OBJECT ORIENTED PROGRAMMING LAB

STUDENT MANUAL

II Year I Semester

2022-23

Prepared by

Mrs.S.Raga Deepthi
Assistant Professor

Mrs. N. Rajeswari
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

Seshadrirao Knowledge Village, Gudlavalleru – 521356

GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institution with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521356

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

INSTITUTE VISION & MISSION

Institute Vision:

To be a leading Institution of Engineering Education and Research, preparing students for leadership in their fields in a caring and challenging learning environment.

Institute Mission:

- To produce quality engineers by providing state-of-the-art engineering education.
- To attract and retain knowledgeable, creative, motivated and highly skilled individuals whose leadership and contributions uphold the college tenets of education, creativity, research and responsible public service.
- To develop faculty and resources to impart and disseminate knowledge and information to students and also to society that will enhance educational level, which in turn, will contribute to social and economic betterment of society.
- To provide an environment that values and encourages knowledge acquisition and academic freedom, making this a preferred institution for knowledge seekers.
- To provide quality assurance.
- To partner and collaborate with industry, government, and R&D institutes to develop new knowledge and sustainable technologies and serve as an engine for facilitating the nation's economic development.
- To impart personality development skills to students that will help them to succeed and lead.
- To instil in students the attitude, values and vision that will prepare them to lead lives of personal integrity and civic responsibility.
- To promote a campus environment that welcomes and makes students of all races, cultures and civilizations feel at home.
- Putting students face to face with industrial, governmental and societal challenges.

DEPARTMENT VISION & MISSION

VISION

To be a centre of excellence in Computer Science and Engineering education and training to meet the challenging needs of the industry and society.

MISSION

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

PROGRAMME EDUCATIONAL OBJECTIVES(PEOs):-

PEO1: To exhibit analytical skills in modeling and solving computing problems by applying mathematical, scientific and engineering knowledge and to pursue their higher studies.

PEO2: To communicate effectively with multi-disciplinary teams to develop quality software systems with an orientation towards research and development for lifelong learning.

PEO3: To address industry and societal needs for the growth of global economy using emerging technologies by following professional ethics.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

Students will be able to

13. **PSO1**: Organize, maintain and protect IT Infrastructural resources.

PSO2: Design and Develop web, mobile, and smart apps based software solutions to the real world problems.

Course Objectives:

- To demonstrate object oriented programming concepts.
- To introduce the creation of GUI using AWT components.

Course Outcomes:

Upon successful completion of the course, the students will be able to

- use inheritance to extend the functionality of classes.
- prepare code that exhibits polymorphism.
- examine multi tasking with multi threading.
- create packages for reusability.
- create an effective GUI using AWT components.
- implement event handling.

Mapping Of Course Outcomes With Program Outcomes:

CT2509 : OBJECT ORIENTED PROGRAMMING LAB														
Course outcomes	Program Outcomes and Program Specific Outcome													
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1: use inheritance to extend the functionality of classes.	3	3	3		2			2	3	2		2		3
CO2: prepare code that exhibits polymorphism.	3	3	3		2			2	3	2		2		3
CO3: examine multi tasking with multi threading.	3	3	3		2			2	3	2		2		3
CO4: create packages for reusability.	3	3	3		2			2	3	2		2		3
CO5: create an effective GUI using AWT components.	3	3	3		2			2	3	2		2		3
CO6: implement event handling.	3	3	3		2			2	3	2		2		3
Object Oriented Programming Lab	3	3	3		3			3	3	3		3		3

LIST OF EXPERIMENTS

	class and Runnable interface) where the first thread displays "Good Morning" every one second, the second thread displays "Hello" every two seconds and the third thread displays "Welcome" every three seconds.		
12	Write JAVA code to handle keyboard events, which echoes keystrokes to the applet window and shows the status of each key event in the status bar.	CO1,5	32-34
13	Write JAVA code to display the position of x and y co-ordinates of the cursor movement using mouse.	CO1,5	35-37
14	Write JAVA code to create a list of items and display the selected item of the list in a text field. Arrange these components by using flow layout control.	CO1,6	38-40
15	Write JAVA code to arrange components by using border layout control	CO1,6	41-42

Experiment-I

Aim:

Write JAVA code to Display default initial values of all primitive data types in java.

Description:

- Java defines eight primitive data types ,they are: **byte, short, int, long, char, float, double, and boolean.**
- The primitive types are also commonly referred to as simple types
- These can be put in four groups:
  - **Integers:** This group includes byte, short, int, and long, which are for whole-valued signed numbers.
  - **Floating-point numbers:** This group includes float and double, which represent numbers with fractional precision
  - **Characters:** This group includes char, which represents symbols in a character set, like letters and numbers.
  - **Boolean:** This group includes boolean, which is a special type for representing true/false values.
- The primitive types represent single values—not complex objects.
- Although Java is otherwise completely object-oriented, the *primitive types are not*. They are analogous to the simple types found in most other non–object-oriented languages. The reason for this is efficiency. Making the primitive types into objects would have degraded performance too much.
- The following chart summarizes the default values for all the above data types.

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	Null
boolean	false

Program:

```
class Exp1 {
//instance variables with all the primitive types
  byte b;
  short s;
  int i;
  long l;

  float f;
  double d;

  char c;

  boolean bool;
   public static void main(String arg[]) {
        Exp1 obj=new Exp1();//Object

        System.out.println("Default initial value of byte variable is\t"+obj.b);
        System.out.println("Default initial value of short variable is\t"+obj.s);
        System.out.println("Default initial value of int variable is\t"+obj.i);
        System.out.println("Default initial value of long variable is\t"+obj.l);

        System.out.println("Default initial value of float variable is\t"+obj.f);
        System.out.println("Default initial value of double variable is\t"+obj.d);

        System.out.println("Default initial value of char variable is : ");

        System.out.println("Default initial value of boolean variable is\t"+obj.bool);
    }
}
```

OUTPUT:

```
Default initial value of byte variable is       0
Default initial value of short variable is      0
Default initial value of int variable is        0
Default initial value of long variable is       0
Default initial value of float variable is      0.0
Default initial value of double variable is     0.0
Default initial value of char variable is
Default initial value of boolean variable is false
```

Experiment-II

Aim:

Write JAVA code to Sort the given list of numbers in ascending order

Algorithm:

```
/*
Algorithm Bubblesort(a<array>, n)
  Input: a is an array with n elements to be sort.
  Output: array elements in ascending order.
  1. for( i = 0 to n-1)
        a) for( j = 0 to n-i-1)
              i) if ( a[j] > a[j+1] )
                    A) t = a[j]
                    B) a[j] = a[j+1]
                    C) a[j+1] = t
              ii) end if
        b) end j for loop
  2. end i for loop
End Bubblesort
*/
```

Program:

```
import java.util.*;
class Exp2 {
  public static void main(String arg[]) {
  Scanner sc=new Scanner(System.in);

  int a[],n;
  int i,j;
  System.out.println("Enter array size");
  n=sc.nextInt();
  System.out.println("Enter "+n+" elements");
  a=new int[n];
  for(i=0;i<n;i++)
        a[i]=sc.nextInt();

  System.out.println("Actual array elements are:");
  for(i=0;i<n;i++)
        System.out.print(a[i]+"\t");
```

```
  for(i=0;i<(n-1);i++) {
        for(j=0;j<(n-i-1);j++) {
              if ( a[j] > a[j+1] ) {
                    int t = a[j];
                    a[j] = a[j+1];
                    a[j+1] = t;
              }
        }
  }
  System.out.println("\nArray elements after sorting are:");
  for(i=0;i<n;i++)
        System.out.print(a[i]+"\t");
  }
}
```

OUTPUT:

Enter array size
6
Enter 6 elements
99
8
1
55
6
24
Actual array elements are:
99 8 1 55 6 24
Array elements after sorting are:
16 8 24 55 99

Experiment-III

Aim:

Write JAVA code to Test whether the given number is Armstrong number or not.

Description:

Given a number x, determine whether the given number is Armstrong number or not. A positive integer of **n digits** is called an Armstrong number of **order n** (order is number of digits) if.

abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ...

Example:

Input : 153
Output : Yes
153 is an Armstrong number.
1*1*1 + 5*5*5 + 3*3*3 = 153

Input : 120
Output : No
120 is not a Armstrong number.
1*1*1 + 2*2*2 + 0*0*0 = 9

Input : 1253
Output : No
1253 is not a Armstrong Number
1*1*1*1 + 2*2*2*2 + 5*5*5*5 + 3*3*3*3 = 723

Input : 1634
Output : Yes
1634 is an Armstrong number.
1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4 = 1634

The idea is to first count number digits (or find order). Let the number of digits be n. For every digit r in input number x, compute r^n. If sum of all such values is equal to n, then return true, else false.

Program:

```
import java.util.*;
public class Exp3 {
        public static void main(String args[]) {
                /*count number digits (or find order).
                    Let the number of digits be n.
                    For every digit r in input number x, compute rn.
                    If sum of all such values is equal to n, then return true, else false.*/
                int x, temp, n=0, r, sum=0;
                Scanner ip = new Scanner(System.in);
                System.out.println("Enter the number");
                x = ip.nextInt();
                temp = x;
                while(temp > 0) {
                        n++;
                        temp /= 10;
                }
                temp = x;
                while(temp > 0) {
                        r = temp % 10;
                        temp /= 10;
                        sum += Math.pow(r,n);
                }
                if(sum == x)
                 System.out.println("The given number " + x + " is an armstrong
number");
                else
                 System.out.println("The given number " + x + " is not an armstrong
number");
        }
}
```

OUTPUT:

Enter the number
5897
The given number 5897 is not an armstrong number

Enter the number
371
The given number 371 is an armstrong number

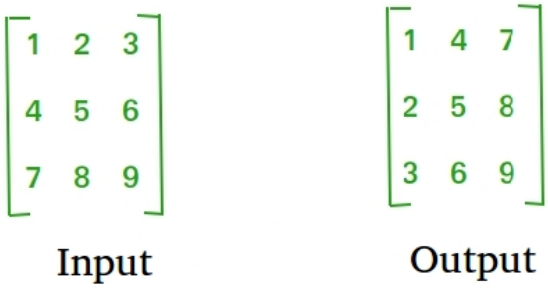
Experiment-IV

Aim:

Write JAVA code to Perform Transpose of a given matrix.

Description:

Transpose of a matrix is obtained by changing rows to columns and columns to rows. In other words, transpose of A[ ][ ] is obtained by changing A[i][j] to A[j][i].



Program:

```
import java.util.*;
class Exp4 {
        public static void main(String arg[]) {

                Scanner sc=new Scanner(System.in);

                int a[][],res[][],r,c,i,j;
                System.out.println("Enter order of the matrix:");
                r=sc.nextInt();
                c=sc.nextInt();

                a=new int[r][c];
                res=new int[c][r];

                System.out.println("\nEnter elements into the matrix:");
                for(i=0;i<r;i++)
                        for(j=0;j<c;j++)
                                a[i][j]=sc.nextInt();

                for(i=0;i<r;i++)
                        for(j=0;j<c;j++)
                                res[j][i]=a[i][j];
```

```
                System.out.println("The actual matrix is:\n");
                for(i=0;i<r;i++) {
                        System.out.print("\n");
                        for(j=0;j<c;j++)
                                System.out.print(a[i][j]+"\t");
                }
                System.out.println("\nThe transpose of the matrix is:\n");
                for(i=0;i<c;i++) {
                        System.out.print("\n");
                        for(j=0;j<r;j++)
                                System.out.print(res[i][j]+"\t");
                }
        }
}
```

OUTPUT:

Enter order of the matrix:
2
12 2

Enter elements into the matrix:
1
2
3
4
The actual matrix is:

1 2
3 4
The transpose of the matrix is:

1 3
2 4

Experiment-V

Aim:

Write JAVA code to Access the instance variables by using this key word

Description:

- 'this' is a reference variable that refers to the current object.
- 'this' keyword is used in the following ways:
  - **to refer current class instance variables**
    →*this.instance_variable*
  - **to invoke current class constructor**
    →this() → default constructor
    →this(p1, p2, ---) → parameterized constructor
  - **to invoke current class method**
    →this.methodName();
  - **as method parameter**
    →methodName(this)
  - **to return the current class instance**
    →return this
- It is specifically useful in the situations of **instance variable hiding**.
- When the local variable has the same name as instance variables, the local variables hide instance variables.
- In such situations, instance variables can be accessed by using "this" keyword.

Program:

```
import java.util.*;
public class Exp5 {
        int a, b, c; //3 instance variables
        public void assignValues(int a, int b, int c) {
                /*this keyword is mainly useful in the cases of instance variable hiding;
                        when instance variables and local variables have same,
                        the instance variables are hidden by the local variables*/
                this.a = a;       // this.a is instance variable; a is local variable
                this.b = b;       //this.b is instance variable; b is local variable
                this.c = c;       //this.c is instance variable; c is local variable
        }
        public static void main(String args[]) {
                Exp5 obj = new Exp5();
                int x, y, z;
                Scanner ip = new Scanner(System.in);
                System.out.println("Enter 3 values");
```

```
                x = ip.nextInt();
                y = ip.nextInt();
                z = ip.nextInt();
                System.out.println("Instance values before assignment:");
                System.out.println("a = " + obj.a);
                System.out.println("b = " + obj.b);
                System.out.println("c = " + obj.c);
                obj.assignValues(x, y, z);
                System.out.println("Instance values after assignment:");
                System.out.println("a = " + obj.a);
                System.out.println("b = " + obj.b);
                System.out.println("c = " + obj.c);
        }
}
```

OUTPUT:

Enter 3 values
2
9
8
Instance values before assignment:
a= 0
b= 0
c= 0
Instance values after assignment:
a= 2
b= 9
c= 8

Experiment-VI

Aim:

Write JAVA code to Access class members by using the keyword 'super'.

Description:

- It refers to superclass (parent) objects.
- It has two general forms:
  - Calls the constructor of super class → **super([param_list]);**
  - To access the members of super class that has been hidden by the members of sub class.
    - Instance variables in super class & sub class having the same name → **super.var_name**
    - Overridden methods in super class & sub class
      → **super.method_name([param_list]);**

Program:

```
import java.util.*;
class Super_Class {
        int a, b;          //instance variables of super class
        Super_Class(int i, int j) {       //Constructor of the Super class
                a = i;
                b = j;
        }
        void show() {
                System.out.println("super class show()");
                System.out.println("a = " + a);
        }
}
class Sub_Class extends Super_Class {
        int b, c;          //instance variables of sub class
        Sub_Class(int i,int j,int k,int l) {       //constructor of sub class
                super(i, j);   //invoking the super class constructor using 'super' keyword
                b = k;
                c = l;
        }
        void show() {
                System.out.println("sub class show()");
                //invoking the show( ) of super class using 'super' keyword
                super.show();
                //accessing super class instance variable using 'super' keyword
```

```
                System.out.println("b from super class is: " + super.b);
                System.out.println("b from sub class is: "+b);
                System.out.println("c = "+c);
        }
}
class Exp6 {
        public static void main(String arg[]) {
                int p, q, r, s;
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter 4 values");
                p = sc.nextInt();
                q = sc.nextInt();
                r = sc.nextInt();
                s = sc.nextInt();

                Super_Class obj1 = new Super_Class(p, q);
                Sub_Class obj2 = new Sub_Class(p, q, r, s);
                obj1.show();
                obj2.show();
        }
}
```

OUTPUT:

```
Enter 4 values
1
2
38
7
super class show()
a= 1
sub class show()
super class show()
a= 1
b from super class is: 2
b from sub class is: 3
c= 7
```

Experiment-VII

Aim:

Write JAVA code to Overload a method called area(), where the method computes the area of a square if the number of parameters are 1 otherwise if the numbers of parameters are 2 it needs to compute the area of a rectangle.

Description:

- Method overloading is one of Java's most exciting and useful features.
- To define two or more methods within the same class that share the same name, as long as their signatures are different. The process is referred to as Overloading Methods.
- Overloaded methods must differ in the type and/or number of their parameters.
- While overloaded methods may have different return types, the return type alone is insufficient to distinguish two versions of a method.
- When Java encounters a call to an overloaded method, it simply executes the version of the method whose parameters match the arguments used in the call.

Program:

```
import java.util.*;
class Overload {
        //Overloaded method area()
        void area(int side) {
            //area() with 1 parameter to computer area of square
            System.out.println("area of a square is " + (side*side) +" sqm");
        }
        void area(int length, int breadth) {
            //area() with 2 parameters to compute area of rectangle
            System.out.println("area of a rectangle is " + (length*breadth) + " sqm");
        }
}

class Exp7 {
        public static void main(String arg[]) {
                Scanner sc=new Scanner(System.in);
                int s, l, b;
                System.out.println("Enter side of a square in meters: ");
                s = sc.nextInt();
                System.out.println("Enter length of a rectangle in meters: ");
                l = sc.nextInt();
```

```
                System.out.println("Enter breadth of a rectangle in meters: ");
                b = sc.nextInt();

                Overload obj = new Overload();
                obj.area(s);
                obj.area(l, b);
        }
}
```

OUTPUT:

Enter side of a square in meters:
25
Enter length of a rectangle in meters:
8
Enter breadth of a rectangle in meters:
9
area of a square is 625 sqm
area of a rectangle is 72 sqm

Experiment-VIII

Aim:

Write JAVA code to Create an abstract class named shape, that contains an empty method named numberofsides(). Define three classes named Trapezoid, Triangle and Hexagon, such that each one of the classes contains only the method numberofsides(), that contains the number of sides in the given geometrical figure.

Description:

- A method without implementation details(without body) is called Abstract Method.
- A method with implementation is called Concrete Method.
- A class which has atleast an abstract method is called as Abstract Class.
- "abstract" keyword is used to make a method or a class abstract.

Program:

```
abstract class Shape {

        //abstract method that has to be overridden in its sub classes
        abstract void numberofSides();
}
class Trapezoid extends Shape {
        void numberofSides() {
                System.out.println("number of sides in a trapezoid are 4");
        }
}
class Triangle extends Shape {
        void numberofSides() {
                System.out.println("number of sides in a triangle are 3");
        }
}
class Hexagon extends Shape {
        void numberofSides() {
                System.out.println("number of sides in a hexagon are 6");
        }
}
class Exp8 {
        public static void main(String args[]) {
                Shape s;
                s=new Trapezoid();
                s. numberofSides();
                s=new Triangle();
```

```
                s. numberofSides();
                s=new Hexagon();
                s. numberofSides();
        }
}
```

OUTPUT:

number of sides in a trapezoid are 4
number of sides in a triangle are 3
number of sides in a hexagon are 6

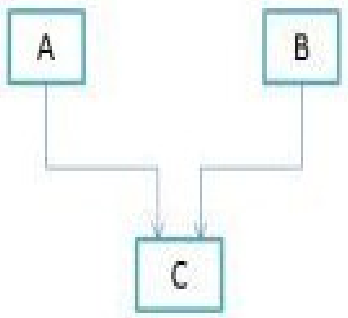
Experiment-IX

Aim:

Write JAVA code to Illustrate multiple inheritance.

Description:

- Multiple inheritance enables to derive a class from multiple parent classes.
- It is not supported by java directly, need to use interfaces.



- Java provides interface approach to support the concept of multiple inheritance.
- An interface can extend multiple interfaces and a class can implement multiple interfaces.

Program:

```
import java.util.*;
interface Car {
  int  speed = 90;        //constant --> variables that are public, static and final
  public void distance();         //abstract method
}
interface Bus {
  int distance = 100;    //constant
  public void speed();  //abstract method
}
class Vehicle implements Car, Bus {
  //a class implementing two interfaces --> multiple inheritance
  int time;
  Vehicle(int t) {
          time = t;
  }
  public void distance( ) {       //abstract method of interface Car
          System.out.println("distance travelled is " + (speed*time));
  }
  public void speed( ) {          //abstract method of interface Bus
          System.out.println("Speed of the vehicle is " + (distance/time));
  }
```

```
}
class Exp9 {
  public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        int t;
        System.out.println("Enter time: ");
        t = sc.nextInt();
        Vehicle v1 = new Vehicle(t);
        v1.distance();
        v1.speed();
  }
}
```

OUTPUT:

Enter time:
65
distance travelled is: 5850
Speed of the vehicle is: 1

Experiment-X

Aim:

Write JAVA code to Create and access a user defined package where the package contains a class named CircleDemo, which inturn contains a method called circleArea() which takes radius of the circle as the parameter and returns the area of the circle.

Description:

- Package is collection of related classes.
- To define a package, place "**package**" keyword as the **first statement** in the java source file so that any class declared within that file will belong to the specified package.
- The syntax of package creation is as follows:

  Syntax: package package_name;

  where pack_name is the name of the package.
- Existing packages can be accessed by means of "import" statement.
- "import" is a keyword that links the package with our program. It is placed before the class definitions.
- **import mypack.*;** to access all the classes in the package.
- **import mypack.ClassName;** to access specific class in the package.

Program:

```
package sub;   //a package "sub" is created; CircleDemo.class is kept inside package
public class CircleDemo {
        public double circleArea(float radius) {
                return (3.14f * radius * radius);
        }
}

import java.util.*;
import sub.*;
class Exp10 {
    public static void main(String arg[]) {
        Scanner sc=new Scanner(System.in);
        float r;
        System.out.println("Enter radius of a circle: ");
        r=sc.nextInt();

        CircleDemo cd=new CircleDemo();  //class imported from "sub" package
        System.out.println("Area of circle with radius :" + r + " is " + cd.circleArea(r));
    }
}
```

OUTPUT:

Enter radius of circle

7

Area of circle is154.0

Experiment-XI

Aim:

Write JAVA code to Create three threads (by using **Thread** class and **Runnable** interface) where the First thread displays "Good Morning" every one second, the second thread displays "Hello" every two seconds and the third thread displays "Welcome" every three seconds.

Description:

- A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.
- Thread is a lightweight process.
- Threads share the same address space and cooperatively share the same heavyweight process.
- Multithreading enables you to write very efficient programs that make maximum use of the CPU, because idle time can be kept to a minimum.
- Java's multithreading system is built upon the Thread class, its methods, and its companion interface, Runnable. Thread encapsulates a thread of execution.
- There are 2 ways to create a new thread, either extend the "Thread" class or implement the "Runnable" interface.

Program:

```
class Thread1 extends Thread {
        public void run() {
                try {
                        for(int i=0;i<10;i++) {
                                System.out.println("Good Morning");
                                Thread.sleep(100);
                        }
                }catch(InterruptedException ie)        {
                        System.out.println("Exception thrown from Thread1");
                }
        }
}
class Thread2 extends Thread {
        public void run() {
                try {
                        for(int i=0;i<10;i++) {
                                System.out.println("Hello");
                                Thread.sleep(200);
                        }
                }catch(InterruptedException ie)        {
                        System.out.println("Exception thrown from Thread2");
                }
        }
```

```
}
class Thread3 extends Thread {
        public void run() {
                try {
                        for(int i=0;i<10;i++) {
                                System.out.println("Welcome");
                                Thread.sleep(300);
                        }
                }catch(InterruptedException ie)        {
                        System.out.println("Exception thrown from Thread3");
                }
        }
}
class Exp11_T {
        public static void main(String args[]) {
                Thread1 t1 = new Thread1();
                Thread2 t2 = new Thread2();
                Thread3 t3 = new Thread3();
                t1.start();
                t2.start();
                t3.start();
        }
}


class Thread1 implements Runnable {
        public void run() {
                try {
                        for(int i=0;i<5;i++) {
                                System.out.println("Good Morning");
                                Thread.sleep(100);
                        }
                }catch(InterruptedException ie) {
                        System.out.println("Exception from Thread1");
                }
        }
}
class Thread2 implements Runnable {
        public void run() {
                try {
                        for(int i=0;i<5;i++) {
                                System.out.println("Hello");
                                Thread.sleep(200);
                        }
                }catch(InterruptedException ie) {
                        System.out.println("Exception from Thread2");
```

```
                }
        }
}
class Thread3 implements Runnable {
        public void run() {
                try {
                        for(int i=0;i<5;i++) {
                                System.out.println("Welcome");
                                Thread.sleep(300);
                        }
                }catch(InterruptedException ie) {
                        System.out.println("Exception from Thread3");
                }
        }
}
class Exp11_I {
        public static void main(String args[]) {
                Thread1 t1 = new Thread1();
                Thread nt1 = new Thread(t1);
                Thread2 t2 = new Thread2();
                Thread nt2 = new Thread(t2);
                Thread3 t3 = new Thread3();
                Thread nt3 = new Thread(t3);
                nt1.start();
                nt2.start();
                nt3.start();
        }
}
```

OUTPUT:

Welcome

Hello

Good Morning

Hello

Welcome

Hello

Good Morning

Welcome

Good Morning

Experiment-XII

Aim:

Write JAVA code to Handle keyboard events, which echoes keystrokes to the applet window and shows the status of each key event in the status bar.

Description:

- An event is an object that describes a state change in a source.
- A source is an object that generates an event.
- A listener is an object that is notified when an event occurs.
- The class AWTEvent is the superclass of all AWT-based events used by the delegation model.
- Commonly used Event classes are - ActionEvent, AdjustmentEvent, ComponentEvent, ContainerEvent, FocusEvent, InputEvent, ItemEvent, KeyEvent, MouseEvent, MouseWheelEvent, TextEvent, WindowEvent.

KeyEvent Class: A KeyEvent is generated when keyboard input occurs. There are three types of key events, which are identified by these integer constants:

- ✓ KEY_PRESSED
- ✓ KEY_RELEASED
- ✓ KEY_TYPED

- The first two events are generated when any key is pressed or released. The last event occurs when a character is generated.

Methods of KeyEvent Class:

- **char getKeyChar()** : returns the character that was enterd
- **int getKeyCode() :** returns the character code.

KeyListener Interface:

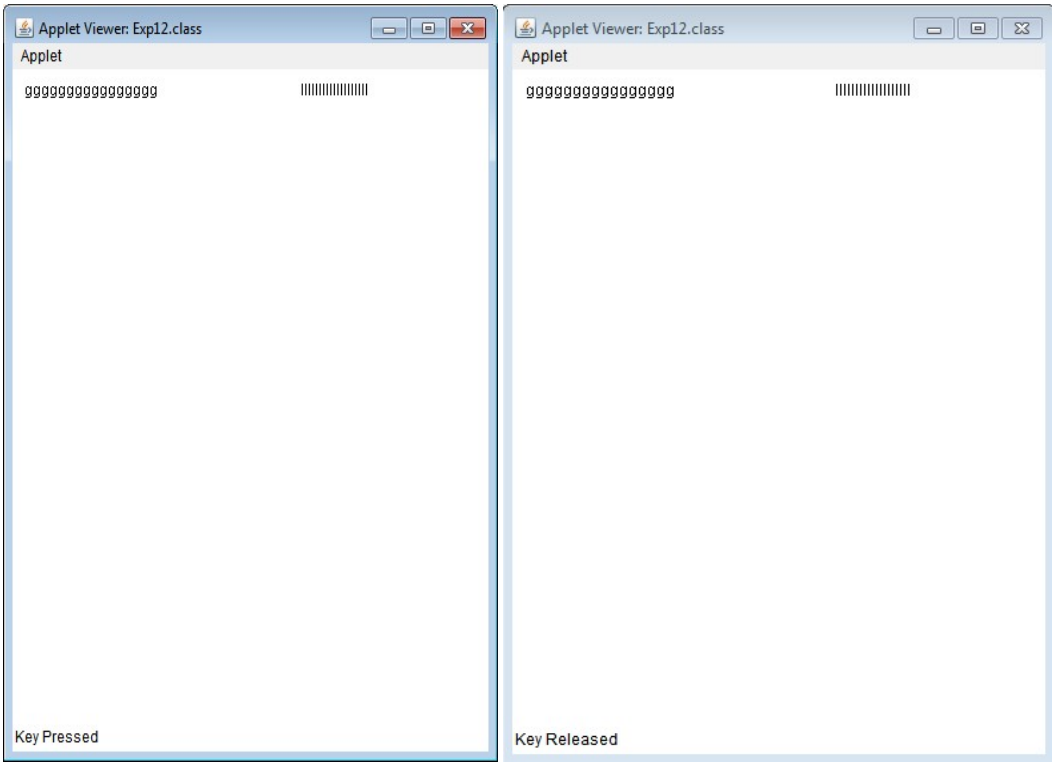
The interface defines three methods:

- void keyPressed(KeyEvent ke): it invokes when a key is pressed.
- void keyReleased(KeyEvent ke): it invokes when a key is released.
- Void keyTyped(KeyEvent ke): it invokes when a key is typed.

Program:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
<applet code="Exp12.class" width=400 height=500>
</applet>
*/
public class Exp12 extends Applet implements KeyListener {
        String msg;
        public void init( ) {
                msg = "";
                //registering the Listener; must override the methods of keyboard events
                addKeyListener(this);
        }
        public void keyPressed(KeyEvent ke) {
                showStatus("Key Pressed");
        }
        public void keyReleased(KeyEvent ke) {
                showStatus("Key Released");
        }
        public void keyTyped(KeyEvent ke) {
                msg += ke.getKeyChar();
                showStatus("Key Typed");
                repaint();
        }
        public void paint(Graphics g) {
                g.drawString(msg,10,20);
        }
}
```

OUTPUT:


Applet Viewer: Exp12.class
Applet
gggggggggggggggg
llllllllllllllll
Key Pressed
Applet Viewer: Exp12.class
Applet
gggggggggggggggg
llllllllllllllll
Key Released

Experiment-XIII

Aim:

Write JAVA code to Display the position of x and y co-ordinates of the cursor movement using mouse.

Description:

- An event is an object that describes a state change in a source.
- A source is an object that generates an event.
- A listener is an object that is notified when an event occurs.
- The class AWTEvent is the superclass of all AWT-based events used by the delegation model.
- Commonly used Event classes are - ActionEvent, AdjustmentEvent, ComponentEvent, ContainerEvent, FocusEvent, InputEvent, ItemEvent, KeyEvent, MouseEvent, MouseWheelEvent, TextEvent, WindowEvent.

MouseEvent Class: The MouseEvent class defines 8 types of mouse events. It defines the following integer constants that can be used to identify them:

1.MOUSE_CLICKED : The user clicked the mouse

2.MOUSE_DRAGGED : The user dragged the mouse

3.MOUSE_ENTERED : The mouse entered a component

4.MOUSE_EXITED : The mouse exited from a component

5.MOUSE_MOVED : The mouse moved

6.MOUSE_PRESSED : The mouse was pressed

7.MOUSE_RELEASED : The mouse was released

8.MOUSE_WHEEL : The mouse wheel was moved

Methods of MouseEvent Class:

1.int getX() : returns the X coordinates of the mouse when an event occurred

2.int getY(): returns the Y coordinates of the mouse when an event occurred.

3.Point getPoin(): returns the coordinates of the mouse.

MouseListener Interface:

This interface defines five methods:

1. void mouseClicked(MouseEvent me): it invokes if the mouse is pressed and released at the same point
2. void mouseEntered(MouseEvent me): it invokes when the mouse enters a component.
3. void mouseExited(MouseEvent me): it invokes when the mouse leaves the component.
4. void mousePressed(MouseEvent me): it invokes when the mouse is pressed.
5. void mouseReleased(MouseEvent me): it invokes when the mouse is released.

MouseMotionListener Interface:

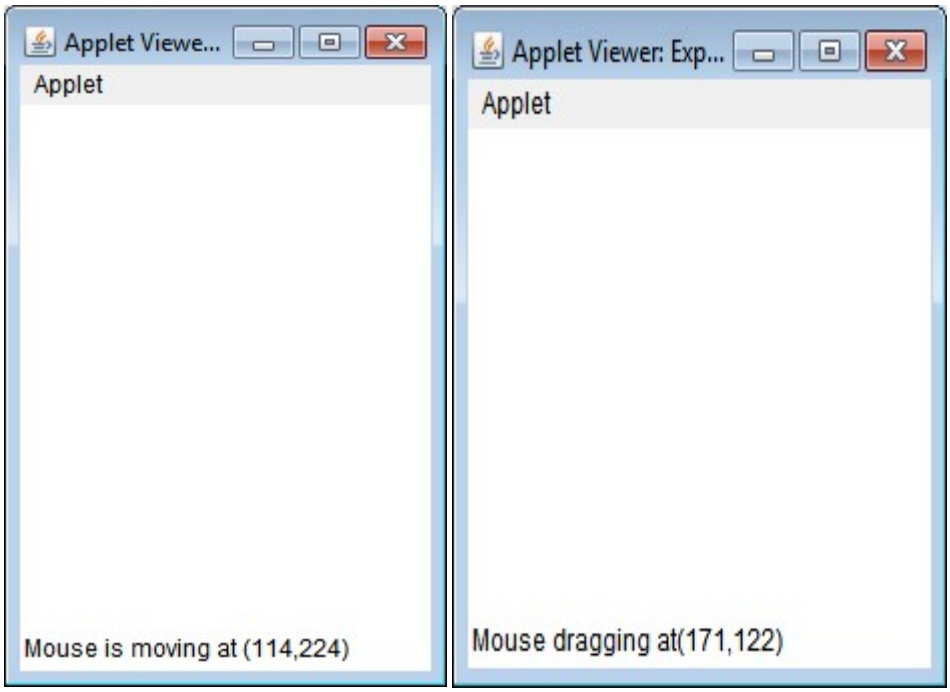
This interface defines two methods:

1. void mouseDragged(MouseEvent me): it invokes when multiple times as the mouse is dragged.
2. void mouseMoves(MovesEvent me): it invokes when multiple times as the mouse is moved

Program:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
<applet code="Exp13.class" width=400 height=500>
</applet>
*/
public class Exp13 extends Applet implements MouseMotionListener {
        String msg;
        public void init( ) {
                //registering the Listener; must override the methods of mouse events
                addMouseMotionListener(this);
        }
        public void mouseDragged(MouseEvent me) {
                showStatus("Mouse dragging at("+me.getX()+","+me.getY()+")");
        }
        public void mouseMoved(MouseEvent me) {
                showStatus("Mouse is moving at ("+me.getX()+","+me.getY()+")");
        }
        public void paint(Graphics g) {
                g.drawString(msg,10,20);
        }
}
```

OUTPUT:


Applet Viewe...
Applet
Mouse is moving at (114,224)
Applet Viewer: Exp...
Applet
Mouse dragging at(171,122)

Experiment-XIV

Aim:

Write JAVA code to Create a list of items and display the selected item of the list in a text field. Arrange these components by using Flow Layout control.

Description:

- The List class provides a compact, multiple-choice, scrolling selection list.
- The list can be configured so that user can choose either one item or multiple items.
- List class provides the following constructors and methods:

List Constructor or Method Name	Description
List()	Creates a new scrolling list.
List(int rows)	Creates a new list with the number of visible rows specified.
List(int rows, boolean multipleMode)	Creates a new list with the number of visible rows and the specified selection mode behavior.
void add(String item)	Adds a new item to the list.
void add(String item, int index)	Adds a new item at the specified position.
void addItem(Sring item)	Adds a new item to the list.
void addItem(String item, int index)	Adds a new item at the specified position.
void deselect(int index)	Deselects the item at the specified index.
String getItem(int index)	Gets the item at the specified index.
int getItemCount()	Gets the number of elements in the list
String [] getItems()	Gets an array of the element names.
int getRows()	Gets the number of lines that are currently visible.
int getSelectedIndex()	Gets the index of the selected item.
int [] getSelectedIndexes()	Gets a list of items that are all selected.
String getSelectedItem()	Gets the string that represents the text of the

	selected item.
String [] getSelectedItems()	Gets a list of the strings that represent the selected items.
Object [] getSelectedObjects()	Gets a list of selected strings as Objects.
int getVisibleIndex()	Gets the index of the item that was last made visible by the makeVisible() method.
boolean isIndexSelected (int index)	Indicates if the specified index represents a selected element.
boolean isMultipleMode()	Indicates if multiple elements can be simultaneously selected.
void makeVisible(int index)	Makes the item at the specified index visible.
void remove(int position)	Removes the item at the specified position.
void remove(String item)	Removes the first occurrence of the item that matches the string.
void removeAll()	Removes all items from this list.
void replaceItem(String newValue, int index)	Replaces the item at the specified position with the new item.
void select(int index)	Selects the item at the specified position.
void setMultipleMode (boolean b)	Makes this list use a multiple selection policy.

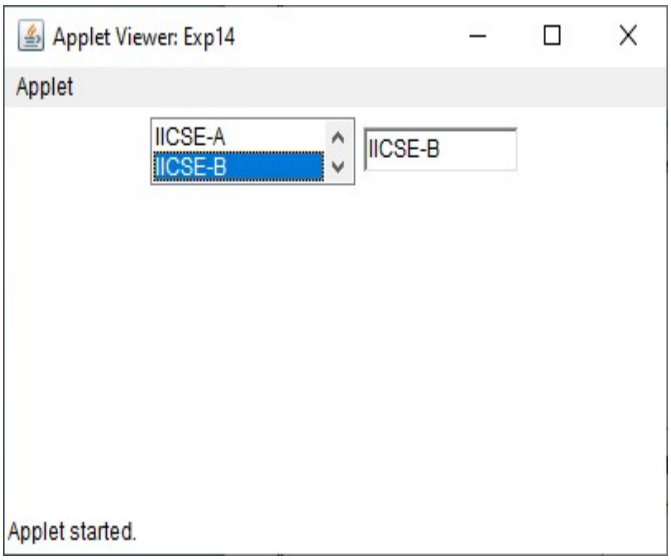
Handling Lists

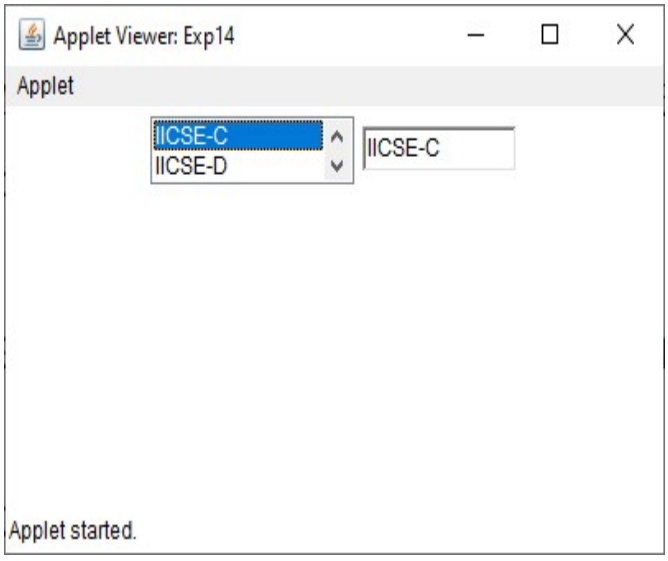
- To process list events, you will need to implement the ActionListener interface.
- Each time a List item is double-clicked, an ActionEvent object is generated. Its getActionCommand( ) method can be used to retrieve the name of the newly selected item.
- Also, each time an item is selected or deselected with a single click, an ItemEvent object is generated. Its getStateChange( ) method can be used to determine whether a selection or deselection triggered this event.
- getItemSelectable( ) returns a reference to the object that triggered this event.

Program:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="Exp17" width=400 height=200>
</applet>
*/
public class Exp17 extends Applet implements ItemListener {
        List l1;
        TextField tf;
        public void init( ) {
                setLayout(new FlowLayout());
                l1=new List(2); //creating object of List Control
                //adding items into the List control
                l1.add("II CSE-A");
                l1.add("II CSE-B");
                l1.add("II CSE-C");
                l1.add("II CSE-D");
                add(l1);  //adding the List control to the window
                tf=new TextField(10);   //creating object of TextField control
                l1.addItemListener(this);          //registering Listener to handle ItemEvent
                add(tf);  //adding the TextField control to the window
        }
        public void itemStateChanged(ItemEvent ie) {
                  //changing the text of TextField to the value of item selected in the list
                  tf.setText(l1.getSelectedItem());
}
```

OUTPUT:


Applet Viewer: Exp14
Applet
IICSE-A
IICSE-B
IICSE-B
Applet started.


Applet Viewer: Exp14
Applet
IICSE-C
IICSE-D
IICSE-C
Applet started.

Experiment-XV

Aim:

Write JAVA code to Arrange components by using Border Layout control

Description:

- A layout Manger is an instance of any class that implement the **Layout Manager** interface.
- The Layout manager is set by the **setLayout()** method.
- **LAYOUTS** are following types:
  - ✓ Flow Layout.
  - ✓ Border Layout
  - ✓ Grid Layout
  - ✓ Card Layout
  - ✓ Grid Bag Layout
- The **BorderLayout**class implements a common layout style for top-level windows.
- It has four narrow, fixed-width components at the edges and one large area in the center.
- The four sides are referred to as north, south, east, and west. The middle area is calledthe center.
- Here are the constructors defined by **BorderLayout:**
  - BorderLayout( )
  - BorderLayout(int *horz*, int *vert*)
- The first form creates a **default** border layout.
- The second allows you to specify thehorizontal and vertical space left between components in *horz* and *vert,* respectively.
- **BorderLayout** defines the following constants that specify the regions:
  - BorderLayout.CENTER
  - BorderLayout.SOUTH
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.NORTH
- When adding components, you will use these constants with the following form of **add( )**, which is defined by **Container:**
- void add(Component *compObj,* Object *region*);
- Here, *compObj*is the component to be added, and *region* specifies where the componentwill be added.

Program:

```
import java.awt.*;
import java.applet.*;
import java.util.*;
/*
<applet code="Exp15" width=400 height=200>
</applet>
*/
public class Exp15 extends Applet {
        public void init( ) {
                setLayout(new BorderLayout());
                Button b1 = new Button("North");
                Button b2 = new Button("South");
                Button b3 = new Button("East");
                Button b4 = new Button("West");
                Button b5 = new Button("Center");
                add(b1, BorderLayout.NORTH);
                add(b2, BorderLayout.SOUTH);
                add(b3,  BorderLayout.EAST);
                add(b4, BorderLayout.WEST);
                add(b5, BorderLayout.CENTER);
        }
}
```

OUTPUT: