WEEK 11

1. Create an interface Account having methods- deposit(), withdraw() and aboutBank() (abountBank() is a static method). Create two classes Saving and Current which implement the Account interface. Call the methods of Saving and Current classes in main method.

2. In the previous question, create a new method in Account interface- takeLoan() (takeLoan() is a default method). takeLoan() method would be implemented by Saving class only. Call the methods of Saving and Current classes in main method.

Main.java | Account.java | Current.java | Saving.java

```java
package week11_1;

public class Main {
    public static void main(String[] args) {
        Saving saving = new Saving();
        Current current = new Current();

        saving.deposit();
        saving.withdraw();
        saving.takeLoan();
        current.deposit();
        current.withdraw();
    }
}
```

Main.java | Account.java | Current.java | Saving.java

```java
package week11_1;

public interface Account {  3 usages  2 implementations
    void deposit();  2 usages  2 implementations
    void withdraw();  2 implementations
    static void aboutBank() { System.out.println("This is a bank."); }
    default void takeLoan() { System.out.println("You are eligible for a loan."); }
}
```

WEEK 11

Main.java | Account.java | Current.java | Saving.java

```java
package week11_1;

public class Current implements Account{ 2 usages

    @Override 2 usages
    public void deposit() { System.out.println("Depositing in Current Account."); }

    @Override
    public void withdraw() { System.out.println("Withdrawing from Current Account."); }
}
```

Main.java | Account.java | Current.java | Saving.java

```java
package week11_1;

public class Saving implements Account{ 2 usages

    @Override 2 usages
    public void deposit() { System.out.println("Depositing in Saving Account."); }

    @Override
    public void withdraw() { System.out.println("Withdrawing from Saving Account."); }

    @Override 2 usages
    public void takeLoan() { Account.super.takeLoan(); }
}
```

```
Depositing in Saving Account.
Withdrawing from Saving Account.
You are eligible for a loan.
Depositing in Current Account.
Withdrawing from Current Account.
```

3. Create interfaces Bike and Scooty, both of which have two methods- offer() and details() (details() is default method). Create a new class BuySomething which implements both interfaces. To remove ambiguity, create a method details() in BuySomething class as well in which call the details() method of both interfaces. Call the methods of BuySomething class in main method.

WEEK 11

Main.java | Bike.java | Scooty.java | BuySomething.java

```java
package week11_3;

public class Main {
    public static void main(String[] args) {
        BuySomething buySomething = new BuySomething();
        buySomething.details();
        buySomething.offer();
    }
}
```

Main.java | Bike.java | Scooty.java | BuySomething.java

```java
package week11_3;

public interface Bike {
    void offer();
    default void details() { System.out.println("This bike costs 50k"); }
}
```

Main.java | Bike.java | Scooty.java | BuySomething.java

```java
package week11_3;

public interface Scooty { 2 usages  1 implementation
    void offer();  1 usage  1 implementation
    default void details() { System.out.println("This scooty costs 45k"); }
}
```

Main.java | Bike.java | Scooty.java | BuySomething.java

```java
package week11_3;

public class BuySomething implements Bike,Scooty {
    @Override
    public void offer() { System.out.println("50 percent off"); }

    @Override
    public void details() {
        Bike.super.details();
        Scooty.super.details();
    }
}
```

```
This bike costs 50k
This scooty costs 45k
50 percent off
```

WEEK 11

4. Create two interfaces Printer and Scanner, both having methods connect() and details() (details() is a default method). Create a class MultiFunctionMachine that implements both interfaces. In MultiFunctionMachine, override the details() method to resolve ambiguity and call the details() methods of both interfaces. Call all methods of MultiFunctionMachine in the main() method.

Main.java

```java
package week11_4;

public class Main {
    public static void main(String[] args) {
        MultiFunctionMachine mf = new MultiFunctionMachine();
        mf.connect();
        mf.details();
    }
}
```

Printer.java

```java
package week11_4;

public interface Printer {  2 usages  1 implementation
    void connect();  1 usage  1 implementation
    default void details() { System.out.println("This machine can print"); }
}
```

Scanner.java

```java
package week11_4;

public interface Scanner {
    void connect();  1 usage  1 implementation
    default void details() { System.out.println("This machine can scan"); }
}
```

MultiFunctionMachine.java

```java
package week11_4;

public class MultiFunctionMachine implements Printer, Scanner{  2 usa
    @Override  2 usages
    public void connect() { System.out.println("Connecting..."); }

    @Override
    public void details() {
        Printer.super.details();
        Scanner.super.details();
    }
}
```

```
Connecting...
This machine can print
This machine can scan
```

WEEK 11

5. Create an interface Device with a method powerOn(). Create another interface SmartDevice that extends Device and adds a method connectWiFi(). Create a class SmartPhone that implements SmartDevice. Demonstrate calling both powerOn() and connectWiFi() using a SmartPhone object in the main() method.

Main.java | Device.java | SmartDevice.java | Sn

```
1   package week11_5;
2
3   public class Main {
4       public static void main(String[] args) {
5           SmartPhone smartPhone = new SmartPhone();
6           smartPhone.powerOn();
7           smartPhone.connectWifi();
8       }
9   }
```

Main.java | Device.java

```
1   package week11_5;
2
3   public interface Device {
4       void powerOn();  1 usage
5   }
```

Main.java | Device.java | SmartDevice.java

```
1   package week11_5;
2
3   public interface SmartDevice extends Device{
4       void connectWifi();  1 usage  1 implementation
5   }
```

Main.java | Device.java | SmartDevice.java | SmartPhone.java

```
1   package week11_5;
2
3   public class SmartPhone implements SmartDevice{  2 usages
4
5       @Override  1 usage
6       public void connectWifi() { System.out.println("Connecting..."); }
9
10      @Override  1 usage
11      public void powerOn() { System.out.println("Booting..."); }
14  }
```

```
Booting...
Connecting...
```