SVR ENGINEERING COLLEGE

AYYALURUMETTA (V), NANDYAL, KURNOOL DT.
ANDHRA PRADESH – 518502

2018-19

LABORATORY MANUAL

OF

Java Programming Laboratory

(15A05405)

(R-15 REGULATION)

Prepared by

Mr. J.OMKAR REDDY
Asst. Professor

For

B.Tech II YEAR – II SEM. (CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SVR ENGINEERING COLLEGE

(AFFILIATED TO JNTUA ANANTHAPURAM- AICITE-INDIA)
AYYALURUMETTA (V), NANDYAL, KURNOOL DT.
ANDHRA PRADESH – 518502

LAB MANUAL CONTENT

Java Programming Laboratory

(15A05405**)**

Institute Vision & Mission, Department Vision & Mission

1. PO, PEO& PSO Statements.
2. List of Experiments
3. CO-PO Attainment
4. Experiment Code and Outputs

1. Institute Vision & Mission, Department Vision & Mission

Institute Vision:

To produce Competent Engineering Graduates & Managers with a strong base of Technical & Managerial Knowledge and the Complementary Skills needed to be Successful Professional Engineers & Managers.

Institute Mission:

To fulfill the vision by imparting Quality Technical & Management Education to the Aspiring Students, by creating Effective Teaching/Learning Environment and providing State – of the – Art Infrastructure and Resources.

Department Vision:

To produce Industry ready Software Engineers to meet the challenges of 21st Century.

Department Mission:

- Impart core knowledge and necessary skills in Computer Science and Engineering through innovative teaching and learning methodology.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society.
- Cultivate the students with all-round competencies, for career, higher education and self-employability.

2. PO, PEO& PSO Statements

PROGRAMME OUTCOMES (POs)

PO-1: Engineering knowledge - Apply the knowledge of mathematics, science, engineering fundamentals of Computer Science& Engineering to solve complex real-life engineering problems related to CSE.

PO-2: Problem analysis - Identify, formulate, review research literature, and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3: Design/development of solutions - Design solutions for complex engineering problems related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, cultural, societal and environmental considerations.

PO-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.

PO-5: Modern tool usage - Select/Create and apply appropriate techniques, resources and modern engineering and IT tools and technologies for rapidly changing computing needs, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

PO-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 CSE professional engineering practice.

PO-7: Environment and Sustainability - Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO-8: Ethics - Apply ethical principles and commit to professional ethics and responsibilities and norms of the relevant engineering practices.

PO-9: Individual and team work - Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication - Communicate effectively on complex engineering activities with the engineering community and with the society-at-large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, give and receive clear instructions.

PO-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.

PO-12: Life-long learning - Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadcast context of technological changes.

Program Educational Objectives (PEOs):

PEO 1:Graduates will be prepared for analyzing, designing, developing and testing the software solutions and products with creativity and sustainability.

PEO 2: Graduates will be skilled in the use of modern tools for critical problem solvingand analyzing industrial and societal requirements.

PEO 3:Graduates will be prepared with managerial and leadership skills for career and starting up own firms.

Program Specific Outcomes (PSOs):

PSO 1: Develop creative solutions by adapting emerging technologies / tools for real time applications.

PSO 2: Apply the acquired knowledge to develop software solutions and innovative mobile apps for various automation applications

2.1 Subject Time Table

<table>
<tr><td colspan="9">SVR ENGINEERING COLLEGE::NANDYAL
DEPARTMENT OF CSE</td></tr>
<tr><td colspan="5">J.OMKAR REDDY</td><td colspan="4">II-II</td></tr>
<tr><td>Day/ Time</td><td>9:30 AM</td><td>10:20 AM</td><td>11:30 AM</td><td>12:20 PM-</td><td rowspan="9">LUNCH BREAK</td><td>02:00 PM</td><td>02:50 PM</td><td>03:40 PM</td></tr>
<tr><td></td><td>10:20 AM</td><td>11:10AM</td><td>12:20 PM</td><td>01:10 PM</td><td>02:50 PM</td><td>03:40 PM</td><td>04:30 PM</td></tr>
<tr><td>MON</td><td></td><td></td><td></td><td></td><td colspan="3">JAVA LAB</td></tr>
<tr><td>TUE</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>WED</td><td></td><td colspan="3">JAVA LAB</td><td></td><td></td><td></td></tr>
<tr><td>THU</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>FRI</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>SAT</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

LIST OF EXPERIMENTS (SYLLABUS)

(15A05405) JAVA PROGRAMMING LABORATORY

Course Objectives:

- Learn to use object orientation to solve problems and use java language to implement them.
- To experiment with the syntax and semantics of java language and gain experience with java programming

Course Outcomes:

- Ability to write portable programs which work in all environments
- Ability to create user friendly interfaces
- Ability to solve the problem using object oriented approach and design solutions which are robust

List of Experiments

1. Preparing and practice – Installation of Java software, study of any Integrated development environment, sample programs on operator precedence and associativity, class and package concept, scope concept, control structures, constructors and destructors. Learn to compile, debug and execute java programs.
2. Write Java program(s) on use of inheritance, preventing inheritance using final, abstract classes.
3. Write Java program(s) on dynamic binding, differentiating method overloading and overriding.
4. Write Java program(s) on ways of implementing interface.
5. Write a program for the following
    - Develop an applet that displays a simple message.
    - Develop an applet for waving a Flag using Applets andThreads.
6. Write Java program(s) which uses the exception handling features of the language, creates exceptions and handles them properly, uses the predefined exceptions, and create own exceptions
7. Write java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it's not a duplicate of any number already read. Display the complete set of unique values input after the user enters each new value.

Write Java program(s) on creating multiple threads, assigning priority to threads, synchronizing threads, suspend and resume threads

10) Write a java program to split a given text file into n parts. Name each part as the name of the original file followed by .part<n> where n is the sequence number of the part file.

11) Write a java program to create a super class called Figure that receives the dimensions of two dimensional objects. It also defines a method called area that computes the area of an object. The program derives two subclasses from Figure. The first is Rectangle and second is Triangle. Each of the sub classes override area() so that it returns the area of a rectangle and triangle respectively.

12) Write a Java program that creates three threads. 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

13) Design a simple calculator which performs all arithmetic operations. The interface should look like the calculator application of the operating system. Handle the exceptions if any.

14) Write a java program to handle mouse events

15) Write a java program to handle keyboard events

16) Write a java program that allows conduction of object type examination containing multiple choice questions, and true/false questions. At the end of the examination when the user clicks a button the total marks have to be displayed inthe form of the message.

17) Write a java program that creates menu which appears similar to the menu ofnotepad application of the Microsoft windows or any editor of your choice.

18) Write a java program that creates dialog box which is similar to the save dialog box of the Microsoft windows or any word processor of your choice.

19) Write a Java program that correctly implements producer consumer problemusing the concept of inter thread communication

20) Write a java program to find and replace pattern in a given file.

21).Use inheritance to create an exception super class called ExceptionA and exception sub classes ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a java Program to demonstrate that the catch block for type ExceptionA catches exception of type ExceptionB and ExceptionC.

22)Write a Java program which opens a connection to standard port on well known server, sends the data using socket and prints the returned data.

23)Write a Java program to create a URLConnection and use it to examine the documents properties and content.

24)Write a Java program which uses TCP/IP and Datagrams to communicate client and server.

25)Create an interface for stack with push and pop operations. Implement the stack in two ways: fixed size stack and Dynamic stack (stack size is increased when stack is full).

26)Create multiple threads to access the contents of a stack. Synchronize thread to prevent simultaneous access to push and pop operations.

References:

1. "Java: How to Program", P.J.Deitel and H.M.Deitel, PHI.
2. "Object Oriented Programming through Java", P.Radha Krishna, Universities Press.
3. "Thinking in Java", Bruce Eckel, Pearson Education
4. "Programming in Java", S.Malhotra and S.Choudhary, Oxford Univ.Press

1. CO-PO Attainment

SVR ENGINEERING COLLEGE			
Department:	**COMPUTER SCIENCE & ENGINEERING**		
Course Outcome Attainment - Internal Assessments			
Name of the faculty :	J.OMKAR REDDY //M MADHAVI LATHA	Academic Year:	2018-19
Branch & Section:	COMPUTER SCIENCE & ENGINEERING	Exam:	15A05405
Course:	JAVA PROGRAMMING LAB	Semester:	II-II SEM

Course Outcomes	**Internal Lab**		**Internal Lab**	**University Exam**
15A05405.1	3		3	3
15A05405.2	3		3	3
15A05405.3	3		3	3
15A05405.4	3		3	3
15A05405.5	3		3	3

Course Outcomes		Attainment Level
15A05405.1	Ability to write portable programs which work in all environments	**3**
15A05405.2	Ability to create user friendly interfaces	**3**
15A05405.3	Ability to solve the problem using object oriented approach and design solutions which are robust	**3**
15A05405.4	The skills to apply OOP in Java programming in problem solving	**3**
15A05405.5	Use of GUI components.	**3**
	Average Attainment	**3**

Overall Course Attainment	**3**

SVR ENGINEERING COLLEGE

DEPARTMENT	COMPUTER SCIENCE & ENGINEERING		

PROGRAM OUTCOME ATTAINMENT

Name of Faculty:	J.OMKAR REDDY //M MADHAVI LATHA	Academic Year	2018-19
Branch & Section:	COMPUTER SCIENCE & ENGINEERING	SUB CODE:	15A05405
Course:	Java Programming LAB	Semester:	II-II

COURSE OUTCOME ATTAINMENT

Course outcome attainment	**Internal lab**		**Internal lab**	**External lab**
15A05405.1	3		3	3
15A05405.2	3		3	3
15A05405.3	3		3	3
15A05405.4	3		3	3
15A05405.5	3		3	3

COURSE OUTCOMES AND PROGRAM OUTCOMES MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
15A05405.1	3	3	2	2	1	2			1		1	2	3	1
15A05405.2	3	2	1	1		1		1				1	2	2
15A05405.3	3	3	2	2	2	1	1			1			2	1
15A05405.4	3	3	1	2		2			2		2	2	2	2
15A05405.5	3	2	2	2		2		2		1			3	1
Average	3.0	2.6	1.6	1.8	1.5	1.6	1.0	1.5	1.5	1.0	1.5	1.7	2.4	1.4

PO-ATTAINMENT

		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
INTERNAL	15A05405.1	9	9	6	6	3	6			3		3	6	9	3
	15A05405.2	9	6	3	3		3		3				3	6	6
	15A05405.3	9	9	6	6	6	3	3			3			6	3
	15A05405.4	9	9	3	6		6			6		6	6	6	6

	15A05405.5	9	6	6	6		6		6		3			9	3
UNIVERSITY	15A05102.1	9	9	6	6	3	6			3		3	6	9	3
	15A05102.2	9	6	3	3		3		3				3	6	6
	15A05102.3	9	9	6	6	6	3	3			3			6	3
	15A05102.4	9	9	3	6		6			6		6	6	6	6
	15A05102.5	9	6	6	6		6		6		3			9	3
OVERALL	15A05102.1	3	3	3	3	3	3			3		3	3	3	3
	15A05102.2	3	3	3	3		3		3				3	3	3
	15A05102.3	3	3	3	3	3	3	3			3			3	3
	15A05102.4	3	3	3	3		3			3		3	3	3	3
	15A05102.5	3	3	3	3		3		3		3			3	3
Attainment		**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Faculty J.OMKAR REDDY //M MADHAVI LATHA	Head of the Department

TITLE: Install and Execute Java Programs SVRE/CSE/EXPT-JAVA1/21

Program 1(a):

Aim: Preparing and practice – Installation of Java software, study of any Integrated development environment.

Installing JDK is first step in learning Java Programming. If you are using Windows 8 or Windows 7 Operating System, than installing JDK is quite easy as you just need to follow instruction given by Java SE Installation wizard. Only thing which requires some attention is, choosing correct JDK installer based upon, whether you are running with *32-bit or 64-bit Windows 8* or Windows 7 OS. JDK 7 is latest Java version but JDK 6 is still most popular in software and programming world. You can choose to install JDK 7 or JDK 6 based upon your course material. In this Java tutorial, we will learn **how to install JDK 7 in Windows 8 operating system** by following step by step guide. Another thing, which is part of JDK installation is setting PATH for Java in Windows 8, this will enable to run javac and java command from any directory in Windows 8. See that link for step by step guide on setting PATH in Windows 8. Though, I will use Windows 8 operating system to install JDK 7, You can still *install JDK 6 or 7 in Windows 7 by following those steps*. In order to choose correct JDK installer, we first need to check if Windows is running with 32-bit or 64-bit Windows 8. For that you can go to Control Panel-->System and Security and check System Type, if it shows 32-bit operating system means you need Windows x86 Java Installer, if it shows 64-bit operating system than you need Windows x64 Java Installer. Let's see steps involved to install JDK 6 or JDK 7 in Windows 8 operating system.

Read more: http://javarevisited.blogspot.com/2013/02/how-to-install-jdk-7-on-windows-8-java-32bit-64.html#ixzz4hQAU9yZV

A Java IDE (Integrated Development Environment) is a software application which enables users to more easily write and debug Java programs. Many IDEs provide features like syntax highlighting and code completion, which help the user to code more easily.

Eclipse is a Free and Open Source IDE, plus a developer tool framework that can be extended for a particular development need. IBM was behind its development, and it replaced IBM VisualAge tool. The idea was to create a standard look and feel that can be extended via plugins. The extensibility distinguishes Eclipse from other IDEs. Eclipse was also meant to compete with Microsoft Visual Studio tools. Microsoft tools give a standard way of developing code in the Microsoft world. Eclipse gives a similar standard way of developing code in the Java world, with a big success so far. With the online error checking only, coding can be sped up by at least 50% (coding does not include programming).

The goals for Eclipse are twofold:

1. Give a standard IDE for developing code
2. Give a starting point, and the same look and feel for all other more sophisticated tools built on Eclipse

IBM's WSAD, and later IBM Rational Software Development Platform, are built on Eclipse.

Standard Eclipse features:

- Standard window management (perspectives, views, browsers, explorers, ...)
- Error checking as you type (immediate error indications, ...)
- Help window as you type (type ., or <ctrl> space, ...)

- Automatic build (changes in source code are automatically compiled, ...)
- Built-in debugger (full featured GUI debugger)
- Source code generation (getters and setters, ...)
- Searches (for implementation, for references, ...)
- Code refactoring (global reference update, ...)

Program 1(b):

Aim: Write a Java sample programs on operator precedence and associativity

Program:

```
  public class Precedence {
  public static void main(String[] args) {
    System.out.println( 3 + 3 * 2 );
    System.out.println( 3 * 3 - 2 );
    System.out.println( 3 * 3 / 2 );
    System.out.println("--");
     System.out.println( 1 * 1 + 1 * 1);
    System.out.println( 1 + 1 / 1 - 1);
    System.out.println( 3 * 3 / 2 + 2);
    System.out.println("--");
     int x = 1;
    System.out.println( x++ + x++ * --x );
     x = 1;
    System.out.println( x << 1 * 2 >> 1);
     x = 0xf;
    System.out.println( 0xf & 0x5 | 0xa );
  }
}
```

Output:

```
# java Precedence
9
7
4
--
2
1
6
--
5
2
15
```

Program 1(c):

Aim: Write a Java Program to calculate Area of Rectangle using classes

Program:

```
class Rectangle {
  double length;
  double breadth;
```

```
}

// This class declares an object of type Rectangle.
class RectangleDemo {
  public static void main(String args[]) {
    Rectangle myrect = new Rectangle();
    double area;

    // assign values to myrect's instance variables
    myrect.length = 10;
    myrect.breadth = 20;

    // Compute Area of Rectangle
    area =  myrect.length * myrect.breadth ;

    System.out.println("Area is " + area);
  }
}
```

Output:-

Area is 200.0

Program 1(d):

Aim: Write a Java Program to implement Constructors:

Constructors:It is used to initialize some values to instance variables during Object Creation.

```
public class constructoremp
 {
int eid;
String ename;
//user defined constructors executed during Object creation
constructoremp()
{
        eid=111;
        ename="HARI";
}
void disp()
{
        System.out.println("emp id=" +eid);

        System.out.println("emp name=" +ename);

}
                public static void main(String[] args)
            {
                constructoremp e=new constructoremp();
                e.disp();
        }

}
```

output:-

emp id=111

emp name=HARI

Program 2(a):

Aim: Write Java program(s) on use of inheritance.

PROGRAM:-

```
/ A class to display the attributes of the vehicle
class Vehicle {
  String color;
  int speed;
  int size;
  void attributes() {
    System.out.println("Color : " + color);
    System.out.println("Speed : " + speed);
    System.out.println("Size : " + size);
  }
}
// A subclass which extends for vehicle
class Car extends Vehicle {
  int CC;
  int gears;
  void attributescar() {
    // The subclass refers to the members of the superclass
    System.out.println("Color of Car : " + color);
    System.out.println("Speed of Car : " + speed);
    System.out.println("Size of Car : " + size);
    System.out.println("CC of Car : " + CC);
    System.out.println("No of gears of Car : " + gears);
  }
}
public class Test {
  public static void main(String args[]) {
    Car b1 = new Car();
    b1.color = "Blue";
    b1.speed = 200 ;
    b1.size = 22;
    b1.CC = 1000;
    b1.gears = 5;
    b1.attributescar();
  }
}
```

output is:-

```
Color of Car : Blue
Speed of Car : 200
Size of Car : 22
CC of Car : 1000
No of gears of Car : 5
```

Program 2(b):
Aim: write a Java Program Using final to Prevent Inheritance.
Program:

When a class is declared as final then it cannot be subclassed i.e. no any other class can extend it. This is particularly useful, for example, when creating an immutable class like the predefined String class. The following fragment illustrates final keyword with a class:

```
final class A
{
    // methods and fields
}
// The following class is illegal.
class B extends A
{
        // ERROR! Can't subclass A
}
```

TITLE: Method Overloading and Overriding SVRE/CSE/EXPT-JAVA/3/21

Program 3(a)
Aim: To write a Java program for implementing Method Overloading

Program:

```
class DisplayOverloading
{
   public void disp(char c)
   {
       System.out.println(c);
   }
   public void disp(char c, int num)
   {
       System.out.println(c + " "+num);
   }
}
class Sample
{
  public static void main(String args[])
  {
     DisplayOverloading obj = new DisplayOverloading();
     obj.disp('a');
     obj.disp('a',10);
  }
}
```

Output:-

```
a
a 10
```

Program 3)b):
Aim: To write a Java program for implementing Method Overriding

Program:

```
class Parent
{
   void show() { System.out.println("Parent's show()"); }
}

// Inherited class
class Child extends Parent
{
   // This method overrides show() of Parent
   @Override
   void show() { System.out.println("Child's show()"); }
}

// Driver class
class Main
{
   public static void main(String[] args)
   {
      // If a Parent type reference refers
      // to a Parent object, then Parent's
      // show is called
      Parent obj1 = new Parent();
      obj1.show();

      // If a Parent type reference refers
      // to a Child object Child's show()
      // is called. This is called RUN TIME
      // POLYMORPHISM.
      Parent obj2 = new Child();
      obj2.show();
   }
}
```

Output:

```
Parent's show()
Child's show()
```

TITLE: **Interface** SVRE/CSE/EXPT-JAVA/4/21

Program 4)a):
Aim: Write a Java program on implementing interface
Program:

```
interface MyInterface
{
  /* compiler will treat them as:
   * public abstract void method1();
   * public abstract void method2();
   */
  public void method1();
  public void method2();
}
class Demo implements MyInterface
{
  /* This class must have to implement both the abstract methods
   * else you will get compilation error
   */
  public void method1()
  {
        System.out.println("implementation of method1");
  }
  public void method2()
  {
        System.out.println("implementation of method2");
  }
  public static void main(String arg[])
  {
        MyInterface obj = new Demo();
        obj.method1();
  }
}
```

Output:

implementation of method1

Program 4(b):
Aim: Write a Java program(s) on ways of implementing interface.
.
Program:

```
import java.io.*;
 interface Vehicle {
       // all are the abstract methods.
   void changeGear(int a);
   void speedUp(int a);
   void applyBrakes(int a);
}
```

```
 class Bicycle implements Vehicle{
        int speed;
       int gear;
        // to change gear
   @Override
   public void changeGear(int newGear){
            gear = newGear;
   }
       // to increase speed
   @Override
   public void speedUp(int increment){
            speed = speed + increment;
   }
       // to decrease speed
   @Override
   public void applyBrakes(int decrement){
            speed = speed - decrement;
   }
   public void printStates() {
       System.out.println("speed: " + speed
           + " gear: " + gear);
   }
}
class Bike implements Vehicle {
       int speed;
       int gear;
       // to change gear
   @Override
   public void changeGear(int newGear){
            gear = newGear;
   }

   // to increase speed
   @Override
   public void speedUp(int increment){
            speed = speed + increment;
   }
       // to decrease speed
   @Override
   public void applyBrakes(int decrement){
            speed = speed - decrement;
   }
   public void printStates() {
       System.out.println("speed: " + speed
          + " gear: " + gear);
   }
    }
class GFG {
       public static void main (String[] args) {
          // creating an inatance of Bicycle
       // doing some operations
```

```
        Bicycle bicycle = new Bicycle();
        bicycle.changeGear(2);
        bicycle.speedUp(3);
        bicycle.applyBrakes(1);
         System.out.println("Bicycle present state :");
        bicycle.printStates();
        // creating instance of bike.
        Bike bike = new Bike();

       bike.changeGear(1);
        bike.speedUp(4);
        bike.applyBrakes(3);
         System.out.println("Bike present state :");
        bike.printStates();
    }
}
```

Output;
Bicycle present state :
speed: 2 gear: 2
Bike present state :
speed: 1 gear: 1

TITLE: Applet Program	SVRE/CSE/EXPT-Java/5/21 Date: 01-01- 2018

Program 5:

Aim: 5(a)To write a Java Program for Develop an applet that displays a simple message

Program:

```
import java.applet.*;
import java.awt.*;
 public class FA extends Applet
{
       private int w, h;
       public void init()
       {
          w = 45;
          h = 50;
       }

       public void paint(Graphics g)
       {
          g.drawRect(w, h, 20, 80);
       }
}
<html>
 <p> This file launches the 'A' applet: A.class! </p>
 <applet code="FA.class" height=200 width=320>
 No Java?!
 </applet>
 </html>
```

AIM:-

5(b). Write a Java Program for waving a Flag using Applets and Threads

Program:-

```
//<applet code="Flag.class" height=300 width=300></applet>

//<applet code="Flag1.class" height=300 width=300></applet>
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Flag1 extends Applet implements ActionListener {
Button but = new Button ("click");
double add = 0;
public void init () {
setBackground (Color.white);
add (but);
but.addActionListener (this);
}
public void paint (Graphics g) {
Dimension d = getSize();
int height = d.height, width = d.width;
int x[ ] = new int[2*width], y[ ] = new int[2*width];
for (int color=0; color<3; color++) {
switch(color) {
case 0: g.setColor (Color.orange); break;
case 1: g.setColor (Color.white); break;
case 2: g.setColor (Color.green);
}
// The following 2 lines are actually one:
int size = fillArrays
(x, y, width/8, 3*width/4, (1+color)*height/5, height/5, 5, add);
g.fillPolygon (x, y, size);
}
}
// The following 2 lines are actually one:
int fillArrays
(int h[ ], int v[ ], int xStart, int wd, int yStart, int ht, int step, double extra) {
int count = 0;
// top line of the colour band:
for (int i=xStart; (i<=xStart+wd); i += step) {
h[count] = i;
v[count++] = yStart+(int)(ht*Math.sin(i/(0.2*wd)+extra));
}
int n = count -1;
// The values of the bottom line are not calculated.
// They're equal to to the values of the top line + ht
for (int i=h[count-1]; i>=xStart; i-=step) {
h[count] = i;
v[count++] = v[n--]+ht;
}
return count;
}
public void actionPerformed (ActionEvent evt) {
add += 0.7;
```

```
repaint();
}
}
```

TITLE: Exception Handling	SVRE/CSE/EXPT-JAVA/6/21

Program 6:

Aim

Write Java program(s) which uses the exception handling features of the language, creates exceptions and handles them properly, uses the predefined exceptions, and create own exceptions

Program:

Examples of Built-in Exception:

- Arithmetic exception

```
// Java program to demonstrate ArithmeticException
class ArithmeticException_Demo
{
   public static void main(String args[])
   {
      try {
         int a = 30, b = 0;
         int c = a/b;  // cannot divide by zero
         System.out.println ("Result = " + c);
      }
      catch(ArithmeticException e) {
         System.out.println ("Can't divide a number by 0");
      }
   }
}
```

- Output:

```
Can't divide a number by 0
```

- NullPointer Exception

```
//Java program to demonstrate NullPointerException
class NullPointer_Demo
{
   public static void main(String args[])
   {
      try {
         String a = null; //null value
         System.out.println(a.charAt(0));
      } catch(NullPointerException e) {
         System.out.println("NullPointerException..");
      }
   }
}
```

- Output:

```
NullPointerException..
```

- StringIndexOutOfBound Exception

```
// Java program to demonstrate StringIndexOutOfBoundsException
```

```
class StringIndexOutOfBound_Demo
{
   public static void main(String args[])
   {
      try {
         String a = "This is like chipping "; // length is 22
         char c = a.charAt(24); // accessing 25th element
         System.out.println(c);
      }
      catch(StringIndexOutOfBoundsException e) {
         System.out.println("StringIndexOutOfBoundsException");
      }
   }
}
```

- Output:

```
StringIndexOutOfBoundsException
```

- FileNotFound Exception

```
//Java program to demonstrate FileNotFoundException
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
 class File_notFound_Demo {

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

         // Following file does not exist
         File file = new File("E://file.txt");

         FileReader fr = new FileReader(file);
      } catch (FileNotFoundException e) {
        System.out.println("File does not exist");
      }
   }
}
```

- Output:

```
File does not exist
```

- NumberFormat Exception

```
// Java program to demonstrate NumberFormatException
class  NumberFormat_Demo
{
   public static void main(String args[])
   {
      try {
         // "akki" is not a number
         int num = Integer.parseInt ("akki") ;

         System.out.println(num);
      } catch(NumberFormatException e) {
         System.out.println("Number format exception");
      }
```

```
    }
}
```
• Output:

Number format exception

• ArrayIndexOutOfBounds Exception

```
// Java program to demonstrate ArrayIndexOutOfBoundException
class ArrayIndexOutOfBound_Demo
{
    public static void main(String args[])
    {
        try{
            int a[] = new int[5];
            a[6] = 9; // accessing 7th element in an array of

                    // size 5
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println ("Array Index is Out Of Bounds");
        }
    }
}
```
• Output:

Array Index is Out Of Bounds

```
// Java program to demonstrate user defined exception

// This program throws an exception whenever balance
// amount is below Rs 1000
class MyException extends Exception
{
    //store account information
    private static int accno[] = {1001, 1002, 1003, 1004};

    private static String name[] =
                {"Nish", "Shubh", "Sush", "Abhi", "Akash"};

    private static double bal[] =
        {10000.00, 12000.00, 5600.0, 999.00, 1100.55};

    // default constructor
    MyException() {    }

    // parametrized constructor
    MyException(String str) { super(str); }

    // write main()
    public static void main(String[] args)
    {
        try  {
            // display the heading for the table
            System.out.println("ACCNO" + "\t" + "CUSTOMER" +
                                "\t" + "BALANCE");
```

```
        // display the actual account information
        for (int i = 0; i < 5 ; i++)
        {
           System.out.println(accno[i] + "\t" + name[i] +
                                   "\t" + bal[i]);

           // display own exception if balance < 1000
           if (bal[i] < 1000)
           {
              MyException me =
                new MyException("Balance is less than 1000");
              throw me;
           }
        }
      } //end of try


      catch (MyException e) {
        e.printStackTrace();
      }
   }
}
```

Output:-

```
ACCNO   CUSTOMER       BALANCE
1001    Nish    10000.0
1002    Shubh   12000.0
1003    Sush    5600.0
1004    Abhi    999.0
MyException: Balance is less than 1000
```

TITLE: Duplicate Number Deletion	SVRE/CSE/EXPT-JAVA/7/21

Program 7:

Aim:

Write java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it's not a duplicate of any number already read. Display the complete set of unique values input after the user enters each new value.

Program:

```
import java.util.Scanner;

public class Duplicate
 {

        public static void main(String[] args)
         {
                        int a[]={0,0,0,0,0},t,i,j,s=0,r=0;
                        Scanner z=new Scanner(System.in);
                        System.out.println("Enter 5 unique values between 10 & 100 ");
```

```
                for(j=0;j<5;j++)
                {
                        t=z.nextInt();
                        if(t>=10&&t<=100)
                        {
                                for(i=0;i<r;i++)
                                {
                                        if(a[i]==t)
                                        {
                                                s++;
                                        }
                                }
                        if(s>0)
                        {
                                System.out.println("Duplicate value found retry");
                                        s--;
                                        j--;
                                        continue;
                        }
                        else
                                {
                                        a[j]=t;

                                                r++;
                                }
                        }
                        else
                        {
                        System.out.println("value must be in between 10 & 100");
                                j--;
                        }
                }
                         System.out.print("The five unique values are ");
                for(i=0;i<5;i++)
                {
                                System.out.print(a[i]+" ");
                }

        }

}
```

Output:-

```
Enter 5 unique values between 10 & 100
0
value must be in between 10 & 100
10
20
10
Duplicate value found retry
30
```

40
50
The five unique values are 10 20 30 40 50

TITLE: MultiThreading Program	SVRE/CSE/EXPT-JAVA/8/21

Program 8:
Aim:
Write Java program(s) on creating multiple threads, assigning priority to threads, synchronizing threads.

Code for creating multiple thread.

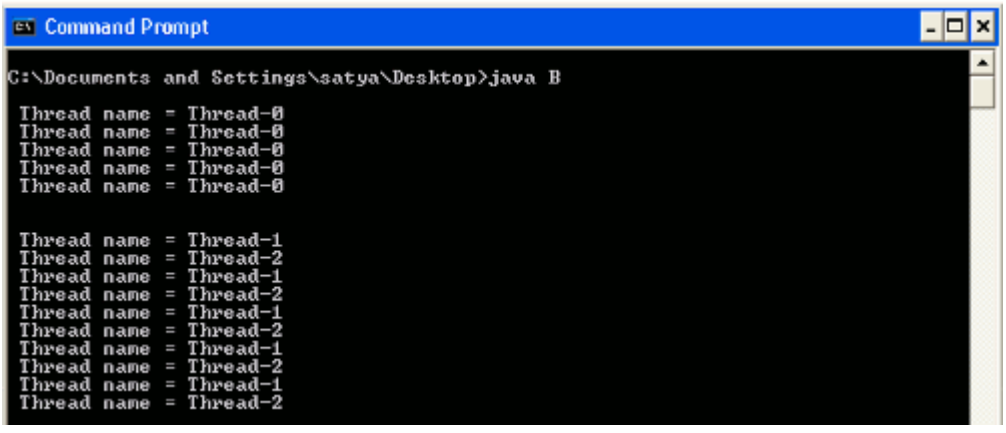
```
public class B extends Thread {

public void run() {
System.out.println();
for (int i = 1; i <= 5; i++) {
System.out.println(" Thread name = "+ Thread.currentThread().getName());
}
}
public static void main(String[] args) {
B t1 = new B();
t1.start();

B t2 = new B();
t2.start();

B t3 = new B();
t3.start();
}
```

Output : When you compile and execute the program the output will be as below




```
// Java program to demonstrate getPriority() and setPriority()
import java.lang.*;

class ThreadDemo extends Thread
{
```

```
    public void run()
    {
        System.out.println("Inside run method");
    }

    public static void main(String[]args)
    {
        ThreadDemo t1 = new ThreadDemo();
        ThreadDemo t2 = new ThreadDemo();
        ThreadDemo t3 = new ThreadDemo();

        System.out.println("t1 thread priority : " +
                           t1.getPriority()); // Default 5
        System.out.println("t2 thread priority : " +
                           t2.getPriority()); // Default 5
        System.out.println("t3 thread priority : " +
                           t3.getPriority()); // Default 5

        t1.setPriority(2);
        t2.setPriority(5);
        t3.setPriority(8);

        // t3.setPriority(21); will throw IllegalArgumentException
        System.out.println("t1 thread priority : " +
                           t1.getPriority());  //2
        System.out.println("t2 thread priority : " +
                           t2.getPriority()); //5
        System.out.println("t3 thread priority : " +
                           t3.getPriority());//8

        // Main thread
        System.out.print(Thread.currentThread().getName());
        System.out.println("Main thread priority : "
                    + Thread.currentThread().getPriority());

        // Main thread priority is set to 10
        Thread.currentThread().setPriority(10);
        System.out.println("Main thread priority : "
                    + Thread.currentThread().getPriority());
    }
}
```

Output:

```
t1 thread priority : 5
t2 thread priority : 5
t3 thread priority : 5
t1 thread priority : 2
t2 thread priority : 5
t3 thread priority : 8
Main thread priority : 5
Main thread priority : 10
```

example of multi threading with synchronized.

```
// A Java program to demonstrate working of  synchronized.
import java.io.*;
import java.util.*;
 // A Class used to send a message
class Sender
{
    public void send(String msg)
    {
        System.out.println("Sending\t"  + msg );
        try
        {
            Thread.sleep(1000);
        }
        catch (Exception e)
        {
            System.out.println("Thread  interrupted.");
        }
        System.out.println("\n" + msg + "Sent");
    }
}
 // Class for send a message using Threads
class ThreadedSend extends Thread
{
    private String msg;
    private Thread t;
    Sender  sender;
     // Recieves a message object and a string
    // message to be sent
    ThreadedSend(String m,  Sender obj)
    {
        msg = m;
        sender = obj;
    }

     public void run()
    {
        // Only one thread can send a message
        // at a time.
        synchronized(sender)
        {
            // synchronizing the snd object
            sender.send(msg);
        }
    }
}
 // Driver class
class SyncDemo
{
    public static void main(String args[])
    {
```

```
        Sender snd = new Sender();
        ThreadedSend S1 = new ThreadedSend( " Hi " , snd );
        ThreadedSend S2 = new ThreadedSend( " Bye " , snd );
         // Start two threads of ThreadedSend type
        S1.start();
        S2.start();
         // wait for threads to end
        try
        {
           S1.join();
           S2.join();
        }
        catch(Exception e)
        {
           System.out.println("Interrupted");
        }
    }
}
```

Output:

```
Sending         Hi

 Hi Sent
Sending         Bye

 Bye Sent
```

TITLE: split a given text file into n parts SVRE/CSE/EXPT-OS/9/21

Program 9:

Aim:

Write a java program to split a given text file into n parts. Name each part as the name of the original file followed by .part<n> where n is the sequence number of the part file.

Program:

```
import java.io.*;
import java.util.Scanner;
public class Split {
public static void main(String args[])
{
 try{
  // Reading file and getting no. of files to be generated
  String inputfile = "D:\\test.txt"; //  Source File Name.
  double nol = 2000.0; //  No. of lines to be split and saved in each output file.
  File file = new File(inputfile);
  Scanner scanner = new Scanner(file);
  int count = 0;
  while (scanner.hasNextLine())
  {
   scanner.nextLine();
   count++;
  }
  System.out.println("Lines in the file: " + count);    // Displays no. of lines in the input file.

  double temp = (count/nol);
  int temp1=(int)temp;
  int nof=0;
  if(temp1==temp)
  {
   nof=temp1;
  }
  else
  {
   nof=temp1+1;
  }
  System.out.println("No. of files to be generated :"+nof); // Displays no. of files to be generated.

  //--------------------------------------------------------------------------------------------------------

  // Actual splitting of file into smaller files

    FileInputStream fstream = new FileInputStream(inputfile); DataInputStream in = new
DataInputStream(fstream);

  BufferedReader br = new BufferedReader(new InputStreamReader(in)); String strLine;
```

```
 for (int j=1;j<=nof;j++)
 {
  FileWriter fstream1 = new FileWriter("C:/New Folder/File"+j+".txt");    // Destination File Location
  BufferedWriter out = new BufferedWriter(fstream1);
  for (int i=1;i<=nol;i++)
  {
   strLine = br.readLine();
   if (strLine!= null)
   {
    out.write(strLine);
    if(i!=nol)
    {
     out.newLine();
    }
   }
  }
  out.close();
 }
 in.close();
 }catch (Exception e)
 {
  System.err.println("Error: " + e.getMessage());
 }
}
}
```

TITLE: Super and Sub Classes	SVRE/CSE/EXPT-JAVA/10/21 Date: 01-01- 2018

Program 10:

Aim:

Write a java program to create a super class called Figure that receives the dimensions of two dimensional objects. It also defines a method called area that computes the area of an object. The program derives two subclasses from Figure. The first is Rectangle and second is Triangle. Each of the sub classes override area() so that it returns the area of a rectangle and triangle respectively.

PROGRAM:

```
// Using abstract methods and classes.
abstract class Figure {
double dim1;
double dim2;
Figure(double a, double b) {
dim1 = a;
dim2 = b;
}
// area is now an abstract method
abstract double area();
}
class Rectangle extends Figure {
Rectangle(double a, double b) {
```

```
super(a, b);
}
// override area for rectangle
double area() {
System.out.println("Inside Area for Rectangle.");


return dim1 * dim2;
}
}
class Triangle extends Figure {
Triangle(double a, double b) {
super(a, b);
}
// override area for right triangle
double area() {
System.out.println("Inside Area for Triangle.");
return dim1 * dim2 / 2;
}
}
class AbstractAreas {
public static void main(String args[]) {
// Figure f = new Figure(10, 10); // illegal now
Rectangle r = new Rectangle(9, 5);
Triangle t = new Triangle(10, 8);
Figure figref; // this is OK, no object is created
figref = r;
System.out.println("Area is " + figref.area());
figref = t;
System.out.println("Area is " + figref.area());
}
}
```

TITLE: Java program that creates three threads SVRE/CSE/EXPT-Java/11/21

Program 11:

Aim: Write a Java program that creates three threads. 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.

Program:

```
class A extends Thread
{
synchronized public void run()
{
try
{
while(true)
```

```
{
sleep(1000);
System.out.println("good morning");
}
}
catch(Exception e)
{                    }
}
}
class B extends Thread
{
synchronized public void run()
{
try
{
while(true)
{
sleep(2000);
System.out.println("hello");
}
}
catch(Exception e)
{                    }
}
}
class C extends Thread
{
synchronized public void run()
{
try
{
while(true)
{
sleep(3000);
System.out.println("welcome");
}
}
catch(Exception e)
{                    }
}
}
class ThreadDemo
{
public static void main(String args[])
{
A t1=new A();
B t2=new B();
C t3=new C();
```

```
t1.start();
t2.start();
t3.start();
}
}
```

OUTPUT:-

```
good morning
good morning
hello
welcome
good morning
hello
good morning
good morning
hello
welcome
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
good morning
hello
```

12)

AIM:- Design a simple calculator which performs all arithmetic operations .

Program:-

```
import java.util.Scanner;
        public class JavaProgram
    {
  public static void main(String args[])
  {
     float a, b, res;
     char choice, ch;
     Scanner scan = new Scanner(System.in);

     do
     {
        System.out.print("1. Addition\n");
        System.out.print("2. Subtraction\n");
        System.out.print("3. Multiplication\n");
        System.out.print("4. Division\n");
        System.out.print("5. Exit\n\n");
        System.out.print("Enter Your Choice : ");
        choice = scan.next().charAt(0);
        switch(choice)
        {
           case '1' : System.out.print("Enter Two Number : ");
              a = scan.nextFloat();
              b = scan.nextFloat();
              res = a + b;
              System.out.print("Result = " + res);
              break;
           case '2' : System.out.print("Enter Two Number : ");
              a = scan.nextFloat();
              b = scan.nextFloat();
              res = a - b;
              System.out.print("Result = " + res);
              break;
           case '3' : System.out.print("Enter Two Number : ");
              a = scan.nextFloat();
              b = scan.nextFloat();
              res = a * b;
              System.out.print("Result = " + res);
              break;
           case '4' : System.out.print("Enter Two Number : ");
              a = scan.nextFloat();
              b = scan.nextFloat();
              res = a / b;
              System.out.print("Result = " + res);
              break;
           case '5' : System.exit(0);
              break;
```

```
            default : System.out.print("Wrong Choice!!!");
                break;
            }
            System.out.print("\n---------------------------------------\n");
        }while(choice != 5);
    }
}
```

OUTPUT:-

```
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

Enter Your Choice : 1
Enter Two Number : 23
45
Result = 68.0
---------------------------------------
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
```

13)

AIM:-**Write a java program to handle mouse events.**

Program:-

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class MouseEvents extends Applet implements MouseListener,MouseMotionListener
{
    String msg=" ";
    public void init()
    {
        addMouseListener(this);
        addMouseMotionListener(this);
    }
  public void paint(Graphics g)
    {
        g.drawString(msg,20,20);
    }
    public void mousePressed(MouseEvent me)
    {
        msg=" mouse pressed";
        repaint();
    }
    public void mouseClicked(MouseEvent me)
    {
       msg=" mouse clicked";
        repaint();
    }
  public void mouseEntered(MouseEvent me)
    {
        msg=" mouse entered";
        repaint();
    }
  public void mouseExited(MouseEvent me)
    {
        msg=" mouse exited";
        repaint();
    }
  public void mouseReleased(MouseEvent me)
    {
        msg=" mouse released";
        repaint();
    }
    public void mouseMoved(MouseEvent me)
    {
        msg=" mouse moved";
        repaint();
```

```
    }
    public void mouseDropped(MouseEvent me)
    {
        msg=" mouse dropped";
        repaint();
    }
    public void mouseDragged(MouseEvent me)
    {
        msg=" mouse dragged";
      repaint();
    }
}
```

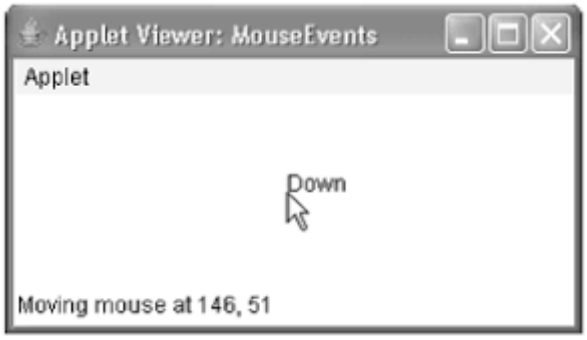
HTML Code

```
<html>
<title>Keyboard Events</title>
<Body><Applet code="MouseEvents.class" height=100 width=700></applet>
</body>
</html>
```

Output:-

F:\>javac MouseEvents.java

F:\>appletviewer MouseEvents.html



14)

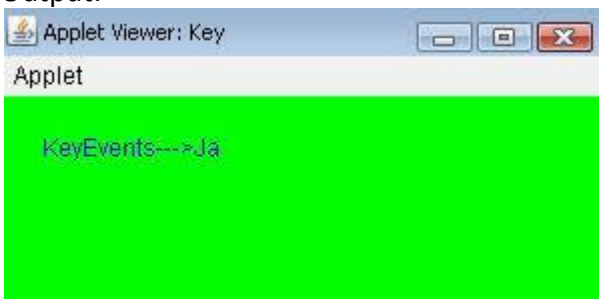
AIM:- Write a java program to handle keyboard events

Program:-

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="Key" width=300 height=400>
</applet>
*/
public class Key extends Applet
implements KeyListener
{
        int X=20,Y=30;
        String msg="KeyEvents--->";
        public void init()
        {
                addKeyListener(this);
                requestFocus();
                setBackground(Color.green);
                setForeground(Color.blue);
        }
        public void keyPressed(KeyEvent k)
        {
                showStatus("KeyDown");
                int key=k.getKeyCode();
                switch(key)
                {
                        case KeyEvent.VK_UP:
                                showStatus("Move to Up");
                                break;
                        case KeyEvent.VK_DOWN:
                                showStatus("Move to Down");
                                break;
                        case KeyEvent.VK_LEFT:
                                showStatus("Move to Left");
                                break;
                        case KeyEvent.VK_RIGHT:
                                showStatus("Move to Right");
                                break;
                }
                repaint();
        }
        public void keyReleased(KeyEvent k)
        {
```

```
                showStatus("Key Up");
        }
        public void keyTyped(KeyEvent k)
        {
                msg+=k.getKeyChar();
                repaint();
        }
        public void paint(Graphics g)
        {
                g.drawString(msg,X,Y);
        }
}
```

Output:-



TITLE: Dialog Box Program in Java	SVRE/CSE/EXPT-OS/15/21

AIM:

15) Write a java program that creates dialog box which is similar to the save dialog box of the Microsoft windows or any word processor of your choice.

PROGRAM:-

```
// Demonstrate Dialog box.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="DialogDemo" width=250 height=250>
</applet>
*/
// Create a subclass of Dialog.
class SampleDialog extends Dialog implements ActionListener {
SampleDialog(Frame parent, String title) {
super(parent, title, false);
setLayout(new FlowLayout());
setSize(300, 200);
add(new Label("Press this button:"));
Button b;
add(b = new Button("Cancel"));
b.addActionListener(this);
```

```
}
public void actionPerformed(ActionEvent ae) {
dispose();
}
public void paint(Graphics g) {
g.drawString("This is in the dialog box", 10, 70);
}
}
// Create a subclass of Frame.
class MenuFrame extends Frame {
String msg = "";
CheckboxMenuItem debug, test;
MenuFrame(String title) {
super(title);
// create menu bar and add it to frame
MenuBar mbar = new MenuBar();
setMenuBar(mbar);
// create the menu items
Menu file = new Menu("File");
MenuItem item1, item2, item3, item4;
file.add(item1 = new MenuItem("New..."));
file.add(item2 = new MenuItem("Open..."));
file.add(item3 = new MenuItem("Close"));
file.add(new MenuItem("-"));
file.add(item4 = new MenuItem("Quit..."));
mbar.add(file);

Menu edit = new Menu("Edit");
MenuItem item5, item6, item7;
edit.add(item5 = new MenuItem("Cut"));
edit.add(item6 = new MenuItem("Copy"));
edit.add(item7 = new MenuItem("Paste"));
edit.add(new MenuItem("-"));
Menu sub = new Menu("Special", true);
MenuItem item8, item9, item10;
sub.add(item8 = new MenuItem("First"));
sub.add(item9 = new MenuItem("Second"));
sub.add(item10 = new MenuItem("Third"));
edit.add(sub);
// these are checkable menu items
debug = new CheckboxMenuItem("Debug");
edit.add(debug);
test = new CheckboxMenuItem("Testing");
edit.add(test);
mbar.add(edit);
// create an object to handle action and item events
MyMenuHandler handler = new MyMenuHandler(this);
// register it to receive those events
```

```
item1.addActionListener(handler);
item2.addActionListener(handler);
item3.addActionListener(handler);
item4.addActionListener(handler);
item5.addActionListener(handler);
item6.addActionListener(handler);
item7.addActionListener(handler);
item8.addActionListener(handler);
item9.addActionListener(handler);
item10.addActionListener(handler);
debug.addItemListener(handler);
test.addItemListener(handler);
// create an object to handle window events
MyWindowAdapter adapter = new MyWindowAdapter(this);
// register it to receive those events
addWindowListener(adapter);
}
public void paint(Graphics g) {
g.drawString(msg, 10, 200);
if(debug.getState())
g.drawString("Debug is on.", 10, 220);
else
g.drawString("Debug is off.", 10, 220);
if(test.getState())
g.drawString("Testing is on.", 10, 240);
else
g.drawString("Testing is off.", 10, 240);
}
}
class MyWindowAdapter extends WindowAdapter {
MenuFrame menuFrame;
public MyWindowAdapter(MenuFrame menuFrame) {
this.menuFrame = menuFrame;
}
public void windowClosing(WindowEvent we) {
menuFrame.dispose();
}
}
class MyMenuHandler implements ActionListener, ItemListener {
MenuFrame menuFrame;
public MyMenuHandler(MenuFrame menuFrame) {
this.menuFrame = menuFrame;
}
// Handle action events.
public void actionPerformed(ActionEvent ae) {
String msg = "You selected ";
String arg = ae.getActionCommand();
// Activate a dialog box when New is selected.
```

```
if(arg.equals("New...")) {
msg += "New.";
SampleDialog d = new
SampleDialog(menuFrame, "New Dialog Box");
d.setVisible(true);
}
// Try defining other dialog boxes for these options.
else if(arg.equals("Open..."))
msg += "Open.";
else if(arg.equals("Close"))
msg += "Close.";
else if(arg.equals("Quit..."))
msg += "Quit.";
else if(arg.equals("Edit"))
msg += "Edit.";
else if(arg.equals("Cut"))
msg += "Cut.";
else if(arg.equals("Copy"))
msg += "Copy.";
else if(arg.equals("Paste"))
msg += "Paste.";
else if(arg.equals("First"))
msg += "First.";
else if(arg.equals("Second"))
msg += "Second.";
else if(arg.equals("Third"))
msg += "Third.";
else if(arg.equals("Debug"))
msg += "Debug.";
else if(arg.equals("Testing"))
msg += "Testing.";
menuFrame.msg = msg;
menuFrame.repaint();
}
public void itemStateChanged(ItemEvent ie) {
menuFrame.repaint();
}
}
// Create frame window.
public class DialogDemo extends Applet {
Frame f;
public void init() {
f = new MenuFrame("Menu Demo");
int width = Integer.parseInt(getParameter("width"));
int height = Integer.parseInt(getParameter("height"));
setSize(width, height);
f.setSize(width, height);
f.setVisible(true);
```

```
}
public void start() {
f.setVisible(true);
}
public void stop() {
f.setVisible(false);
}
}
```

TITLE: MenuBar Program in Java	SVRE/CSE/EXPT-OS/16/21 Date: 01-01- 2018

AIM:

16) Write a java program that creates menu which appears similar to the menu of notepad application of the Microsoft windows or any editor of your choice.

PROGRAM:-

```
// Illustrate menus.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="MenuDemo" width=250 height=250>
</applet>
*/
// Create a subclass of Frame.
class MenuFrame extends Frame {
String msg = "";
CheckboxMenuItem debug, test;
MenuFrame(String title) {
super(title);
// create menu bar and add it to frame
MenuBar mbar = new MenuBar();
setMenuBar(mbar);
// create the menu items
Menu file = new Menu("File");
MenuItem item1, item2, item3, item4, item5;

file.add(item1 = new MenuItem("New..."));
file.add(item2 = new MenuItem("Open..."));
file.add(item3 = new MenuItem("Close"));
file.add(item4 = new MenuItem("-"));
file.add(item5 = new MenuItem("Quit..."));
mbar.add(file);
Menu edit = new Menu("Edit");
MenuItem item6, item7, item8, item9;
edit.add(item6 = new MenuItem("Cut"));
edit.add(item7 = new MenuItem("Copy"));
edit.add(item8 = new MenuItem("Paste"));
```

```
edit.add(item9 = new MenuItem("-"));
Menu sub = new Menu("Special");
MenuItem item10, item11, item12;
sub.add(item10 = new MenuItem("First"));
sub.add(item11 = new MenuItem("Second"));
sub.add(item12 = new MenuItem("Third"));
edit.add(sub);
// these are checkable menu items
debug = new CheckboxMenuItem("Debug");
edit.add(debug);
test = new CheckboxMenuItem("Testing");
edit.add(test);
mbar.add(edit);
// create an object to handle action and item events
MyMenuHandler handler = new MyMenuHandler(this);
// register it to receive those events
item1.addActionListener(handler);
item2.addActionListener(handler);
item3.addActionListener(handler);
item4.addActionListener(handler);
item5.addActionListener(handler);
item6.addActionListener(handler);
item7.addActionListener(handler);
item8.addActionListener(handler);
item9.addActionListener(handler);
item10.addActionListener(handler);
item11.addActionListener(handler);
item12.addActionListener(handler);
debug.addItemListener(handler);
test.addItemListener(handler);
// create an object to handle window events
MyWindowAdapter adapter = new MyWindowAdapter(this);
// register it to receive those events
addWindowListener(adapter);
}
public void paint(Graphics g) {
g.drawString(msg, 10, 200);
if(debug.getState())
g.drawString("Debug is on.", 10, 220);
else
g.drawString("Debug is off.", 10, 220);
if(test.getState())
g.drawString("Testing is on.", 10, 240);
else
g.drawString("Testing is off.", 10, 240);
}
}
class MyWindowAdapter extends WindowAdapter {
```

```
MenuFrame menuFrame;
public MyWindowAdapter(MenuFrame menuFrame) {
this.menuFrame = menuFrame;
}
public void windowClosing(WindowEvent we) {
menuFrame.setVisible(false);
}
}
class MyMenuHandler implements ActionListener, ItemListener {
MenuFrame menuFrame;
public MyMenuHandler(MenuFrame menuFrame) {
this.menuFrame = menuFrame;
}
// Handle action events.
public void actionPerformed(ActionEvent ae) {
String msg = "You selected ";
String arg = ae.getActionCommand();
if(arg.equals("New..."))
msg += "New.";
else if(arg.equals("Open..."))
msg += "Open.";
else if(arg.equals("Close"))
msg += "Close.";
else if(arg.equals("Quit..."))
msg += "Quit.";
else if(arg.equals("Edit"))
msg += "Edit.";
else if(arg.equals("Cut"))
msg += "Cut.";
else if(arg.equals("Copy"))
msg += "Copy.";
else if(arg.equals("Paste"))
msg += "Paste.";
else if(arg.equals("First"))
msg += "First.";
else if(arg.equals("Second"))
msg += "Second.";
else if(arg.equals("Third"))
msg += "Third.";
else if(arg.equals("Debug"))
msg += "Debug.";
else if(arg.equals("Testing"))
msg += "Testing.";
menuFrame.msg = msg;
menuFrame.repaint();
}
// Handle item events.
public void itemStateChanged(ItemEvent ie) {
```

```
menuFrame.repaint();
}
}
// Create frame window.
public class MenuDemo extends Applet {
Frame f;
public void init() {
f = new MenuFrame("Menu Demo");
int width = Integer.parseInt(getParameter("width"));
int height = Integer.parseInt(getParameter("height"));
setSize(new Dimension(width, height));
f.setSize(width, height);
f.setVisible(true);
}
public void start() {
f.setVisible(true);
}
public void stop() {
f.setVisible(false);
}
}
```

TITLE: producer consumer problem in Java SVRE/CSE/EXPT-OS/17/21

AIM:

17)Write a Java program that correctly implements producer consumer problem using the concept of inter thread communication

PROGRAM:

```
// Java program to implement solution of producer
// consumer problem.
import java.util.LinkedList;

public class Threadexample
{
   public static void main(String[] args)
                    throws InterruptedException
   {
      // Object of a class that has both produce()
      // and consume() methods
      final PC pc = new PC();

      // Create producer thread
      Thread t1 = new Thread(new Runnable()
      {
         @Override
         public void run()


         {
```

```
            try
            {
                pc.produce();
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    });

    // Create consumer thread
    Thread t2 = new Thread(new Runnable()
    {
        @Override
        public void run()
        {
            try
            {
                pc.consume();
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    });

    // Start both threads
    t1.start();
    t2.start();

    // t1 finishes before t2
    t1.join();
    t2.join();
}

// This class has a list, producer (adds items to list
// and consumber (removes items).
public static class PC
{
    // Create a list shared by producer and consumer
    // Size of list is 2.
    LinkedList<Integer> list = new LinkedList<>();
    int capacity = 2;

    // Function called by producer thread
    public void produce() throws InterruptedException
```

```
{
   int value = 0;
   while (true)
   {
      synchronized (this)
      {
         // producer thread waits while list
         // is full
         while (list.size()==capacity)
            wait();

         System.out.println("Producer produced-"
                              + value);

         // to insert the jobs in the list
         list.add(value++);

         // notifies the consumer thread that
         // now it can start consuming
         notify();

         // makes the working of program easier
         // to  understand
         Thread.sleep(1000);
      }
   }
}

// Function called by consumer thread
public void consume() throws InterruptedException
{
   while (true)
   {
      synchronized (this)
      {
         // consumer thread waits while list
         // is empty
         while (list.size()==0)
            wait();

         //to retrive the ifrst job in the list
         int val = list.removeFirst();

         System.out.println("Consumer consumed-"  + val);

         // Wake up producer thread
         notify();
```

```
            // and sleep
            Thread.sleep(1000);
          }
        }
      }
    }
}
```

Output:

```
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
```

TITLE: find and replace pattern in a given file in Java	SVRE/CSE/EXPT-OS/18/21 Date: 01-01- 2018

AIM:

18) Write a java program to find and replace pattern in a given file.

Program:

```
import java.io.*;

public class BTest
  {
   public static void main(String args[])
     {
     try
       {
       File file = new File("file.txt");
       BufferedReader reader = new BufferedReader(new FileReader(file));
       String line = "", oldtext = "";
       while((line = reader.readLine()) != null)
         {
         oldtext += line + "\r\n";
       }
       reader.close();
       // replace a word in a file
       //String newtext = oldtext.replaceAll("drink", "Love");

       //To replace a line in a file
       String newtext = oldtext.replaceAll("This is test string 20000", "blah blah blah");

       FileWriter writer = new FileWriter("file.txt");
       writer.write(newtext);writer.close();
     }
     catch (IOException ioe)
       {
```

```
            ioe.printStackTrace();
        }
    }
}
```

OutPut:
I drink Java
I sleep Java

This is test string 1
This is test string 20000

TITLE: Use inheritance to create an exception in Java SVRE/CSE/EXPT-OS/19/21

AIM:-
19)Use inheritance to create an exception super class called EexceptionA and exception sub class ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a java program to demonstrate that the catch block for type ExceptionA catches exception of type ExceptionB and ExceptionC.

Program:

```
class ExceptionsA{
  static void genException(){
    int num[]= {4,8,16,32,64,128,256};


    int div[]= {2,2,4,4,0,8};

    for(int i=0; i<num.length; i++){
      System.out.println(num[i] + "/" +
          div[i] + "is" +
          num[i]/div[i]);
    }
  }
}
class ExceptionsB extends ExceptionsA{
  ExceptionsB(){
    String zero;
    zero = ("Division by zero not allowed");
  }
}
class ExceptionsC extends ExceptionsB{
  ExceptionsC(){
    String array;
    array = ("Divisor element not found");
  }
}
class ExceptionTest{
  public static void main(String[] args){
    try {
      ExceptionsA.genException();
```

```
    }
    catch (ArithmeticException zero){
      System.out.printf("%s",zero);
    }
    catch (ArrayIndexOutOfBoundsException array){
      System.out.printf("%s",array);
    }//end catch
  }//end class
}//end method
```

Output:-

```
4/2is2
8/2is4
16/4is4
32/4is8
java.lang.ArithmeticException: / by zero
```

TITLE: java program to create a URL Connection in Java	SVRE/CSE/EXPT-OS/20/21

AIM:-

20)write a java program to create a URLConnection and use it to examine the documents properties and content

Program:-

```
// Demonstrate URLConnection.
import java.net.*;
import java.io.*;
import java.util.Date;
class UCDemo
{
public static void main(String args[]) throws Exception {
int c;
URL hp = new URL("http://www.java.com");
URLConnection hpCon = hp.openConnection();
System.out.println("Date: " + new Date(hpCon.getDate()));
System.out.println("Content-Type: " +
hpCon.getContentType());
System.out.println("Expires: " + hpCon.getExpiration());
System.out.println("Last-Modified: " +
new Date(hpCon.getLastModified()));
int len = hpCon.getContentLength();
System.out.println("Content-Length: " + len);
if (len > 0) {
System.out.println("=== Content ===");
InputStream input = hpCon.getInputStream();
int i = len;
while (((c = input.read()) != -1) && (—i > 0)) {
System.out.print((char) c);
}
input.close();
```

```
} else {
System.out.println("No Content Available");
}
}
}
```

TITLE: java program to create a URL Connection in Java SVRE/CSE/EXPT-OS/21/21

AIM:-Write a Java Program which uses TCP/IP and Datagrams to communicate client and server.

Program:

```
import java.io.*;
import java.net.*;

public class Client {
	public static void main(String[] args) {
		try {
			Socket s = new Socket("localhost", 6666);
			DataOutputStream dout = new DataOutputStream(s.getOutputStream());
			dout.writeUTF("Hello Server");
			dout.flush();
			dout.close();
			s.close();
		} catch (Exception e) {
			System.out.println(e);
		}
	}
}

import java.io.*;
import java.net.*;
public class Server {
	public static void main(String[] args) {
		try {
			ServerSocket ss = new ServerSocket(6666);
			Socket s = ss.accept();// establishes connection
			DataInputStream dis = new DataInputStream(s.getInputStream());
			String str = (String) dis.readUTF();
			System.out.println("message= " + str);
			ss.close();
		} catch (Exception e) {
			System.out.println(e);
		}
```

```
        }
}
```