RNS Rajapakshe   27058

Practical 05

1)

a)
```
public interface MyFirstInterface
 {
   int x = 10;

   void display();
}
```

b)
```
public interface MyFirstInterface {
   int x = 10; // Equivalent to "public static final int x = 10;"
}
```

c)
```
public class IfImplemented implements MyFirstInterface {
   @Override
   public void display() {
      x = 20; // This will result in a compilation error
      System.out.println("Value of x inside display(): " + x);
   }
}
```

If a class implements an interface, it must provide explicit (non-contextual) implementations for all abstractions declared in the interface. By default, the InterfaceImplemented class implements MyFirstInterface.
But the variable x is stored (always), its value cannot be changed in the implementation class. Trying to change the value of x in the display() method in the above code results in a compile error. Example message "Cannot assign a value to final type 'x'. This is because interface variables are immutable and cannot change their values after initialization."
In Java interfaces, variables can be used as constants (static and final), and their values are set at compile time. So any attempt to change its value is not allowed in the app section.