```
Interface
=========
program 1
// weaker privileges
interface itf1   // abstract
{
    void m1();    //public abstract
    void m2();
    void m3();
}
class Test implements itf1
{

    void m1()
    {
        System.out.println("m1 method ");
    }
    void m2()
    {
        System.out.println("m2 method ");
    }
    void m3()
    {
        System.out.println("m2 method ");
    }
    public static void main(String [] args )
    {
        Test t = new Test();
        t.m1();
        t.m2();
        t.m3();
    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
interface.java:17: error: m3() in Test cannot implement m3() i
        void m3()
             ^
  attempting to assign weaker access privileges; was public
```

```
interface.java:13: error: m2() in Test cannot implement m2() i
        void m2()
             ^
  attempting to assign weaker access privileges; was public
interface.java:9: error: m1() in Test cannot implement m1() in
        void m1()
             ^
  attempting to assign weaker access privileges; was public
3 errors
==============================
program 2
//  declare all overriding method as public
interface itf1   // abstract
{
    void m1();    //public abstract
    void m2();
    void m3();
}
class Test implements itf1
```

```java
{
    public void m1()
    {
        System.out.println("m1 method ");
    }
    public void m2()
    {
        System.out.println("m2 method ");
    }
    public void m3()
    {
        System.out.println("m2 method ");
    }
    public static void main(String [] args )
    {
        Test t = new Test();
        t.m1();
        t.m2();
        t.m3();
    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test
m1 method
m2 method
m2 method
=======================================
program 3:
// multiple classes implementing overriden methods
interface itf1   // abstract
{
    void m1();    //public abstract
    void m2();
    void m3();
}
abstract class Test implements itf1
{
    public void m1()
    {
        System.out.println("m1 method ");
    }
    public void m2()
    {
        System.out.println("m2 method ");
    }

}
class Test1 extends Test
{
    public void m3()
    {
        System.out.println("m2 method ");
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();
        t.m2();
        t.m3();
    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
m1 method
m2 method
m2 method
=====================================
program 4:
// accesssing through interface reference variable
interface itf1   // abstract
{
    void m1();    //public abstract
    void m2();
    void m3();
}
abstract class Test implements itf1
{
    public void m1()
    {
        System.out.println("m1 method ");
    }
    public void m2()
    {
        System.out.println("m2 method ");
    }

}
class Test1 extends Test
{
    public void m3()
    {
        System.out.println("m2 method ");
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();
        t.m2();
        t.m3();

        itf1 i = new Test1();  // accesssing through interface
        i.m1();
        i.m2();
        i.m3();
    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
m1 method
m2 method
m2 method
m1 method
m2 method
m2 method
=======================================
program 5
// Adapter class
```

```java
interface itf1   // abstract
{
    void m1();    //public abstract
    void m2();
    void m3();
}
class Demo implements itf1    // Adampter class
{
    public void m1(){}
    public void m2(){}
    public void m3(){}
}
class Test1 extends Demo
{
    public void m1()
    {
        System.out.println("m1 method ");
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();
        t.m2();
        t.m3();


    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
m1 method
=========================================
program 6
// nested interface
interface itf1          // nested interface
{
    interface itf2
    {
        void m1();
    }
}

class Test1 implements itf1.itf2
{
    public void m1()
    {
        System.out.println("m1 method ");
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
m1 method
==========================================
program 7
// interface inside a class
class Demo
{
    interface itf2
    {
        void m1();
    }
}

class Test1 implements Demo.itf2
{
    public void m1()
    {
        System.out.println("m1() method ");
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
m1() method

================================
program 8
// final variables
interface itf1
{
    int x =100;   // public static final
    void m1(); // public abstract
}

class Test1 implements itf1
{
    public void m1()
    {
        x = x+200;
        System.out.println(x);
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}

D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
interface.java:11: error: cannot assign a value to final varia
                x = x+200;
                ^
1 error

========================================
program 9
// ambigous error
interface itf1
{
    int x =100;   // public static final

}
interface itf2
{
    int x = 200;
}

class Test1 implements itf1, itf2
{
    public void m1()
    {
        System.out.println(x);  // ambiguous
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}

D:\AY 2023-24\SEM-I\JAVA\DIVB>javac interface.java
interface.java:15: error: reference to x is ambiguous
                System.out.println(x);
                                   ^
  both variable x in itf1 and variable x in itf2 match
1 error

===============================================
program 10
//
interface itf1
{
    int x =100;   // public static final

}
interface itf2
{
    int x = 200;
}

class Test1 implements itf1, itf2
{
    public void m1()
    {
        System.out.println(itf1.x);
        System.out.println(itf2.x);
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
100
200
========================================
program 11
// calling variables with corresponding interfaces to avoid am
interface itf1
{
    int x =100;   // public static final

}
interface itf2
{
    int x = 200;
}

```

```
class Test1 implements itf1, itf2
{
    public void m1()
    {
        System.out.println(itf1.x);
        System.out.println(itf2.x);
    }
    public static void main(String [] args )
    {
        Test1 t = new Test1();
        t.m1();

    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Test1
100
200

========================================
program 12
// cloing
class Demo implements Cloneable
{
    int x = 100;
    int y =200;

    public static void main(String [] args ) throws Exception
    {
        Demo d = new Demo();
        System.out.println(d.x);
        System.out.println(d.y);

        d.x = 500;
        d.y=600;
        System.out.println(d.x);
        System.out.println(d.y);
        Demo d1= (Demo)d.clone();    // cloning /duplicating o
        ;;;;
        ;;;;
        ;;;;;
        ;;;;
        ;;;;
        ;;;
        ;;;;
        d.x = 700;
        d.y=800;
        System.out.println(d.x);
        System.out.println(d.y);
        // i want to print 500 and 600
        System.out.println(d1.x);
        System.out.println(d1.y);


    }
}
D:\AY 2023-24\SEM-I\JAVA\DIVB>java Demo
100
```

```
200
500
600
700
800
500
600

========================================
Marker Interface

D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.io.Serializable
Compiled from "Serializable.java"
public interface java.io.Serializable {
}

D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.lang.Cloneable
Compiled from "Cloneable.java"
public interface java.lang.Cloneable {
}

D:\AY 2023-24\SEM-I\JAVA\DIVB>javap java.util.RandomAccess
Compiled from "RandomAccess.java"
public interface java.util.RandomAccess {
}
```