LAB SHEET 06 : INHERITANCE & ABSTRACT CLASSES – ANSWERS

Exercise 01:

```
public interface MyFirstInterface
{
   int x = 10;   // Declaration of integer type variable x

   void display();   // Declaration of abstract method display()
}
```

1.

- ✓ Declare the variable with public static final keywords

```
public interface MyFirstInterface
{

// Declaration of integer type variable x with public static final keywords
 public static final int x = 10;

 // Declaration of abstract method display()
 void display();
}
```

- ✓ Declare the variable without public static final keywords

```
public interface MyFirstInterface
{

// Declaration of integer type variable x
   int x = 10;


// Declaration of abstract method display()
   void display();
}
```

- ✓ Difference

  There is no difference between these two approaches.

- ✓ Why

  In Java, all variables declared within an interface are implicitly considered as public static final (constants), whether you explicitly specify these keywords or not.

2.

- ✓ Declare the abstract method with abstract keyword

```
public interface MyFirstInterface
{

// Declaration of integer type variable x
   int x = 10;

// Declaration of abstract method display() with the abstract keyword
   abstract void display();
}
```

- ✓ Declare the abstract method without abstract keyword

```
public interface MyFirstInterface
 {

// Declaration of integer type variable x
   int x = 10;

// Declaration of abstract method display()
   void display();
}
```

- ✓ Difference

  There is no difference between these two approaches.

- ✓ Why

  When defining methods in an interface, all methods are implicitly considered abstract. You do not need to use the abstract keyword explicitly.

3.

```
public class InterfaceImplemented implements MyFirstInterface
{
   @Override
   public void display()
{


// Trying to change the value of x inside this method
       x = 20;

       System.out.println("Value of x: " + x);
   }
}
```

- ✓ Changing

  Not Possible

- ✓ Why

  In this implementation, you'll encounter a compilation error. The reason is that variables declared within an interface are implicitly considered as constants (public static final). Constants cannot be modified once they are assigned a value. So, attempting to change the value of x inside the display() method will result in a compilation error.

  To fix this, you should not attempt to change the value of x inside the implementing class. If you need to modify the value of x, you should do it outside the class, preferably in the interface itself. However, for good programming practices, it's better to avoid modifying constants at runtime.

Exercise 02:

```
interface Speaker
 {
public void speak( );
// Constant/Final Variable Declaration
// Methods Declaration – no method body
}

// Implementing Interfaces
class Politician implements Speaker
{
public void speak()
{
System.out.println(“Talk politics”);
```

```
}
}

class Priest implements Speaker
{
public void speak()
{
System.out.println("Religious Talks");
}
}

class Lecturer implements Speaker
{
public void speak()
{
System.out.println("Talks Object Oriented Design and
Programming!");
}
}
```