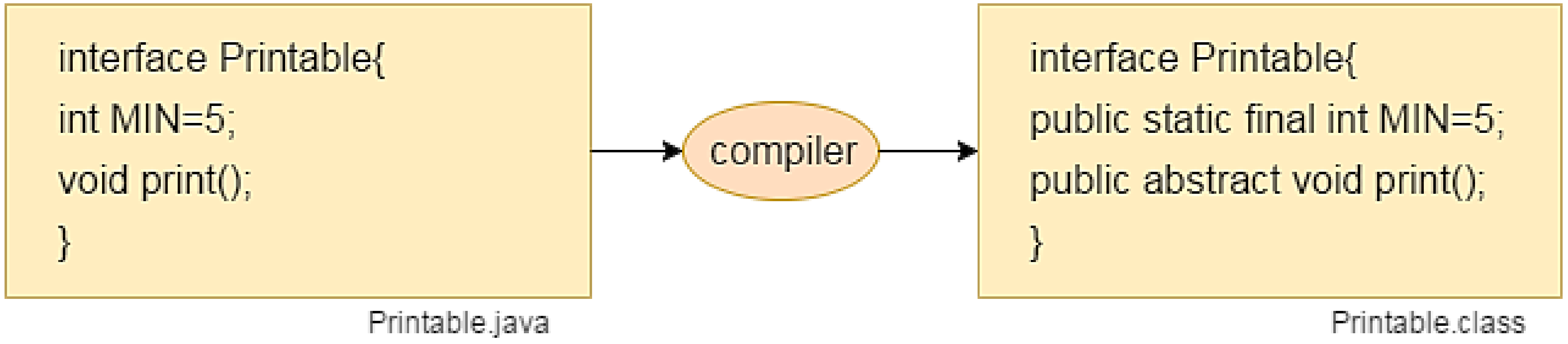
- You can specify what a class must do, but not how it does.

- They are syntactically similar to classes, but they lack instance variables, and their methods are declared without any body.

- Once created, any number of classes can implement an interface.

Internal addition by the compiler

- The Java compiler adds public and abstract keywords before the interface method.
- Also, it adds public, static and final keywords before data members.



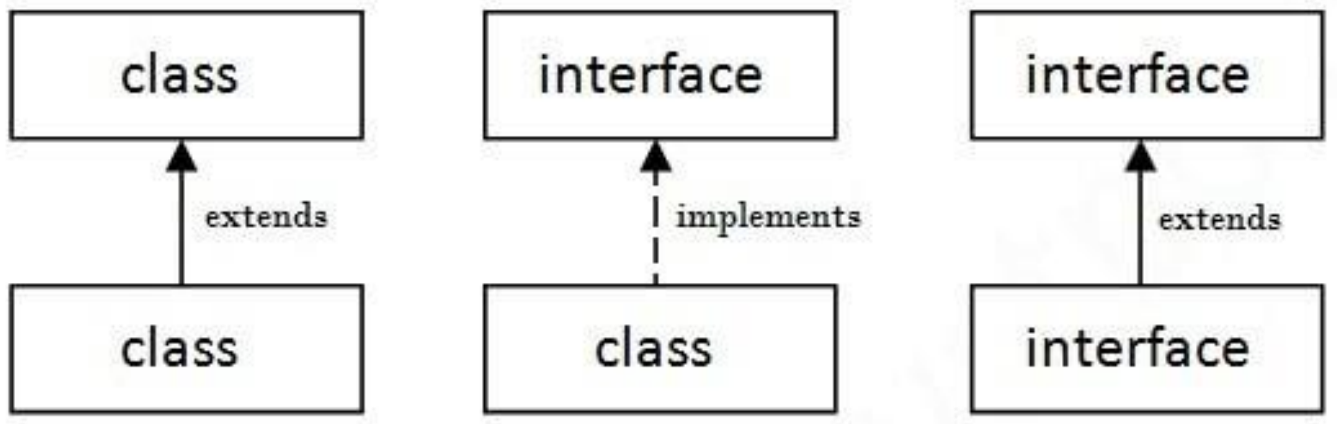
```
interface Printable{
int MIN=5;
void print();
}
```

Printable.java

```
interface Printable{
public static final int MIN=5;
public abstract void print();
}
```

Printable.class

Relationship between Classes and Interfaces



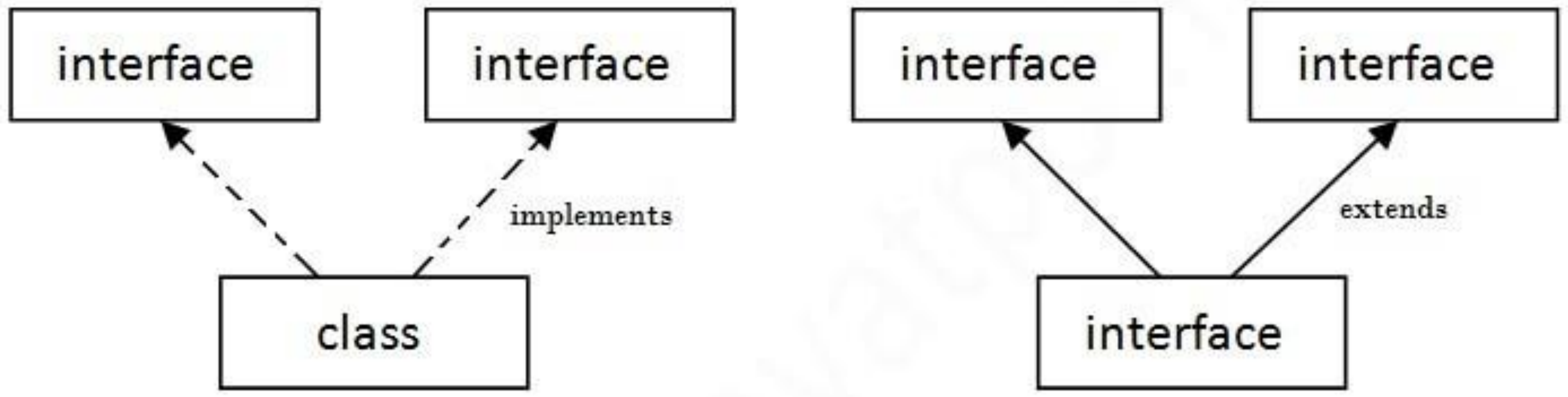
Interfaces - Example

```
interface Bank {
 void deductFee();
 void withdraw(float amount);
}

class BankAccount implements Bank{
   …
   public void deductFee(){ }
}

class CheckingAccount extends BankAccount implements Bank{
 .
 .
 .
}
```

Multiple Inheritance in Interface



Multiple Inheritance in Java

Multiple Inheritance using Interface

```
interface Printable{
   void print();
   void show();
}
interface Showable{
   void show();
   void print();
}

class trial implements Printable, Showable {
   public void show(){
     System.out.println("Within Show");
   }
   public void print() {
     System.out.println("Within Print");
   }
}
```

```
public class test {
 public static void main(String[] args) {
   trial t = new trial();
   t.print();
   t.show();
 }
}
```

```
Within Print
Within Show
```

Default Methods in Interface (defender or virtual extension)

- Before Java 8, interfaces could have only abstract methods. Implementation is provided in a separate class
- If a new method is to be added in an interface, implementation code has to be provided in all the classes implementing the interface.
- To overcome this, default methods are introduced which allow the **interfaces to have methods with implementation** without affecting the classes.

Default Methods

```
interface Printable{
   void print();
   default void show(){
    System.out.println("Within Show");
   }
}

class trial implements Printable {
   public void print(){
    System.out.println("Within Print");
   }
}
```

```
public class test{
   public static void main(String[] args) {
    trial t = new trial();
    t.print();
    t.show();
   }
}
```

Default Methods & Multiple Inheritance

```
interface Printable{
 void print();
 default void show(){
   System.out.println("Within Printable");
 }
}

interface Showable{
 default void show(){
   System.out.println("Within Showable");
 }
 void print();
}
```

```
class trial implements Printable, Showable{
 public void show(){
   Printable.super.show();
   Showable.super.show();
 }
 public void print(){
   System.out.println("Within Print"); }
 }
 public class test{
   public static void main(String[] args){
    trial t = new trial();
    t.print();
    t.show();
   }
}
```

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You!