1) Programs on class and objects

Create a Class:

To create a class, use the keyword `class`:

Class name.java

//Create a class named "`Main`" with a variable x:

```
public class Main {

int x = 5;

}
```

//Create an object called "`myObj`" and print the value of x:

```
public class Main {

int x = 5;

 public static void main(String[] args) {

 Main myObj = new Main();

    System.out.println(myObj.x);

   }

}
```

```
/*
  Write a Java program to create a new Box class in Java.
*/
class Box {
double width;
double height;
double depth;
}
class BoxDemo {
public static void main(String[] args) {
Box myBox=new Box();
double vol;
myBox.width=10;
myBox.height=200;
myBox.depth=15;
vol=myBox.width*myBox.height*myBox.depth;
System.out.println("Volum is"+vol);
}
}
```

OutPut:

```
C:\Users\IGOI_1\Desktop>java BoxDemo
Volum is30000.0
```

2) Program on Packages

```
package letmecalculate;

public class Calculator {
   public int add(int a, int b){
       return a+b;
   }
   public static void main(String args[]){
       Calculator obj = new Calculator();
       System.out.println(obj.add(10, 20));
   }
}
```

Java Packages & API

A package in Java is used to group related classes. Think of it as **a folder in a file directory**. We use packages to avoid name conflicts, and to write a better maintainable code.

Packages are divided into two categories:

- Built-in Packages (packages from the Java API)
- User-defined Packages (create your own packages)

Built-in Packages

Syntax

```
import package.name.Class;   // Import a single class

import package.name.*;   // Import the whole package
```

// Using the Scanner class to get user input:

```java
import java.util.Scanner; // import the Scanner class

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);
    String userName;

    // Enter username and press Enter
    System.out.println("Enter username");
    userName = myObj.nextLine();

    System.out.println("Username is: " + userName);
  }
}
```

OutPut :

```
Enter username
 Username is: Dhruva
```

User-defined Packages

To create your own package, you need to understand that Java uses a file system directory to store them. Just like folders on your computer:

Example

```
└── root

        └── mypack

            └── MyPackageClass.java
```

```java
package mypack;

class MyPackageClass {
  public static void main(String[] args) {
    System.out.println("This is my package!");
  }
}
```

O/P :

```
This is my package!
```

2)

```
 package MyPackage;
public class Compare {
  int num1, num2;
  Compare(int n, int m) {
     num1 = n;
     num2 = m;
}
public void getmax(){
    if ( num1 > num2 ) {
        System.out.println("Maximum value of two numbers is " + num1);
  }
    else {
        System.out.println("Maximum value of two numbers is " + num2);
    }
}
public static void main(String args[]) {
        Compare current[] = new Compare[3];
        current[1] = new Compare(5, 10);
        current[2] = new Compare(123, 120);
          for(int i=1; i < 3 ; i++)
             {
             current[i].getmax();
             }
       }
}
```

Output:

```
1
2
Maximum value of two numbers is 10
Maximum value of two numbers is 123
```

3) Program on 2D array, strings functions

```java
import java.util.Scanner;

public class TwoDArray
{
   public static void main(String[] args)
   {
      int row, col, i, j;
      Scanner scan = new Scanner(System.in);

      System.out.print("Enter Row size: ");
      row = scan.nextInt();
      System.out.print("Enter column Size: ");
      col = scan.nextInt();

      int[][] arr = new int[row][col];

      System.out.print("\nEnter " +(row*col)+ " Elements: ");
      for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
            arr[i][j] = scan.nextInt();
      }

      System.out.println("\nArray's Elements with its indexes: ");
      for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
            System.out.print("arr["+i+"]["+j+"] = " +arr[i][j]+"\t");
         System.out.print("\n");
      }
   }
}
```

Out Put:

```
C:\Users\IGOI_1\Desktop>javac TwoDArray.java


C:\Users\IGOI_1\Desktop>java TwoDArray

Enter Row size: 4

Enter column Size: 3


Enter 12 Elements: 1

2

3

4

5

6
```

```
7

8

9

1

2

3


Array's Elements with its indexes:

arr[0][0] = 1   arr[0][1] = 2   arr[0][2] = 3

arr[1][0] = 4   arr[1][1] = 5   arr[1][2] = 6

arr[2][0] = 7   arr[2][1] = 8   arr[2][2] = 9

arr[3][0] = 1   arr[3][1] = 2   arr[3][2] = 3


C:\Users\IGOI_1\Desktop>
```

4) Program on String Buffer and Vectors

Java StringBuffer Class with Example

a) Code to understand the length() method of StringBuffer class:

It defines the number of characters in the String.

```
import java.io. * ;
class Main {
  public static void main(String[] args) {
    StringBuffer str = new StringBuffer("ContentWriter");
    int len = str.length();
    System.out.println("Length : " + len);
  }
}
```

Output:

```
Length: 13
```

2. StringBuffer: capacity()

It defines the capacity of the string occupied in the buffer. The capacity method also counts the reserved space occupied by the string.

Code to understand the capacity() method of StringBuffer class:

```
import java.io. * ;

class Main {

  public static void main(String[] args) {

    StringBuffer str = new StringBuffer("ContentWriter");

    int cap = str.capacity();

    System.out.println("Capacity : " + cap);

  }

}
```

Output:

```
Capacity: 29
```

3. StringBuffer: append()

The append() method is used to append the string or number at the end of the string.

Code to understand the append() method of StringBuffer class:

```
import java.io.*;
class Main
{
public static void main(String[] args)
{
StringBuffer str = new StringBuffer("ICOE");
str.append("KALYAN"); // appends a string in the previously defined string.
System.out.println(str);
str.append(0); // appends a number in the previously defined string.
System.out.println(str);
}
}
```

Output:

```
ICOEKALYANICOEKALYAN0
```

4. StringBuffer: insert()

The insert() method is used to insert the string into the previously defined string at a particular indexed position.

In the insert method as a parameter, we provide two-argument first is the index and second is the string.

Code to understand insert() method of StringBuffer class:

```
public class StringBufferInsert {
  public static void main(String[] args) {
    StringBuffer stringName = new StringBuffer(" Welcome");
    System.out.println(stringName);
    stringName.insert(8, " to ");
    System.out.println(stringName);
    stringName.insert(12, "ICOE ");
    System.out.println(stringName);
    stringName.insert(22, " Tutorial ");
    System.out.println(stringName);
    stringName.insert(31, " of ");
    System.out.println(stringName);
    stringName.insert(35, "Java");
    System.out.println(stringName);
  }
}
```

Output:

Advertisement

```
Welcome
Welcome to
Welcome to ICOE
Welcome to ICOE Tutorial
Welcome to ICOE Tutorial of
Welcome to ICOE Tutorial of Java
```

5. StringBuffer: reverse()

The reverse() method reverses all the characters of the object of the StringBuffer class. And, as an output, this method returns or gives the reversed String object.

Code to understand reverse() method of StringBuffer class:

```
import java.util. * ;

public class StringBufferReverse {

  public static void main(String[] args) {

    StringBuffer stringName = new StringBuffer("Welcome to INDALA");

    System.out.println("Original String: " + stringName);

    stringName.reverse();

    System.out.println("Reversed String: " + stringName);

  }

}
```

Output:

```
Original String: Welcome to INDALA

Reversed String: ALADNI ot emocleW
```

6. StringBuffer: delete()

Let’s move to the next method of the StringBuffer class which is the delete() method. The delete() method deletes a sequence of characters from the StringBuffer object.

We provide two arguments to this method in which the first argument is the starting index of the String which we want to delete and the second index the last index of the String up to which we want to delete.

Thus, with this method, we can delete or remove a sequence of characters from the string. The output is the characters. String after removing the specified characters.

Syntax:

stringName.delete(int startIndex, int endIndex)

Code to understand delete() method of StringBuffer class:

```
import java.io. * ;

public class DeleteMethodDemo {

  public static void main(String[] args) {

    StringBuffer myString = new StringBuffer("IndalaCollege");

    System.out.println("Original String: " + myString);

    myString.delete(0, 4);

    System.out.println("Resulting String after deleting sequence of characters: " + myString);

  }

}
```

Output:

```
Original String: IndalaCollegeResulting

String after deleting sequence of characters: laCollege
```

7. StringBuffer deleteCharAt()

The deleteCharAt() method deletes only the character at the specified index.

We provide the index number as the argument of the method and it deletes the character present at that index and returns the resulting string after deleting that specific character.

Syntax:

stringName.deleteCharAt(int index)

Code to understand deleteCharAt() method of StringBuffer class:

```
import java.io. * ;

public class DeleteCharAtDemo {

  public static void main(String[] args) {

    StringBuffer myString = new StringBuffer("IndalaCollege");

    System.out.println("Original String: " + myString);


    myString.deleteCharAt(0);

    System.out.println("Resulting String after deleting a character at 0th index: " + myString);


    myString.deleteCharAt(6);

    System.out.println("Resulting String after deleting a character at 6th index: " + myString);

  }

}
```

Output:

```
Original String: IndalaCollege

Resulting String after deleting a character at 0th index: ndalaCollege

Resulting String after deleting a character at 6th index: ndalaCllege
```

8. StringBuffer replace()

The replace() method of the StringBuffer class replaces a sequence of characters with another sequence of characters.

It takes three arguments, one is the starting index and the second is the last index, while the last argument is the sequence of characters which we want to replace with the specified characters.

Syntax:

stringName.replace(int startIndex, int endIndex, String string)

Code to understand replace() method of StringBuffer class:

```
import java.io. * ;

public class ReplaceMethodDemo {

  public static void main(String[] args) {

    StringBuffer s = new StringBuffer("IndalaCollege");

    System.out.println("Original String: " + s);

    s.replace(10, 14, "Tutorial");

    System.out.println("Resulting String after replacing: " + s);

  }

}
```

Output:

```
Original String: IndalaCollege

Resulting String after replacing: IndalaCollTutorial
```

5) Program on types of inheritance

```
class PetAnimal {
  // field and method of the parent class
  String name;
  public void eat() {
    System.out.println("I can eat");
  }
}
// inherit from PetAnimal
class Dog extends PetAnimal {
  // new method in subclass
  public void display() {
    System.out.println("My name is " + name);
  }
}
class Main {
  public static void main(String[] args) {
    // create an object of the subclass
    Dog labrador = new Dog();
    // access field of superclass
    labrador.name = "Rohu";
    labrador.display();
    // call method of superclass
    // using object of subclass
    labrador.eat();
  }
}
```

```
C:\Users\IGOI_1\Desktop>javac Main.java
C:\Users\IGOI_1\Desktop>java Main
My name is Rohu
I can eat
```

2)

format for Inheritance

```
class superclass

{
 // superclass data variables
 // superclass member functions
}
class subclass extends superclass
{
 // subclass data variables
 // subclass member functions
}
```

A program that demonstrates multiple inheritance by interface in Java is given as follows:

5) Program on Multiple Inheritance

```
interface AnimalEat {
  void eat();
}
interface AnimalTravel {
  void travel();
}
class Animal implements AnimalEat, AnimalTravel {
  public void eat() {
    System.out.println("Animal is eating");
  }
  public void travel() {
    System.out.println("Animal is travelling");
  }
}
public class Demo {
  public static void main(String args[]) {
    Animal a = new Animal();
    a.eat();
    a.travel();
  }
}
```

Output

```
Animal is eating
Animal is travelling
```

6) Program on abstract class and abstract methods.

```
// abstract class
abstract class Multiply {
   // abstract methods
   // sub class must implement these methods
   public abstract int MultiplyTwo (int n1, int n2);
   public abstract int MultiplyThree (int n1, int n2, int n3);
   // regular method with body
   public void show() {
    System.out.println ("Method of abstract class Multiply");
   }
}
// Regular class extends abstract class
class AbstractMethodEx1 extends Multiply {
  // if the abstract methods are not implemented, compiler will give an error
   public int MultiplyTwo (int num1, int num2) {
    return num1 * num2;
   }
   public int MultiplyThree (int num1, int num2, int num3) {
    return num1 * num2 * num3;
   }
   // main method
   public static void main (String args[]) {
    Multiply obj = new AbstractMethodEx1();
    System.out.println ("Multiplication of 2 numbers: " + obj.MultiplyTwo (10, 50));
    System.out.println ("Multiplication of 3 numbers: " + obj.MultiplyThree (5, 8, 10));
    obj.show();
   }
}
```

Out Put:

```
C:\Users\IGOI_1\Desktop>java AbstractMethodEx1

Multiplication of 2 numbers: 500

Multiplication of 3 numbers: 400

Method of abstract class Multiply


C:\Users\IGOI_1\Desktop>
```

7) Program using super and final keyword

The super keyword in java is a reference variable that is used to refer parent class objects. The keyword “super” came into the picture with the concept of Inheritance. It is majorly used in the following contexts:

- Use of super with variables
- Use of super with methods
- Use of super with constructors

```
// superclass Person

class Person

{

    int id=111;

    void message()

   {

     System.out.println("Welcome to INDALA!");

   }

   Person()

   {

      System.out.println("Person class Constructor");

   }

}

// subclass Student extending the Person class

class Student extends Person // Inheritance

{

   Student()

   {

   super(); // invoke or call parent class constructor

   System.out.println("Student class Constructor");

   }

   void message()

   {

    System.out.println("Technologies");

    }

    void display()

    {

   super.message(); // calling super class method
message();
System.out.println("Student Id: "+super.id); //accessing super class variable
  }
}
class Test
{
  public static void main(String[] args)
  {
    Student s = new Student();
    s.display();
  }
}
```

```
C:\Users\IGOI_1\Desktop>javac Test.java


C:\Users\IGOI_1\Desktop>java Test
Person class Constructor
Student class Constructor
Welcome to INDALA!
Technologies
Student Id: 111
C:\Users\IGOI_1\Desktop>
```

Final Keyword in JAVA

final keyword is used in different contexts. First of all, final is a non-access modifier applicable only to a variable, a method or a class.Following are different contexts where final is used.

- **final variables -** to create constant variables
- **final methods -** prevent method overriding
- **final classes -** prevent inheritance

Program Code

```
class Doctor
{
 public final void message()       // cannot be overridden by subclass
 {
   System.out.println("Welcome here!!!");
 }
}
final class Patient
{
 public void fun()
 {
System.out.println("Do fun!!");
 }
}
public class Hospital extends Doctor //cannot extend final class Patient
{
 public static void main(String args[])
 {
   final int x=100; // final variable
   Hospital h1 = new Hospital();
   h1.message();
   // x=200;  Compilation error
   Patient p1=new Patient();
   p1.fun();
 }
}
```

Out Put:

```
C:\Users\IGOI_1\Desktop>java Hospital
Welcome here!!!
Do fun!!

C:\Users\IGOI_1\Desktop>
```

8) Program on Exception handling

TryCatchExample1.java

```
public class TryCatchExample1 {

    public static void main(String[] args) {

        int data=50/0; //may throw exception

        System.out.println("rest of the code");

    }

}
```

OutPut

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
//TryCatchExample
public class TryCatchExample6 {

    public static void main(String[] args) {
        int i=50;
        int j=0;
        int data;
        try
        {
        data=i/j; //may throw exception
        }
            // handling the exception
        catch(Exception e)
        {
             // resolving the exception in catch block
            System.out.println(i/(j+2));
        }
    }
}
```

OutPut

```
25
```

Let's see an example to handle checked exception.

TryCatchExample

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class TryCatchExample10 {

    public static void main(String[] args) {


        PrintWriter pw;
        try {
            pw = new PrintWriter("jtp.txt"); //may throw exception
            pw.println("saved");
        }
// providing the checked exception handler
 catch (FileNotFoundException e) {

            System.out.println(e);
        }
    System.out.println("File saved successfully");
    }
}
```

Out PUT

```
File saved successfully
```

MultipleCatchBlock.java

```
public class MultipleCatchBlock3 {

    public static void main(String[] args) {

        try{
            int a[]=new int[5];
            a[5]=30/0;
            System.out.println(a[10]);
           }
           catch(ArithmeticException e)
             {
              System.out.println("Arithmetic Exception occurs");
             }
           catch(ArrayIndexOutOfBoundsException e)
             {
              System.out.println("ArrayIndexOutOfBounds Exception occurs");
             }
           catch(Exception e)
             {
              System.out.println("Parent Exception occurs");
             }
           System.out.println("rest of the code");
    }
}
```

Out Put

```
Arithmetic Exception occurs

rest of the code
```

NestedTryBlock.java

```
public class NestedTryBlock2 {

    public static void main(String args[])
    {
        // outer (main) try block
        try {

            //inner try block 1
            try {

                // inner try block 2
                try {
                    int arr[] = { 1, 2, 3, 4 };

                    //printing the array element out of its bounds
                    System.out.println(arr[10]);
                }

                // to handles ArithmeticException
                catch (ArithmeticException e) {
                    System.out.println("Arithmetic exception");
                    System.out.println(" inner try block 2");
                }
            }

            // to handle ArithmeticException
            catch (ArithmeticException e) {
                System.out.println("Arithmetic exception");
                System.out.println("inner try block 1");
            }
        }

        // to handle ArrayIndexOutOfBoundsException
        catch (ArrayIndexOutOfBoundsException e4) {
            System.out.print(e4);
```

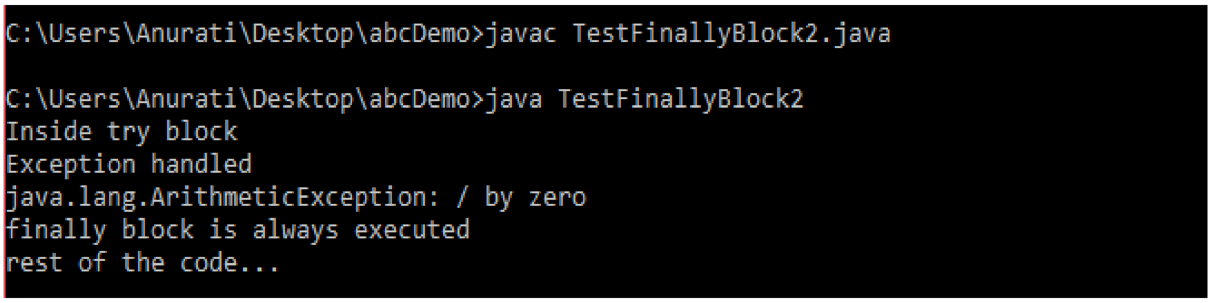
```
                System.out.println(" outer (main) try block");
            }
            catch (Exception e5) {
                System.out.print("Exception");
                System.out.println(" handled in main try-block");
            }
        }
    }
```

OutPut

TestFinallyBlock.java

```java
public class TestFinallyBlock2{
      public static void main(String args[]){

      try {

        System.out.println("Inside try block");

        //below code throws divide by zero exception
       int data=25/0;
       System.out.println(data);
      }

      //handles the Arithmetic Exception / Divide by zero exception
      catch(ArithmeticException e){
        System.out.println("Exception handled");
        System.out.println(e);
      }

      //executes regardless of exception occured or not
      finally {
        System.out.println("finally block is always executed");
      }

      System.out.println("rest of the code...");
      }
    }
```

Output:



```
C:\Users\Anurati\Desktop\abcDemo>javac TestFinallyBlock2.java

C:\Users\Anurati\Desktop\abcDemo>java TestFinallyBlock2
Inside try block
Exception handled
java.lang.ArithmeticException: / by zero
finally block is always executed
rest of the code...
```

```
class InvalidAgeException extends Exception{
 InvalidAgeException(String s){
  super(s);
 }
 public String toString()
 {
  return "Candidate is less than 18 year is not allowed to vote.";
 }
}
class ValidateCandidate{
   static void validate(int age) throws InvalidAgeException{
     if(age < 18)
        throw new InvalidAgeException("invalid candidate");
     else
        System.out.println("welcome to vote");
   }
   public static void main(String args[]){
      try{
        validate(13);
      }catch(Exception ex){
        System.out.println("Exception occured: "+ex);
      }
      System.out.println("rest of the code...");
  }
}
```

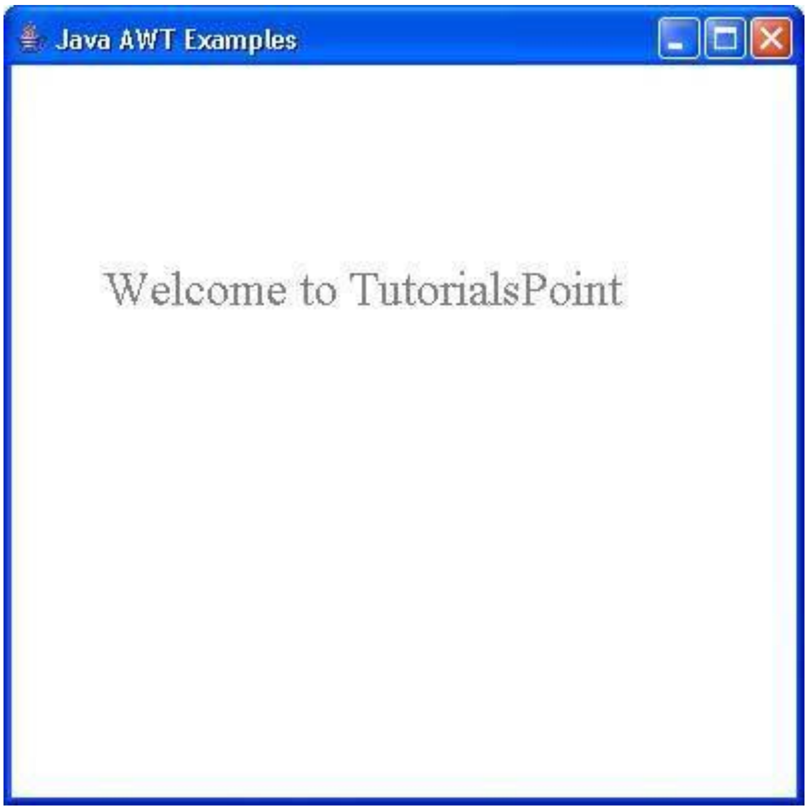
OutPut:

```
C:\Users\IGOI_1\Desktop>java ValidateCandidate
Exception occured: Candidate is less than 18 year is not allowed to vote.
rest of the code...
C:\Users\IGOI_1\Desktop>
```

9) Program on Graphics class

```
public class AWTGraphicsDemo extends Frame {
  public AWTGraphicsDemo(){
    super("Java AWT Examples");
    prepareGUI();
  }
  public static void main(String[] args){
    AWTGraphicsDemo  awtGraphicsDemo = new AWTGraphicsDemo();
    awtGraphicsDemo.setVisible(true);
  }
  private void prepareGUI(){
    setSize(400,400);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent windowEvent){
        System.exit(0);
      }
    });
  }

  @Override
  public void paint(Graphics g) {
    g.setColor(Color.GRAY);
    Font font = new Font("Serif", Font.PLAIN, 24);
    g.setFont(font);
    g.drawString("Welcome to TutorialsPoint", 50, 150);
  }
}
```

output



```
Welcome to TutorialsPoint
```

10) Program on applet class

```
// Applet.java
import java.applet.*;
import java.awt.*;

public class CheckerApplet extends Applet {
  int squareSize = 50;   // initialized to default size
  public void init() {}
  private void parseSquareSize (String param) {}
  private Color parseColor (String param) {}
  public void paint (Graphics g) {}
}
```

OutPut

//Applet's init() and private parseSquareSize() methods –

```
public void init () {
   String squareSizeParam = getParameter ("squareSize");
   parseSquareSize (squareSizeParam);

   String colorParam = getParameter ("color");
   Color fg = parseColor (colorParam);

   setBackground (Color.black);
   setForeground (fg);
}

private void parseSquareSize (String param) {
   if (param == null) return;
   try {
      squareSize = Integer.parseInt (param);
   } catch (Exception e) {
      // Let default value remain
   }
}
```

Out put

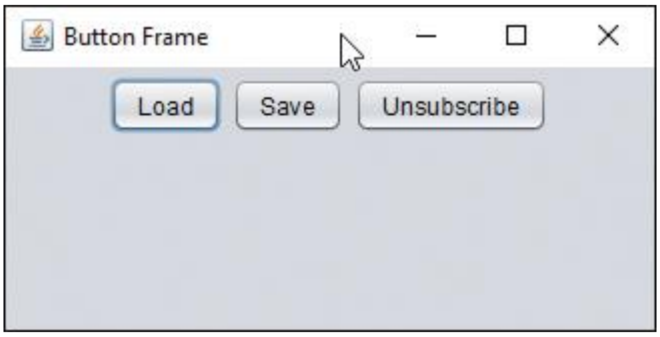
//Specifying Applet Parameters

```
<html>
  <title>Checkerboard Applet</title>
  <hr>
  <applet code = "CheckerApplet.class" width = "480" height = "320">
    <param name = "color" value = "blue">
    <param name = "squaresize" value = "30">
  </applet>
  <hr>
</html>
```

11) Program to create GUI application

```
//The Full Text of ButtonFrame.java
package com.java21days;
 import javax.swing.*;
 public class ButtonFrame extends JFrame {
    JButton load = new JButton("Load");
    JButton save = new JButton("Save");
    JButton unsubscribe = new JButton("Unsubscribe");
    public ButtonFrame() {
        super("Button Frame");
        setSize(340, 170);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel pane = new JPanel();
        pane.add(load);
        pane.add(save);
        pane.add(unsubscribe);
        add(pane);
        setVisible(true);
    }
    private static void setLookAndFeel() {
        try {
            UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
        } catch (Exception exc) {
            System.out.println(exc.getMessage());
        }
    }
    public static void main(String[] arguments) {
        setLookAndFeel();
        ButtonFrame bf = new ButtonFrame();
    }
}
```

OUTPUT


Button Frame
Load
Save
Unsubscribe