

CS2304 JAVA PROGRAMMING

Weak 11-INTERFACE

Name: B Satlas rohit

RegNo:2024503305

CODE1: Demonstrates one interface extending another

```
interface A {  
    void methodA();  
}  
  
interface B extends A {  
    void methodB();  
}  
  
class SampleInterface implements B {  
    // @Override  
    // public void methodA() {  
    //     System.out.println("Method from Interface A");  
    // }  
  
    @Override  
    public void methodB() {  
        System.out.println("Method from Interface B");  
    }  
  
    public static void main(String[] args) {  
        System.out.println("Name:B Satlas rohit \nRegno:2024503305");  
        SampleInterface obj = new SampleInterface();  
    }  
}
```

```
    obj.methodA();
    obj.methodB();
}
}
```

OUTPUT1:

```
C:\Users\pebu\IdeaProjects\week11\src\SampleInterface.java:9
java: SampleInterface is not abstract and does not override abstract method methodA() in A
```

OUTPUT2:

```
Name:B Satlas rohit
Regno:2024503305
Method from Interface A
Method from Interface B
```

CODE2:Multiple Inheritance

```
interface Floatable {
    default void repair() {
        System.out.println("Repairing Floatable object");
    }
}

interface Flyable {
    default void repair() {
        System.out.println("Repairing Flyable object");
    }
}

class ArmoredCar implements Floatable, Flyable {
//  @Override
```

```
// public void repair() {  
//     System.out.println("Resolving conflict between Floatable and  
Flyable");  
//     Floatable.super.repair();  
//     Flyable.super.repair();  
// }  
  
public static void main(String[] args) {  
    System.out.println("Name:B Satlas rohit \nRegno:2024503305");  
    ArmoredCar car = new ArmoredCar();  
    car.repair();  
}  
}
```

OUTPUT1:

```
C:\Users\pebu\IdeaProjects\week11\src\MultipleInheritance.java:11  
java: types Floatable and Flyable are incompatible;  
  class ArmoredCar inherits unrelated defaults for repair() from types Flo|atable and Flyable
```

OUTPUT2:

```
Name:B Satlas rohit  
Regno:2024503305  
Resolving conflict between Floatable and Flyable  
Repairing Floatable object  
Repairing Flyable object
```

CODE3:Interface variable

```
interface Constants {
```

```
    int VALUE = 10;
```

```
}
```

```
class VariableTest implements Constants {
```

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

```
        System.out.println("Name:B Satlas rohit \nRegno:2024503305");
```

```
        System.out.println("Initial VALUE: " + VALUE);
```

```
        VALUE = 20;
```

```
}
```

```
}
```

OUTPUT1:

```
java: cannot assign a value to static final variable VALUE
```

OUTPUT2:

```
Name:B Satlas rohit
```

```
Regno:2024503305
```

```
Initial VALUE: 10
```