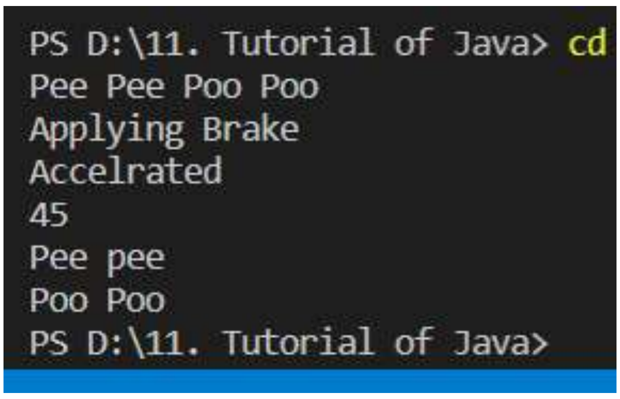
INDEX :

Interfaces :

```java
interface Bicycle{

    int a = 45; // --> By default 'int a = 45' is final.

    void ApplyBrake(int decrement);

    void SpeedUp(int increment);

}


interface HornBicycle{

    void BlowHornK3g(); // By default methods in interface is public.

    public void BlowHornSzzuki();

}


class BMW implements Bicycle, HornBicycle{

    // Class 'BMW' must either be declaired abstract or implement all abstract methods.

    void BlowHorn(){

        System.out.println("Pee Pee Poo Poo");

    }

    @Override

    public void ApplyBrake(int decrement) {

        System.out.println("Applying Brake");

    }


    @Override

    public void SpeedUp(int increment) {

        System.out.println("Accelrated");

    }

    @Override

    public void BlowHornK3g(){

        System.out.println("Pee pee");

    }

    @Override

    public void BlowHornSzzuki(){

        System.out.println("Poo Poo");

    }

}
```

```java
public class X_Interfaces{

    public static void main(String[] args) {

        // Interface :- It is a point where two systems meet and interact, In Java Interface

        // is a group of related methods with empty bodies.

        BMW myBike = new BMW();

        myBike.BlowHorn();

        myBike.ApplyBrake(5);

        myBike.SpeedUp(40);

        System.out.println(myBike.a);

        // We cann't modify the properties in Interfaces as they are final

        // myBike.a = 4554; // --> This will throw error.

        myBike.BlowHornK3g();

        myBike.BlowHornSzzuki();

    }

}
```

Output :-



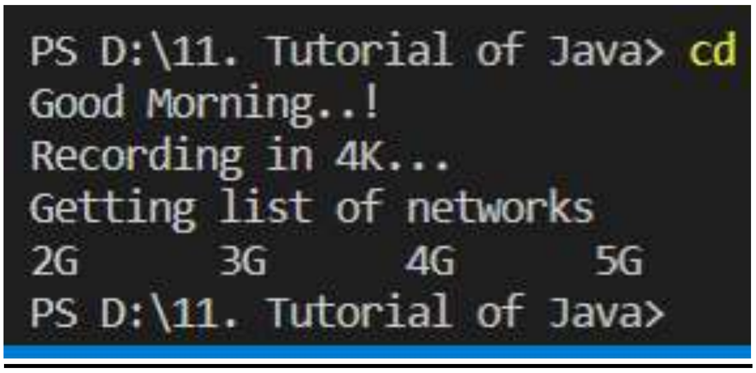
```
PS D:\11. Tutorial of Java> cd
Pee Pee Poo Poo
Applying Brake
Accelrated
45
Pee pee
Poo Poo
PS D:\11. Tutorial of Java>
```

Default Method :

```java
interface Camera{

    void takeSnap();

    void recordVideo();

    private void greet(){ // We may use this private method, if default method contain long

    // logics. We cann't use this private method in implemented class or in main body.

        System.out.println("Good Morning..!");

    }

    /*

    After writing whole code if we add a method in an interface then this will give error

    as whole code getting disturbed because implemented class doesn't have late created

    method, for this we use 'default method'.

    */

    // void record4KVideo(); // --> This will throw error

    default void record4KVideo(){ // We can override this method in implimented class.

        greet(); // We can implement it here only, as it is a private method.

        System.out.println("Recording in 4K...");

    }

}


interface Wifi{

    String [] getNetworks();

    void connectToNetworks(String network);

}


class cellPhone{

    void callNumber(int phoneNumber){

        System.out.println("Calling " + phoneNumber);

    }

    void pickCall(){

        System.out.println("Connecting...");

    }

}
```

```java
class smartPhone extends cellPhone implements Camera, Wifi{

    /*@Override

    public void record4KVideo(){ // This will override the default method.

        System.out.println("Taking snap and recordin in 4K...");

    }

    */

    @Override

    public String[] getNetworks(){

        System.out.println("Getting list of networks");

        String [] networkList = {"2G", "3G", "4G", "5G"};

        return networkList;

    }

    @Override

    public void connectToNetworks(String network){

        System.out.println("Connecting to " + network);

    }

    @Override

    public void takeSnap(){

        System.out.println("Taking a snap");

    }

    @Override

    public void recordVideo(){

        System.out.println("Recording video");

    }

}

public class Y_DefaultMethod{

    public static void main(String[] args) {

        smartPhone ms = new smartPhone();

        ms.record4KVideo();

        String [] arr = ms.getNetworks();

        for(String item : arr){

            System.out.print(item + "\t");

        }

    }

}
```

Output :-



```
PS D:\11. Tutorial of Java> cd
Good Morning..!
Recording in 4K...
Getting list of networks
2G      3G      4G      5G
PS D:\11. Tutorial of Java>
```

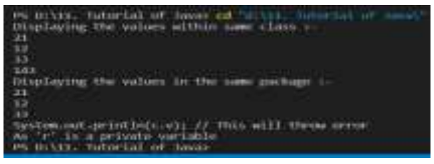
Polymorphism In Interface :

```java
interface Camera{
    void takeSnap();
    void recordVideo();
    private void greet(){
        System.out.println("Good Morning..!");
    }
    default void record4KVideo(){
        greet();
        System.out.println("Recording in 4K...");
    }
}
interface Wifi{
    String [] getNetworks();
    void connectToNetworks(String network);
}
class cellPhone{
    void callNumber(Long phoneNumber){
        System.out.println("Calling " + phoneNumber);
    }
    void pickCall(){
        System.out.println("Connecting...");
    }
}
class smartPhone extends cellPhone implements Camera, Wifi{
    @Override
    public String[] getNetworks(){
        System.out.println("Getting list of networks");
        String [] networkList = {"2G", "3G", "4G", "5G"};
        return networkList;
    }
    @Override
    public void connectToNetworks(String network){
        System.out.println("Connecting to " + network);
    }
    @Override
    public void takeSnap(){
        System.out.println("Taking a snap");
    }
    @Override
    public void recordVideo(){
        System.out.println("Recording video");
    }

}
public class ZA_PolymorphismInInterfaces{
    public static void main(String[] args) {
        Camera MyCM = new smartPhone();
        // MyCM.getNetworks(); // --> Not allowed as object 'MyCM' is for Camera only.
        // In short 'MyCM' object is of SmartPhone but, use it as a camera.
        System.out.println("Object of smartPhone but reference of Camera\n");
        MyCM.record4KVideo();
        MyCM.takeSnap();

        System.out.println("\nObject and reference of smartPhone\n");
        smartPhone sm = new smartPhone();
        sm.getNetworks();
        sm.connectToNetworks("5G");
        sm.takeSnap();
        sm.recordVideo();
        sm.record4KVideo();
        sm.callNumber(6306805527L);
        sm.pickCall();

        cellPhone cp = new smartPhone();
        System.out.println("\nObject of smartPhone but reference of cellPhone\n");
        cp.callNumber(7307871881L);
        cp.pickCall();
    }
}
```

Access Modifier :

```java
class C1{

    public int a = 21;

    protected int n = 12;

    int v = 33;

    private int r = 143;


    public void display(){

        System.out.println(a);

        System.out.println(n);

        System.out.println(v);

        System.out.println(r);

    }

}

public class ZC_AccessModifier{

    public static void main(String[] args){

        C1 c = new C1();

        System.out.println("Displaying the values within same class :-");

        c.display();


        System.out.println("Displaying the values in the same package :-");

        System.out.println(c.a);

        System.out.println(c.n);

        System.out.println(c.v);

        // System.out.println(c.r); // --> This wil throw error as 'r' is a private variable

        System.out.println("System.out.println(c.v); // This will throw error");

        System.out.println("As 'r' is a private variable");


    }

}
```

Output :-



```
PS D:\13. Tutorial of Java> cd "d:\13. Tutorial of Java\"
Displaying the values within same class :-
21
12
33
143
Displaying the values in the same package :-
21
12
33
System.out.println(c.v); // This will throw error
As 'r' is a private variable
PS D:\13. Tutorial of Java>
```