



# **Journal**

## BCA Department

### Laboratory Certificate

This is to certify that Smt./Shri \_\_\_\_\_  
has satisfactory completed **BCA Semester-4** practical experiments of subject **CS-19**  
**Programming with Java** during the academic year **2022-23**. Her/His enrollment  
number is \_\_\_\_\_ registered at Saurashtra University, Rajkot.

Date: \_\_\_\_\_

\_\_\_\_\_  
Subject In-Charge

\_\_\_\_\_  
Head of the Department

# Index

Sr.	Name of Experiments	Page No.	Date of Experiment	Date of Supervision	Remarks
	<b>Unit-1</b>				
1	Hello World Program	6			
2	Java Variables	6			
3	Leap Year	6			
4	Find vowels	6			
5	Passing an array to function	7			
6	Class and Objects	7			
7	Class with Method	8			
8	Parameterized constructor	8			
9	Constructor Overloading	8			
10	Jagged Array	9			
11	Copy constructor	9			
12	Java Inheritance	9			
13	Method Overloading	10			
	<b>Unit-2</b>				
14	Constructor in Inheritance	12			
15	Abstract Class	12			
16	Final Class	13			
17	Java Interface	13			
18	Inner Class	14			
19	util.Date class	14			
20	Java Wrapper Classes	14			
21	Creating user defined package	15			
22	Java StringTokenizer	15			
	<b>Unit-3</b>				
23	Exception Handling	17			
24	Multiple catch statements	17			
25	Multithreading using Thread Class	18			
26	Multithreading using Runnable interface	18			
27	Thread Scheduling	18			
28	Thread Joins	19			
29	Thread Priorates	19			
30	File Class	20			
31	Bytestream Class to read file	20			
32	Bytestream Class to create file	20			
33	Character stream Class to read and write file	21			
	<b>Unit-4</b>				
34	HelloWorld Applet	23			

# Index

Sr.	Name of Experiments	Page No.	Date of Experiment	Date of Supervision	Remarks
35	Applet Life Cycle and Mouse Event Listener	23			
36	Applet Graphics	24			
37	Passing Parameter in Applet	24			
38	Image in Applet	25			
39	Border layout	25			
40	Grid layout	25			
	<b>Unit-5</b>				
41	JFrame and JPanel	28			
42	JButton with Event	28			
43	TextField Example	29			
44	CheckBox Example	29			
45	JList Example	30			

## **Unit – 1**

# **History, Introduction and Language Basics, Classes and Objects**

### 1. Hello World Program

```
1 class HelloJava {
2     public static void main(String arg[]) {
3         System.out.println("Hello Java");
4         System.out.print("Java is an OOP");
5     }
6 }
```

### 2. Java Variables

```
1 //Java Variables
2 class VariableDemo {
3     public static void main(String[] arg) {
4         int i=10;
5         String n="Java";
6         float f=5.5f;
7         System.out.println("Value of i: "+i);
8         System.out.println("Value of n: "+n);
9         System.out.println("Value of f: "+f);
10    }
11 }
```

### 3. Leap Year

```
1 //Leap year example using if...else
2
3 public class LeapYearExample {
4     public static void main(String[] args) {
5         int year=2021;
6         if(((year % 4==0) && (year % 100!=0)) || (year % 400==0)){
7             System.out.println("LEAP YEAR");
8         }
9         else{
10            System.out.println("COMMON YEAR");
11        }
12    }
13 }
```

### 4. Find vowels

```
1 //Vowels using switch...case
2
3 public class SwitchExample {
4     public static void main(String[] args) {
5         char ch='L';
6         switch(ch)
7         {
8             case 'a':
9                 System.out.println("Vowel");
10                break;
11             case 'e':
12                 System.out.println("Vowel");
13                break;
14             case 'i':
15                 System.out.println("Vowel");
16                break;
17             case 'o':
18                 System.out.println("Vowel");
19                break;
20             case 'u':
```

```

21         System.out.println("Vowel");
22         break;
23     case 'A':
24         System.out.println("Vowel");
25         break;
26     case 'E':
27         System.out.println("Vowel");
28         break;
29     case 'I':
30         System.out.println("Vowel");
31         break;
32     case 'O':
33         System.out.println("Vowel");
34         break;
35     case 'U':
36         System.out.println("Vowel");
37         break;
38     default:
39         System.out.println("Consonant");
40     }
41 }
42 }

```

## 5. Passing an array to function

```

1  //Java Program to demonstrate the way of passing an array
2
3  class FindMin{
4      static void min(int arr[]){
5          int min=arr[0];
6          for(int i=1;i<arr.length;i++)
7              if(min>arr[i]) min=arr[i];
8          System.out.println(min);
9      }
10     public static void main(String args[]){
11         int a[]={33,3,1,5}; //declaring and initializing an array
12         min(a); //passing array to method
13     }
14 }

```

## 6. Class and Objects

```

1  //Oop Example
2  class Student{
3      int id;
4      String name;
5  }
6
7  class TestStudent{
8      public static void main(String args[]){
9          Student s1=new Student();
10         Student s2=new Student();
11         s1.id=101;
12         s1.name="Ritul";
13         s2.id=102;
14         s2.name="Amit";
15         System.out.println(s1.id+" "+s1.name);
16         System.out.println(s2.id+" "+s2.name);
17     }
18 }

```

## 7. Class with Method

```
1 //Class with method
2
3 class Employee{
4     int id;
5     String name;
6     float salary;
7     void setData(int i, String n, float s) {
8         id=i;
9         name=n;
10        salary=s;
11    }
12    void getData() {
13        System.out.println(id+" "+name+" "+salary);
14    }
15 }
16 public class TestEmployee {
17     public static void main(String[] args) {
18         Employee e1=new Employee();
19         Employee e2=new Employee();
20         e1.setData(101,"Ravi",45000);
21         e2.setData(102,"Mohit",25000);
22         e1.getData();
23         e2.getData();
24     }
25 }
```

## 8. Parameterized constructor

```
1 //Java Program to demonstrate the use of the parameterized constructor.
2
3 class Student4{
4     int id;
5     String name;
6     //creating a parameterized constructor
7     Student4(int i,String n){
8         id = i;
9         name = n;
10    }
11    //method to display the values
12    void display(){
13        System.out.println(id+" "+name);
14    }
15    public static void main(String args[]){
16        //creating objects and passing values
17        Student4 s1 = new Student4(111,"Ritul");
18        Student4 s2 = new Student4(222,"Ravi");
19        //calling method to display the values of object
20        s1.display();
21        s2.display();
22    }
23 }
```

## 9. Constructor Overloading

```
1 //Java program to overload constructors
2 class Student5{
3     int id;
4     String name;
5     int age;
6     //creating two arg constructor
7     Student5(int i,String n){
8         id = i;
9         name = n;
```



```

10     }
11     //creating three arg constructor
12     Student5(int i,String n,int a){
13         id = i;
14         name = n;
15         age=a;
16     }
17     void display(){System.out.println(id+" "+name+" "+age);}
18
19     public static void main(String args[]){
20         Student5 s1 = new Student5(111,"Mohit");
21         Student5 s2 = new Student5(222,"Priyanshu",25);
22         s1.display();
23         s2.display();
24     }
25 }

```

## 10. Jagged Array

```

1 //Program to Jagged Array.
2
3 class Test
4 {
5     public static void main(String[] args)
6     {
7         int[][] arr = new int[2][]; // Declare the array
8
9         arr[0] = new int[] { 11, 21, 56, 78 }; // Initialize the array
10        arr[1] = new int[] { 42, 61, 37, 41, 59, 63 };
11
12        // Traverse array elements
13        for (int i = 0; i < arr.length; i++)
14        {
15            for (int j = 0; j < arr[i].length; j++)
16            {
17                System.out.print(arr[i][j] + " ");
18            }
19            System.out.println();
20        }
21    }
22 }

```

## 11. Copy constructor

```

1 //Copy constructor...
2
3 class Student6{
4     int id;
5     String name;
6     //constructor to initialize integer and string
7     Student6(int i,String n){
8         id = i;
9         name = n;
10    }
11    //constructor to initialize another object
12    Student6(Student6 s){
13        id = s.id;
14        name =s.name;
15    }
16    void display(){System.out.println(id+" "+name);}
17
18    public static void main(String args[]){
19        Student6 s1 = new Student6(111,"Krupa");
20        Student6 s2 = new Student6(s1);
21        s1.display();
22        s2.display();
23    }
24 }

```

## 12. Java Inheritance

```

1 //Java Inheritance Demo
2
3 class Animal{
4     void eat(){
5         System.out.println("eating...");

```

```

6      }
7  }
8  class Dog extends Animal{
9      void bark(){
10         System.out.println("barking...");
11     }
12 }
13 class BabyDog extends Dog{
14     void weep(){
15         System.out.println("weeping...");
16     }
17 }
18 class TestInheritance{
19     public static void main(String args[]){
20         BabyDog d=new BabyDog();
21         d.weep();
22         d.bark();
23         d.eat();
24     }
25 }

```

### 13. Method Overloading

```

1  //Method Overloading Demo...
2  class Adder{
3      static int add(int a, int b) {
4          return a+b;
5      }
6      static double add(double a, double b) {
7          return a+b;
8      }
9  }
10 class TestOverloading{
11     public static void main(String[] args){
12         System.out.println(Adder.add(11,11));
13         System.out.println(Adder.add(12.3,12.6));
14     }
15 }

```

# **Unit – 2**

## **Inheritance, Java Packages**

#### 14. Constructor in Inheritance

```
1 //Constructor in Inheritance
2 class Animal{
3     Animal() {
4         System.out.println("From animal constructor");
5     }
6     void eat(){
7         System.out.println("eating...");
8     }
9     protected void finalize() {
10        System.out.println("End of animal");
11    }
12 }
13 class Dog extends Animal{
14     Dog() {
15         System.out.println("From dog constructor");
16     }
17     void bark(){
18         System.out.println("barking...");
19     }
20     protected void finalize() {
21         System.out.println("End of dog");
22     }
23 }
24 }
25 class BabyDog extends Dog{
26     BabyDog() {
27         System.out.println("From babydog constructor");
28     }
29     void weep(){
30         System.out.println("weeping...");
31     }
32     protected void finalize() {
33         System.out.println("End of babydog");
34     }
35 }
36 }
37 class TestInheritance2{
38     public static void main(String args[]){
39         BabyDog d=new BabyDog();
40         d.weep();
41         d.bark();
42         d.eat();
43         d=null;
44         System.gc();
45     }
46 }
```

#### 15. Abstract Class

```
1 //abstract class demo.
2
3 abstract class Shape{
4     abstract void draw();
5 }
6
7 class Rectangle extends Shape{
8     void draw(){System.out.println("drawing rectangle");}
9 }
10 class Circle extends Shape{
```

```

11     void draw(){System.out.println("drawing circle");}
12 }
13
14 class TestAbstraction{
15     public static void main(String args[]){
16         Shape s1=new Circle();
17         Shape s2=new Rectangle();
18         s1.draw();
19         s2.draw();
20     }
21 }

```

## 16. Final Class

```

1 //Final Class
2
3 final class ParentClass
4 {
5     void showData()
6     {
7         System.out.println("This is a method of final Parent class");
8     }
9 }
10
11 //It will throw compilation error
12 class ChildClass extends ParentClass
13 {
14     void showData()
15     {
16         System.out.println("This is a method of Child class");
17     }
18 }
19 class MainClass
20 {
21     public static void main(String arg[])
22     {
23         ParentClass obj = new ChildClass();
24         obj.showData();
25     }
26 }

```

## 17. Java Interface

```

1 //Interface Demo...
2 interface Animal {
3     public void eat();
4     public void travel();
5 }
6
7 class MammalInt implements Animal {
8
9     public void eat() {
10         System.out.println("Mammal eats");
11     }
12
13     public void travel() {
14         System.out.println("Mammal travels");
15     }
16
17     public int noOfLegs() {
18         return 0;
19     }
20 }
21
22

```

```

23 public class Main {
24     public static void main(String args[]) {
25         MammalInt m = new MammalInt();
26         m.eat();
27         m.travel();
28     }
29 }

```

## 18. Inner Class

```

1 //Inner class demo.
2
3 class Main {
4     private int data=30;
5     class Inner{
6         void msg(){System.out.println("data is "+data);}
7     }
8     public static void main(String args[]){
9         Main obj=new Main();
10        Main.Inner in=obj.new Inner();
11        in.msg();
12    }
13 }

```

## 19. util.Date class

```

1 import java.util.Date;
2
3 public class Main {
4
5     public static void main(String args[]) {
6
7         Date date = new Date();
8
9         System.out.println(date.toString());
10    }
11 }

```

## 20. Java Wrapper Classes

```

1 //wrapper classes objects and vice-versa
2
3 public class Main {
4     public static void main(String args[]){
5         byte b=10;
6         short s=20;
7         int i=30;
8         long l=40;
9         float f=50.0F;
10        double d=60.0D;
11        char c='a';
12        boolean b2=true;
13
14        //Autoboxing: Converting primitives into objects
15        Byte byteobj=b;
16        Short shortobj=s;
17        Integer intobj=i;
18        Long longobj=l;
19        Float floatobj=f;
20        Double doubleobj=d;
21        Character charobj=c;
22        Boolean boolobj=b2;
23
24        //Printing objects
25        System.out.println("---Printing object values---");
26        System.out.println("Byte object: "+byteobj);
27        System.out.println("Short object: "+shortobj);

```

```

28     System.out.println("Integer object: "+intobj);
29     System.out.println("Long object: "+longobj);
30     System.out.println("Float object: "+floatobj);
31     System.out.println("Double object: "+doubleobj);
32     System.out.println("Character object: "+charobj);
33     System.out.println("Boolean object: "+boolobj);
34
35     //Unboxing: Converting Objects to Primitives
36     byte bytevalue=byteobj;
37     short shortvalue=shortobj;
38     int intvalue=intobj;
39     long longvalue=longobj;
40     float floatvalue=floatobj;
41     double doublevalue=doubleobj;
42     char charvalue=charobj;
43     boolean boolvalue=boolobj;
44
45     //Printing primitives
46     System.out.println("---Printing primitive values---");
47     System.out.println("byte value: "+bytevalue);
48     System.out.println("short value: "+shortvalue);
49     System.out.println("int value: "+intvalue);
50     System.out.println("long value: "+longvalue);
51     System.out.println("float value: "+floatvalue);
52     System.out.println("double value: "+doublevalue);
53     System.out.println("char value: "+charvalue);
54     System.out.println("boolean value: "+boolvalue);
55 }
56 }

```

## 21. Creating user defined package

```

1 //Creating user-defined package..
2
3 package mypack;
4
5 public class Simple{
6     public static void main(String args[]){
7         System.out.println("Welcome to package");
8     }
9 }

```

## 22. Java StringTokenizer

```

1 import java.util.StringTokenizer;
2
3 public class Simple {
4     public static void main(String args[]){
5         StringTokenizer st = new StringTokenizer("Java OOP Programing Language", " ");
6         while (st.hasMoreTokens()) {
7             System.out.println(st.nextToken());
8         }
9     }
}

```

## **Unit – 3**

# **Exception Handling, Threading and Streams (Input and Output)**



### 23. Exception Handling

```
1 //Exception Handling Demonstration
2 public class Main
3 {
4     public static void main(String[] args) {
5         int a=10,b=0,c=0;
6         System.out.println("Start of main()");
7         try{
8             c=a/b;
9         }catch(ArithmeticException ae) {
10             System.out.println(ae);
11         }finally {
12             System.out.println("I am always there...");
13         }
14         System.out.println("Value of C:"+c);
15         System.out.println("End of main()");
16     }
17 }
```

### 24. Multiple catch statements

```
1 //multiple catch statements
2 public class Main {
3
4     public static void main(String[] args) {
5
6         try{
7             int a[]=new int[5];
8             a[5]=30/0;
9         }
10        catch(ArithmeticException e)
11        {
12            System.out.println("Arithmetic Exception occurs");
13        }
14        catch(ArrayIndexOutOfBoundsException e)
15        {
16            System.out.println("ArrayIndexOutOfBoundsException occurs");
17        }
18        catch(Exception e)
19        {
20            System.out.println("Parent Exception occurs");
21        }
22        System.out.println("rest of the code");
23    }
24 }
```

### 24. Custom exception

```
1 //Custom exception example...
2 class InvalidAgeException extends Exception{
3     InvalidAgeException(String s){
4         super(s);
5     }
6 }
7 class Main {
8
9     static void validate(int age)throws InvalidAgeException{
10         if(age<18)
11             throw new InvalidAgeException("not valid");
12         else
13             System.out.println("welcome to vote");
14     }
15
16     public static void main(String args[]){
17         try{
18             validate(13);
19         }catch(Exception m){System.out.println("Exception occurred: "+m);}
20     }
```

```

21     System.out.println("rest of the code...");
22 }
23 }

```

## 25. Multithreading using Thread Class

```

1  public class ThreadDemo1 {
2
3      public static void main(String[] args) {
4          System.out.println("Start of main");
5          MyThread1 mt1 = new MyThread1();
6          MyThread2 mt2 = new MyThread2();
7          mt1.start();
8          mt2.start();
9          System.out.println("End of main");
10     }
11 }
12
13 class MyThread1 extends Thread{
14     public void run(){
15         for(int i=1;i<=10;i++) {
16             System.out.println("MyThread-1."+i);
17         }
18     }
19 }
20
21 class MyThread2 extends Thread{
22     public void run(){
23         for(int i=1;i<=10;i++) {
24             System.out.println("MyThread-2."+i);
25         }
26     }
27 }

```

## 26. Multithreading using Runnable interface

```

1  public class ThreadDemo2 {
2      public static void main(String[] args) {
3          System.out.println("Start of main");
4          MyThread mt = new MyThread();
5          Thread t1 = new Thread(mt,"Thread-1");
6          Thread t2 = new Thread(mt,"Thread-2");
7          t1.start();
8          t2.start();
9          System.out.println("End of main");
10     }
11 }
12
13 class MyThread implements Runnable {
14     public void run() {
15         for(int i=1;i<=10;i++) {
16             System.out.println(Thread.currentThread().getName()+"."+i);
17         }
18     }
19 }

```

## 27. Thread Scheduling

```

1  public class ThreadDemo3 {
2
3      public static void main(String[] args) {
4          System.out.println("Start of main");
5          MyThread1 mt1 = new MyThread1();
6          MyThread2 mt2 = new MyThread2();
7          mt1.start();
8          mt2.start();
9          System.out.println("End of main");
10     }
11 }
12 }
13 }

```

```

14 class MyThread1 extends Thread{
15     public void run(){
16         for(int i=1;i<=10;i++) {
17             System.out.println("MyThread-1."+i);
18             Thread.yield();
19         }
20     }
21 }
22
23 class MyThread2 extends Thread{
24     public void run(){
25         for(int i=1;i<=10;i++) {
26             System.out.println("MyThread-2."+i);
27             Thread.yield();
28         }
29     }
30 }

```

## 28. Thread Joins

```

1 public class ThreadDemo3 {
2
3     public static void main(String[] args) {
4         try {
5             System.out.println("Start of main");
6             MyThread1 mt1 = new MyThread1();
7             MyThread2 mt2 = new MyThread2();
8             mt1.start();
9             mt1.join();
10            mt2.start();
11            mt2.join();
12            System.out.println("End of main");
13        }catch(Exception e){}
14    }
15
16 }
17
18 class MyThread1 extends Thread{
19     public void run(){
20         for(int i=1;i<=10;i++) {
21             System.out.println("MyThread-1."+i);
22             try {
23                 sleep(100);
24             }catch(Exception e){ }
25         }
26     }
27 }
28
29 class MyThread2 extends Thread{
30     public void run(){
31         for(int i=1;i<=10;i++) {
32             System.out.println("MyThread-2."+i);
33             try {
34                 sleep(200);
35             }catch(Exception e){ }
36         }
37     }
38 }

```

## 29. Thread Priorities

```

1 public class ThreadDemo4 {
2
3     public static void main(String[] args) {
4         System.out.println("Start main");
5         MyThread mt = new MyThread();
6         Thread t1 = new Thread(mt,"Thread-1");
7         Thread t2 = new Thread(mt,"Thread-2");
8         t1.start();
9         t2.start();
10        t2.setPriority(t1.getPriority()+5);
11        System.out.println("End main");
12    }
13
14 }
15

```

```

16 class MyThread implements Runnable {
17     public void run() {
18         for(int i = 1; i <= 10; i++) {
19             System.out.println(Thread.currentThread().getName());
20         }
21     }
22 }

```

### 30. File Class

```

1  import java.io.*;
2
3  public class IODemo1 {
4
5      public static void main(String[] args) {
6
7          try {
8              File f = new File("abc.txt");
9              if(f.createNewFile()) {
10                 System.out.println("File Successfully created");
11             }
12             else {
13                 System.out.println("File already exist");
14             }
15             System.out.println("File name : "+f.getName());
16             System.out.println("Path: "+f.getPath());
17             System.out.println("Absolute path: " +f.getAbsolutePath());
18             System.out.println("Parent: "+f.getParent());
19             System.out.println("Exists : "+f.exists());
20             System.out.println("Is writeable: "+f.canWrite());
21             System.out.println("Is readable: "+f.canRead());
22             System.out.println("Is a directory: "+f.isDirectory());
23             System.out.println("File Size in bytes: "+f.length());
24         } catch (Exception e) {
25             System.out.println(e);
26         }
27     }
28 }

```

### 31. Bytestream Class to read file

```

1  import java.io.*;
2
3  public class IODemo3 {
4      public static void main(String[] args) {
5          System.out.println("Content of output.txt file:\n");
6          try {
7              FileInputStream fin = new FileInputStream("output.txt");
8              int c;
9
10             while((c=fin.read())!= -1 ){
11                 System.out.print((char)c);
12             }
13         } catch (Exception e) { }
14     }
15 }

```

### 32. Bytestream Class to create file

```

1  import java.io.*;
2
3  public class IODemo2 {
4
5      public static void main(String[] args) {
6          try {
7              //DataInputStream out = new DataInputStream(System.in);
8              BufferedInputStream out = new BufferedInputStream(System.in);
9              FileOutputStream fout = new FileOutputStream("output.txt");
10             System.out.println("Enter text (enter & to end): &");
11             int ch;
12             while ((ch = (char) out.read()) != '&')
13                 fout.write((char)ch);
14             fout.close();
15         } catch (Exception e) {}
16     }
17 }

```

### 33. Character stream Class to read and write file

```
1 import java.io.File;
2 import java.io.FileReader;
3 import java.io.FileWriter;
4 import java.io.IOException;
5 public class IOStreamsExample {
6     public static void main(String args[]) throws IOException {
7         //Creating FileReader object
8         File file = new File("D:/myFile.txt");
9         FileReader reader = new FileReader(file);
10        char chars[] = new char[(int) file.length()];
11        //Reading data from the file
12        reader.read(chars);
13        //Writing data to another file
14        File out = new File("D:/CopyOfmyFile.txt");
15        FileWriter writer = new FileWriter(out);
16        //Writing data to the file
17        writer.write(chars);
18        writer.flush();
19        System.out.println("Data successfully written in the specified file");
20    }
21 }
```

## **Unit – 4**

# **Applets, Layout Managers**

#### 34. HelloWorld Applet

```
1 //HelloWorld Applet.
2 import java.applet.Applet;
3 import java.awt.Graphics;
4
5
6 public class HelloWorldApplet extends Applet {
7     public void paint (Graphics g) {
8         g.drawString ("Hello World", 25, 50);
9     }
10 }
```

#### 35. Applet Life Cycle and Mouse Event Listener

```
1 import java.awt.event.MouseListener;
2 import java.awt.event.MouseEvent;
3 import java.applet.Applet;
4 import java.awt.Graphics;
5
6 public class ExampleEventHandling extends Applet implements MouseListener
7 {
8     StringBuffer strBuffer;
9
10    public void init() {
11        addMouseListener(this);
12        strBuffer = new StringBuffer();
13        addItem("initializing the apple ");
14    }
15
16    public void start() {
17        addItem("starting the applet ");
18    }
19
20    public void stop() {
21        addItem("stopping the applet ");
22    }
23
24    public void destroy() {
25        addItem("unloading the applet");
26    }
27
28    void addItem(String word) {
29        System.out.println(word);
30        strBuffer.append(word);
31        repaint();
32    }
33
34    public void paint(Graphics g) {
35        // Draw a Rectangle around the applet's display area.
36        g.drawRect(0, 0,
37            getWidth() - 1,
38            getHeight() - 1);
39
40        // display the string inside the rectangle.
41        g.drawString(strBuffer.toString(), 10, 20);
42    }
43
44    public void mouseEntered(MouseEvent event) {
45        addItem("mouse entered! ");
46    }
47 }
```

```

48     public void mouseExited(MouseEvent event) {
49         addItem("mouse exit! ");
50     }
51
52     public void mousePressed(MouseEvent event) {
53     }
54
55     public void mouseReleased(MouseEvent event) {
56     }
57
58     public void mouseClicked(MouseEvent event) {
59         addItem("mouse clicked! ");
60     }

```

### 36. Applet Graphics

```

1  //Applet Graphics Demo
2  import java.applet.Applet;
3  import java.awt.Color;
4  import java.awt.Graphics;
5
6  public class GraphicsDemo extends Applet {
7
8      public void paint(Graphics g){
9          g.setColor(Color.red);
10         g.drawString("Welcome", 50, 50);
11         g.setColor(Color.black);
12         g.drawLine(20,30,50,300);
13         g.drawRect(70,100,30,30);
14         g.setColor(Color.blue);
15         g.fillRect(170,100,30,30);
16         g.drawOval(70,200,30,30);
17
18         g.setColor(Color.pink);
19         g.fillOval(170,200,30,30);
20         g.drawArc(90,150,30,30,30,270);
21         g.fillArc(270,150,30,30,0,180);
22     }
23 }

```

### 37. Passing Parameter in Applet

```

1  //Passing Parameter in Applet
2  import java.applet.Applet;
3  import java.awt.Graphics;
4  public class UseParam extends Applet{
5      public void paint(Graphics g){
6          String str=getParameter("msg");
7          g.drawString(str,50, 50);
8      }
9  }
10
11  /*
12  <html>
13      <body>
14          <applet code="UseParam.class" width="300" height="300">
15              <param name="msg" value="Welcome to applet">
16          </applet>
17      </body>
18  </html>
19  */

```



### 38. Image in Applet

```
1  import java.awt.*;
2  import java.applet.*;
3
4  public class DisplayImage extends Applet {
5
6      Image picture;
7
8      public void init() {
9          picture = getImage(getDocumentBase(),"car.jpg");
10     }
11
12     public void paint(Graphics g) {
13         g.drawImage(picture, 30,30, this);
14     }
15 }
```

### 39. Border layout

```
1  package BorderLayout;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6  public class BorderLayoutDemo {
7      JFrame f;
8      BorderLayoutDemo() {
9          f=new JFrame();
10         JButton b1=new JButton("NORTH");
11         JButton b2=new JButton("SOUTH");
12         JButton b3=new JButton("EAST");
13         JButton b4=new JButton("WEST");
14         JButton b5=new JButton("CENTER");
15         f.add(b1,BorderLayout.NORTH);
16         f.add(b2,BorderLayout.SOUTH);
17         f.add(b3,BorderLayout.EAST);
18         f.add(b4,BorderLayout.WEST);
19         f.add(b5,BorderLayout.CENTER);
20         f.setSize(300,300);
21         f.setVisible(true);
22     }
23
24     public static void main(String[] args) {
25         new BorderLayoutDemo();
26     }
27 }
```

### 40. Grid layout

```
1  package GridLayoutDemo;
2
3  import javax.swing.*;
4  import java.awt.*;
5
6  public class GridLayoutDemo {
7
8      GridLayoutDemo(){
9          JFrame f=new JFrame();
10         JButton b1=new JButton("1");
11         JButton b2=new JButton("2");
12         JButton b3=new JButton("3");
13         JButton b4=new JButton("4");
14         JButton b5=new JButton("5");
```

```
15         JButton b6=new JButton("6");
16         JButton b7=new JButton("7");
17         JButton b8=new JButton("8");
18         JButton b9=new JButton("9");
19         f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
20         f.add(b6);f.add(b7);f.add(b8);f.add(b9);
21         f.setLayout(new GridLayout(3,3));
22         f.setSize(300,300);
23         f.setVisible(true);
24     }
25
26     public static void main(String[] args) {
27         new GridLayoutDemo();
28     }
29 }
```

## **Unit – 5**

# **GUI using SWING Event Handling**

#### 41. JFrame and JPanel

```
1 //JFrame and JPanel Demo.
2
3 package swing1;
4
5 import java.awt.FlowLayout;
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8 import javax.swing.JLabel;
9 import javax.swing.JPanel;
10
11 public class Swing1 {
12     public static void main(String[] args) {
13         JFrame frame = new JFrame("JFrame Example");
14         JPanel panel = new JPanel();
15         panel.setLayout(new FlowLayout());
16         JLabel label = new JLabel("JFrame By Example");
17         JButton button = new JButton();
18         button.setText("Button");
19         panel.add(label);
20         panel.add(button);
21         frame.add(panel);
22         frame.setSize(200, 300);
23         frame.setLocationRelativeTo(null);
24         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         frame.setVisible(true);
26     }
27 }
28 }
```

#### 42. JButton with Event

```
1 //JButton with Event Example
2 package swing2;
3 import javax.swing.*;
4 import java.awt.event.*;
5
6 public class Swing2 {
7
8     public static void main(String[] args) {
9         JFrame f=new JFrame("Button Example");
10        final JTextField tf=new JTextField();
11        tf.setBounds(50,50, 150,20);
12        JButton b=new JButton("Click Here");
13        b.setBounds(50,100,95,30);
14        b.addActionListener(new ActionListener(){
15            public void actionPerformed(ActionEvent e){
16                tf.setText("Welcome to Swing in Java");
17            }
18        });
19        f.add(b);f.add(tf);
20        f.setSize(400,400);
21        f.setLayout(null);
22        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23        f.setVisible(true);
24    }
25 }
```

#### 43. JTextField Example

```
1 //JTextField Example
2
3 package swing3;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 public class Swing3 implements ActionListener {
8
9     JTextField tf1,tf2,tf3;
10    JButton b1,b2;
11    Swing3(){
12        JFrame f= new JFrame("Calculator");
13        tf1=new JTextField();
14        tf1.setBounds(50,50,150,20);
15        tf2=new JTextField();
16        tf2.setBounds(50,100,150,20);
17        tf3=new JTextField();
18        tf3.setBounds(50,150,150,20);
19        tf3.setEditable(false);
20        b1=new JButton("+");
21        b1.setBounds(50,200,50,50);
22        b2=new JButton("-");
23        b2.setBounds(120,200,50,50);
24        b1.addActionListener(this);
25        b2.addActionListener(this);
26        f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
27        f.setSize(300,300);
28        f.setLayout(null);
29        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30        f.setVisible(true);
31    }
32    public void actionPerformed(ActionEvent e) {
33        String s1=tf1.getText();
34        String s2=tf2.getText();
35        int a=Integer.parseInt(s1);
36        int b=Integer.parseInt(s2);
37        int c=0;
38        if(e.getSource()==b1){
39            c=a+b;
40        }else if(e.getSource()==b2){
41            c=a-b;
42        }
43        String result=String.valueOf(c);
44        tf3.setText(result);
45    }
46
47    public static void main(String[] args) {
48        new Swing3();
49    }
50 }
```

#### 44. CheckBox Example

```
1 //CheckBox Example
2
3 package swing4;
4
5 import javax.swing.*;
6 import java.awt.event.*;
7
8 public class Swing4 extends JFrame implements ActionListener {
9
```

```

10     JLabel l;
11     JCheckBox cb1,cb2,cb3;
12     JButton b;
13     Swing4(){
14         this.setTitle("Cafeteria");
15         l=new JLabel("Food Ordering System");
16         l.setBounds(50,50,300,20);
17         cb1=new JCheckBox("Pizza @ 100");
18         cb1.setBounds(100,100,150,20);
19         cb2=new JCheckBox("Burger @ 30");
20         cb2.setBounds(100,150,150,20);
21         cb3=new JCheckBox("Tea @ 10");
22         cb3.setBounds(100,200,150,20);
23         b=new JButton("Order");
24         b.setBounds(100,250,80,30);
25         b.addActionListener(this);
26         add(l);add(cb1);add(cb2);add(cb3);add(b);
27         setSize(400,400);
28         setLayout(null);
29         setVisible(true);
30         setDefaultCloseOperation(EXIT_ON_CLOSE);
31     }
32     public void actionPerformed(ActionEvent e){
33         float amount=0;
34         String msg="";
35         if(cb1.isSelected()){
36             amount+=100;
37             msg="Pizza: 100\n";
38         }
39         if(cb2.isSelected()){
40             amount+=30;
41             msg+="Burger: 30\n";
42         }
43         if(cb3.isSelected()){
44             amount+=10;
45             msg+="Tea: 10\n";
46         }
47         msg+="-----\n";
48         JOptionPane.showMessageDialog(this,msg+"Total: "+amount);
49     }
50
51     public static void main(String[] args) {
52         new Swing4();
53     }
54
55 }

```

#### 45. JList Example

```

1 //JList Demo.
2
3 package swing5;
4 import javax.swing.*;
5 import java.awt.event.*;
6 import static javax.swing.JFrame.EXIT_ON_CLOSE;
7
8 public class Swing5 {
9
10     Swing5(){
11         JFrame f= new JFrame();
12         final JLabel label = new JLabel();
13         label.setSize(500,100);
14         JButton b=new JButton("Show");

```

```

15      b.setBounds(200,150,80,30);
16      final DefaultListModel<String> l1 = new DefaultListModel<>();
17      l1.addElement("C");
18      l1.addElement("Python");
19      l1.addElement("Java");
20      l1.addElement("PHP");
21      final JList<String> list1 = new JList<>(l1);
22      list1.setBounds(100,100, 75,75);
23      DefaultListModel<String> l2 = new DefaultListModel<>();
24      l2.addElement("DJango");
25      l2.addElement("Struts");
26      l2.addElement("Spring");
27      l2.addElement("Larawel");
28      final JList<String> list2 = new JList<>(l2);
29      list2.setBounds(100,200, 75,75);
30      f.add(list1); f.add(list2); f.add(b); f.add(label);
31      f.setSize(450,450);
32      f.setLayout(null);
33      f.setVisible(true);
34      f.setDefaultCloseOperation(EXIT_ON_CLOSE);
35      b.addActionListener(new ActionListener() {
36          public void actionPerformed(ActionEvent e) {
37              String data = "";
38              if (list1.getSelectedIndex() != -1) {
39                  data = "Programming language Selected: " +
list1.getSelectedValue();
40                  label.setText(data);
41              }
42              if(list2.getSelectedIndex() != -1){
43                  data += ", FrameWork Selected: ";
44                  for(Object frame :list2.getSelectedValues()){
45                      data += frame + " ";
46                  }
47              }
48              label.setText(data);
49          }
50      });
51  }
52  public static void main(String[] args) {
53      new Swing5();
54  }
55
56  }

```