# ADVANCED PROGRAMMING CONCEPTS USING JAVA

# (CSX-331)

# ASSIGNMENT-1
# COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DR. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY

JALANDHAR – 144011, PUNJAB (INDIA)

AUGUST-DECEMBER, 2017

SUBMITTED To:                                    SUBMITTED BY:

PARAMVIR SINGH                                    Arshdeep Kaur

Associate Professor                               15103025

Department of CSE                                 G-1,Sem-5

## How to create Immutable class in Java?

Immutable class means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like String, Boolean, Byte, Short) and String class is immutable. We can create our own immutable class as well.

Following are the requirements:
• Class must be declared as final (So that child classes can't be created)
• Data members in the class must be declared as final (So that we can't change the value of it after object creation)
• A parameterized constructor
• Getter method for all the variables in it
• No setters(To not have option to change the value of the instance variable)

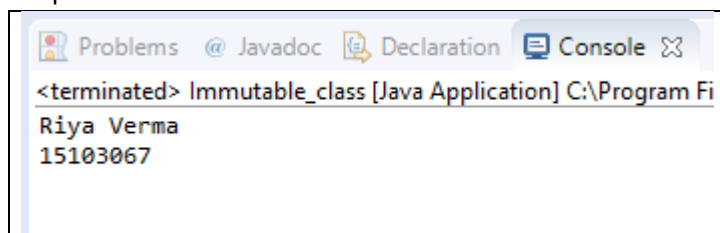### Example to create Immutable class

```java
// An immutable class
public final class Student
{
    final String name;
    final int regNo;

    public Student(String name, int regNo)
    {
        this.name = name;
        this.regNo = regNo;
    }
    public String getName()
    {
        return name;
    }
    public int getRegNo()
    {
        return regNo;
    }
}

// Driver class
class Test
{
    public static void main(String args[])
    {
        Student s = new Student("Riya Verma", 15103067);
        System.out.println(s.name);
        System.out.println(s.regNo);

        // Uncommenting below line causes error
        // s.regNo = 102;
    }
}
```
Output:

**Advantages of Immutable objects over normal objects.**

1. Immutable objects are thread-safe so you will not have any synchronization issues.
2. Immutable objects are good **Map** keys and **Set** elements, since these typically do not change once created.
3. Immutability makes it easier to write, use and reason about the code (class invariant is established once and then unchanged)
4. Immutability makes it easier to parallelize your program as there are no conflicts among objects.
5. The internal state of your program will be consistent even if you have exceptions.
6. References to immutable objects can be cached as they are not going to change.

## Java JTree

The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

DefaultMutableTreeNode class to represent our node. This class has a handy add() method which takes in an instance of MutableTreeNode.

## Declaration

**public class** JTree **extends** JComponent **implements** Scrollable, Accessible
Code:
```
package net.codejava.swing;
import javax.swing.JFrame;
import javax.swing.JTree;
import javax.swing.SwingUtilities;
import javax.swing.tree.DefaultMutableTreeNode;
public class TreeExample extends JFrame
{
    private JTree tree;
    public TreeExample()
    {
        //create the root node
        DefaultMutableTreeNode root = new DefaultMutableTreeNode("Root");
        //create the child nodes
        DefaultMutableTreeNode vegetableNode = new
DefaultMutableTreeNode("Vegetables");
        vegetableNode.add(new DefaultMutableTreeNode("Capsicum"));

        vegetableNode.add(new DefaultMutableTreeNode("Carrot"));
        vegetableNode.add(new DefaultMutableTreeNode("Tomato"));
        vegetableNode.add(new DefaultMutableTreeNode("Potato"));

        DefaultMutableTreeNode fruitNode = new
DefaultMutableTreeNode("Fruits");
        fruitNode.add(new DefaultMutableTreeNode("Mango"));

        fruitNode.add(new DefaultMutableTreeNode("Apple"));
        fruitNode.add(new DefaultMutableTreeNode("Grapes"));

        //add the child nodes to the root node
        root.add(vegetableNode);
        root.add(fruitNode);
```
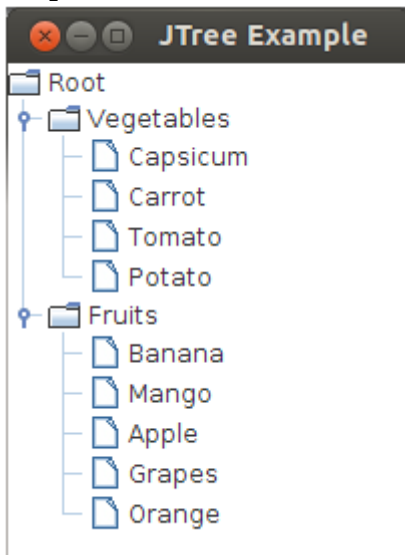
```
        //create the tree by passing in the root node
        tree = new JTree(root);
        add(tree);

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("JTree Example");
        this.pack();
        this.setVisible(true);
    }

    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new TreeExample();
            }
        });
    }
}
```

**Output:**



## Java JTable

The JTable class is used to display data in tabular form. It is composed of rows and columns.

### JTable class declaration

Let's see the declaration for javax.swing.JTable class.

### Commonly used Constructors:

| Constructor | Description |
|---|---|
| JTable() | Creates a table with empty cells. |

| JTable(Object[][] rows, Object[] columns) | Creates a table with the specified data. |
| --- | --- |
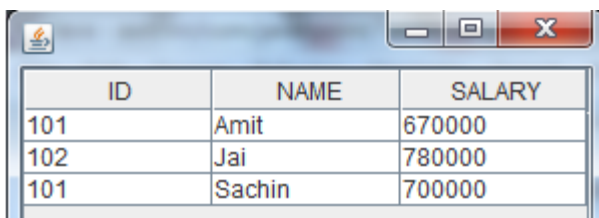
**Java JTable Example**

```java
import javax.swing.*;
public class TableExample {
    JFrame f;
    TableExample(){
    f=new JFrame();
    String data[][]={ {"101","Amit","670000"},
                {"102","Jai","780000"},
                {"101","Sachin","700000"}};
    String column[]={"ID","NAME","SALARY"};
    JTable jt=new JTable(data,column);
    jt.setBounds(30,40,200,300);
    JScrollPane sp=new JScrollPane(jt);
    f.add(sp);
    f.setSize(300,400);
    f.setVisible(true);
}
public static void main(String[] args) {
    new TableExample();
}
}
```

**Output:**



**Usage of mutability in JTree**

A mutable tree node is a node that can mutate ,change. When connected to a JTree and a GUI interface the tree can be dynamically altered. The simples way to use Mutable Tree Node is to use the classes Mutable Tree Node.