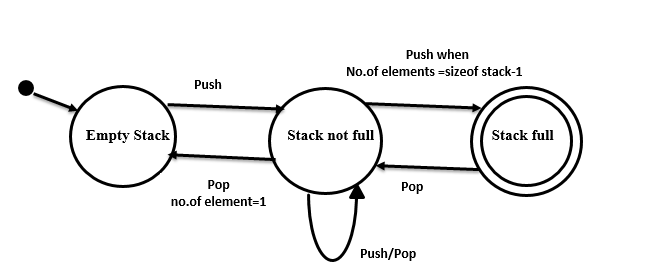
**Termwork : 1**

Problem Statement:

Using State Diagram Design and Describe the behaviour of STACK which contains maximum of FOUR integer elements. Implement the above design in JAVA Programming Language. Design the TEST­DRIVER class to include minimum number of TEST CASES to test the complete features of STACK class designed.

Design Aspects/State Diagram:



Data Structure used:

Stack is implemented using array of fixed size.

Stack data structure provides the following operations:

Pop: Remove element from the stack. This operation can be operated on stack only if the stack is not empty. As stack data structure works as LIFO pop operations always pops the top of the stack.

Push: Used to add the element to the top of the stack. Push operation fails if the stack is full.

Display: Used to display the content of the stack.

isFull() : Used to check whether the stack is full or not. Push calls isFull() method before pushing the element into the stack.

isEmpty(): Used to check whether the stack is empty or not. Display and Pop methods call isEmpty() method before popping or displaying the elements.

In this termwork size of stack is set to to four as follows.

int stk\_elements[]=new int[4];

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Case  Input | Expected  Output | Observed  Output |
| Test Case: 1 | Push operation when stack is empty | push(2) | Pushed successfully | Pushed successfully |
| Test Case: 2 | Pop operation when stack is not empty | pop() i.e., pop the top of the stack | Popped successfully | Popped successfully |
| Test Case: 3 | Display operation when stack is not empty | display() called | Elements are displayed | Elements are displayed |
| Test Case: 4 | Push operation when stack is full | push(4) | Cannot Push the element | Push Failed  Stack Full |
| Test Case: 5 | Pop operation when stack is empty | pop() i.e., pop the top of the stack | No elements to pop | Pop failed |
| Test Case: 6 | Display operation when stack is empty | display() called | No elements to display | Display failed |

JAVA Code:

package termworks;

import java.util.Scanner;

class StackOperations

{

@SuppressWarnings("unused")

private int top=-1,x;

int stk\_elements[]=new int[4];

boolean isEmpty()

{

if(top==-1)

return(true);

return(false);

}

boolean isFull()

{

if(top>=3)

return(true);

return(false);

}

int push(int element)

{

if(isFull())

{

return(4);

}

else if(top==0 && top<=3){

stk\_elements[++top]=element;

return(1);

}

else

{

stk\_elements[++top]=element;

return(1);

}

}//push()

int pop()

{

if(isEmpty())

{

return(5);

}

else

{

x=stk\_elements[top--];

return(2);

}

}

int display(){

if(isEmpty())

return(6);

else{

System.out.println("Stack content");

for(int i=top;i>=0;i--){

System.out.println(stk\_elements[i]);

}

return(3);

}

}

}//display()

class TestDriver {

public void checkTestCase(int status){

switch(status){

case 1:System.out.println("Test case (i) sucessful");

break;

case 2:System.out.println("Test case (ii) sucessful");

break;

case 3:System.out.println("Test case (iii) sucessful");

break;

case 4:System.out.println("Test case (iv) sucessful");

break;

case 5:System.out.println("Test case (v) sucessful");

break;

case 6:System.out.println("Test case (vi) sucessful");

break;

default:System.out.println("In valid test-case id");

}

}

}

public class UseOfStackOp{

public static void main(String args[])

{

StackOperations s=new StackOperations();

TestDriver td=new TestDriver();

Scanner sc = new Scanner(System.in);

int choice,item,choice1;

System.out.println("Test-case (i) for push operation successfull");

System.out.println("Test-case (ii) for pop operation successfull");

System.out.println("Test-case (iii)for successfull display operation ");

System.out.println("Test-case (iv) for push operation failed");

System.out.println("Test-case (v) for pop operation failed");

System.out.println("Test-case (vi)for display operation with stack empty ");

do

{

System.out.println("\*\*\*\*\*\*\* MENU \*\*\*\*\*\*");

System.out.println("1.PUSH");

System.out.println("2.POP");

System.out.println("3.DISPLAY");

System.out.println("4.Quit");

System.out.print(" Enter Your Options : ");

choice=sc.nextInt();

switch (choice) {

case 1:

System.out.println("Enter value to push");

item=sc.nextInt();

choice1=s.push(item);

switch(choice1){

case 1:System.out.println("Push Operation Successfull");

td.checkTestCase(1);

break;

case 4:System.out.println("Push Operation failed");

td.checkTestCase(4);

break;

}

break;

case 2:

choice1= s.pop() ;

switch(choice1){

case 2:System.out.println("Pop Operation Successfull");

td.checkTestCase(2);

break;

case 5:System.out.println("Pop Operation failed");

td.checkTestCase(5);

break;

}

break;

case 3:

choice1=s.display();

switch(choice1){

case 3:System.out.println("Display Operation Successfull");

td.checkTestCase(3);

break;

case 6:System.out.println("Display operation failed as stack is empty");

td.checkTestCase(6);

break;

}

break;

case 4: System.out.println("End of the program");

break;

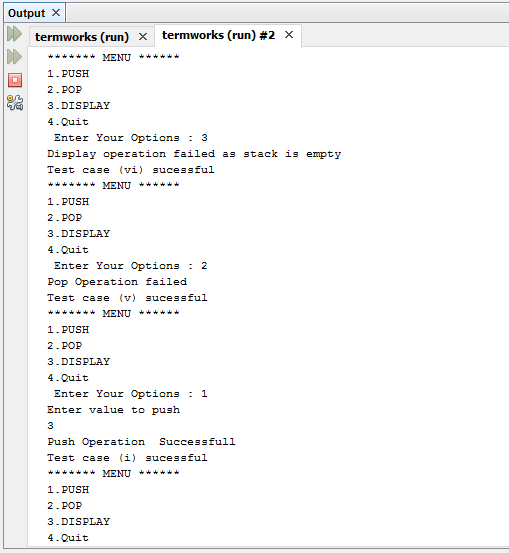
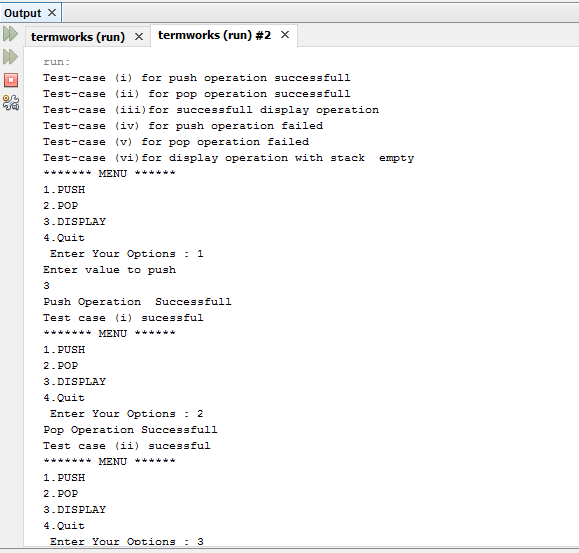
}

}while(choice!=4);

}

}

Output:



**Termwork : 2**

Problem Statement:

Create an appropriate GUI which allows the user to select an item from the menu. When draw menu item is selected, draws the selected shape(Allowed shapes are: Circle, Rectangle & Triangle) in drawing area by getting appropriate dimensions of the selected shape from the user through key Board entry, using the concept of ABSTRACT CLASS, INHERITANCE and DYNAMIC DISPATCH features of JAVA Programming Language. Code must be robust for all possible erroneous input conditions, displaying appropriate error messages in message window specially designed for them.

Design Aspects:

An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

There are some important differences between an applet and a standalone Java application, including the following −

An applet is a Java class that extends the java.applet.Applet class.

A main() method is not invoked on an applet, and an applet class will not define main().

Applets are designed to be embedded within an HTML page.

When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.

A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment

Java code:

package termworks;

import java.awt.FlowLayout;

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Menu;

import java.awt.MenuBar;

import java.awt.MenuItem;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextField;

public class Shapes extends Frame implements ActionListener {

MenuBar mbar;

Menu shapes;

MenuItem rect,tri,cir,q;

int x,y;

int xp[] = new int[2];

int yp[] = new int[2];

int zp[] = new int[2];

public Shapes(){

setTitle("Draw Shapes"); // Set the title

setSize(400,400); // Set size to the frame

setLayout(new FlowLayout()); // Set the layout

setVisible(true); // Make the frame visible

setLocationRelativeTo(null); // Center the frame

// Create the menu bar

mbar=new MenuBar();

// Create the menu

shapes=new Menu("Shapes");

// Create MenuItems

cir=new MenuItem("Circle");

rect=new MenuItem("Rectangle");

tri=new MenuItem("Triangle");

q=new MenuItem("Exit");

// Attach menu items to menu

shapes.add(cir);

shapes.add(rect);

shapes.add(tri);

shapes.add(q);

// Attach menu to menu bar

mbar.add(shapes);

// Set menu bar to the frame

setMenuBar(mbar);

//adding action listner

cir.addActionListener(this);

rect.addActionListener(this);

tri.addActionListener(this);

q.addActionListener(this);

}

public static void main(String args[])

{

new Shapes();

}

@Override

public void actionPerformed(ActionEvent e) {

Integer index=0;

JPanel area = new JPanel();

JTextField jt = new JTextField(5);

JTextField jt1 = new JTextField(5);

JTextField jt2 = new JTextField(5);

JTextField jt3 = new JTextField(5);

JTextField jt4 = new JTextField(5);

JTextField jt5 = new JTextField(5);

// TODO Auto-generated method stub

String cmd=e.getActionCommand();

if(cmd.equals("Circle"))

{

area.add(jt);

index=2;

}

else if(cmd.equals("Rectangle"))

{

index=1;

area.add(jt);

area.add(jt1);

}

else if(cmd.equals("Triangle"))

{

area.add(jt);

area.add(jt1);

area.add(jt2);

area.add(jt3);

area.add(jt4);

area.add(jt5);

index=3;

}

else if(cmd.equals("Exit"))

{

setVisible(false);

}

int result = JOptionPane.showConfirmDialog(null, area, "Enter the dimenisions", JOptionPane.OK\_CANCEL\_OPTION);

if (result == JOptionPane.OK\_OPTION && index==1) {

x= Integer.parseInt(jt.getText());

y= Integer.parseInt(jt1.getText());

}

else if (result == JOptionPane.OK\_OPTION && index==2) {

x= Integer.parseInt(jt.getText());

}else if(result == JOptionPane.OK\_OPTION && index==3){

xp[0] = Integer.parseInt(jt.getText());

xp[1] = Integer.parseInt(jt1.getText());

yp[0] = Integer.parseInt(jt2.getText());

yp[1] = Integer.parseInt(jt3.getText());

zp[0] = Integer.parseInt(jt4.getText());

zp[1] = Integer.parseInt(jt5.getText());

}

drawShape(index);

}

int flag=0;

private void drawShape(Integer index) {

// TODO Auto-generated method stub

flag=index;

repaint();

}

public void paint(Graphics g){

//super.paint(g);

switch (flag) {

case 1:g.drawRect(100, 100, x,y);

break;

case 2:g.drawOval(100,100,x,x);

– break;

case 3:g.drawLine(xp[0],xp[1],yp[0],yp[1]);

g.drawLine(yp[0],yp[1],zp[0],zp[1]);

g.drawLine(zp[0],zp[1],xp[0],xp[1]);

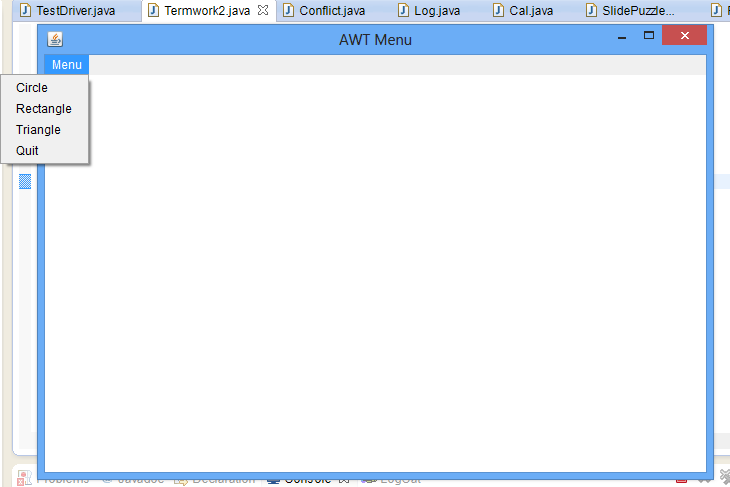
break;

}

}

}

Output:



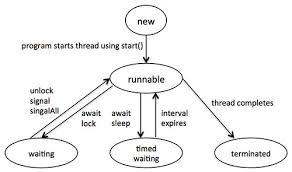
**Termwork : 3**

Problem Statement:

Write Java program to simulate LOST UPDATE or INCONSISTENT READ Transaction issues of database using MULTITHREADING features of JAVA and also Write java program to control the above concurrency issue. Output of the program to be displayed on the screen as well as to be written in a file of user choice.

Design Aspects:

Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs. Thread works as follows.



Thread Synchronization:

Java programming language provides a very handy way of creating threads and synchronizing their task by using synchronized blocks. We mention shared resources within this block. Following is the general form of the synchronized statement −

Syntax

synchronized(objectidentifier) {

// Access shared variables and other shared resources

}

Here, the objectidentifier is a reference to an object whose lock associates with the monitor that the synchronized statement represents.

To demonstrate the lost update problem synchronized block is not used. If there is no synchronization between the threads the update made by thread 1 one is lost as thread 2 overwrite the update made by thread1.

Without Synchronized:

|  |  |  |  |
| --- | --- | --- | --- |
| Thread Scheduling | Update Made | Result | Expected Result |
| Thread :1 | A=200;  A=A+10; | 210 | 210 |
| Thread:2 | A=210  A=A+20; | 220  Update done to A=210 by  Thread: 1 is lost.  So Thread: 2 reads the value of A=200 | 230 |

Java code:

package termworks;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

class T1 extends Thread{

public void run(){

Term3.update(10);

System.out.println("\nT1 thread Done");

}

}

class T2 extends Thread{

public void run(){

Term3.update(20);

System.out.println("\nT2 thread Done");

}

}

public class Term3{

static int a=200;

public static void main(String arg[]) throws IOException

{

File f=null;

try{

f=new File("C:\\Users\\Acer\\Documents\\NetBeansProjects\\termworks\\src\\termworks\\out.dat");

}

catch(Exception e){

System.out.println("File not found");

}

//FileWriter fw=new FileWriter(f);

T1 t1=new T1();

T2 t2=new T2();

t1.start();

t2.start();

try{

t1.join();

t2.join();

}

catch(InterruptedException e){}

System.out.println("\nValue of a:"+a);

BufferedWriter out1 = new BufferedWriter(new FileWriter(f));

out1.write("\nValue of a:"+a);

out1.close();

//fw.write(" "+a);

}

static void update(int value){

int x;

x=a;

try{

Thread.sleep((int)(Math.random()\*1));

}catch(InterruptedException e){}

x=x+value;

try{

Thread.sleep((int)(Math.random()\*1));

}catch(InterruptedException e){}

a=x;

}

}

With Synchronized:

|  |  |  |  |
| --- | --- | --- | --- |
| Thread Scheduling | Update Made | Result | Expected Result |
| Thread :1 | Input: A=200;  A=A+10; | 210 | 210 |
| Thread:2 | Input: A=210  A=A+20; | 230  Update done to A=210 by Thread:1 is retained.  So Thread: 2 reads the value of A=210 | 230 |

Java Code:

package termworks;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

class Thread1 extends Thread{

public void run(){

Tw3.update(10);

System.out.println("\nT1 thread Done");

}

}

class Thread2 extends Thread{

public void run(){

Tw3.update(20);

System.out.println("\nT2 thread Done");

}

}

public class Tw3{

static int a=200;

public static void main(String arg[]) throws IOException

{

File f=null;

try{

f=new File("C:\\Users\\Acer\\Documents\\NetBeansProjects\\termworks\\src\\termworks\\out.dat");

}

catch(Exception e){

System.out.println("File not found");

}

//FileWriter fw=new FileWriter(f);

Thread1 t1=new Thread1();

Thread2 t2=new Thread2();

t1.start();

t2.start();

try{

t1.join();

t2.join();

}

catch(InterruptedException e){}

System.out.println("\nValue of a:"+a);

BufferedWriter out1 = new BufferedWriter(new FileWriter(f));

out1.write("\nValue of a:"+a);

out1.close();

//fw.write(" "+a);

}

static synchronized void update(int value){

int x;

x=a;

try{

Thread.sleep((int)(Math.random()\*1));

}catch(InterruptedException e){}

x=x+value;

try{

Thread.sleep((int)(Math.random()\*1));

}catch(InterruptedException e){}

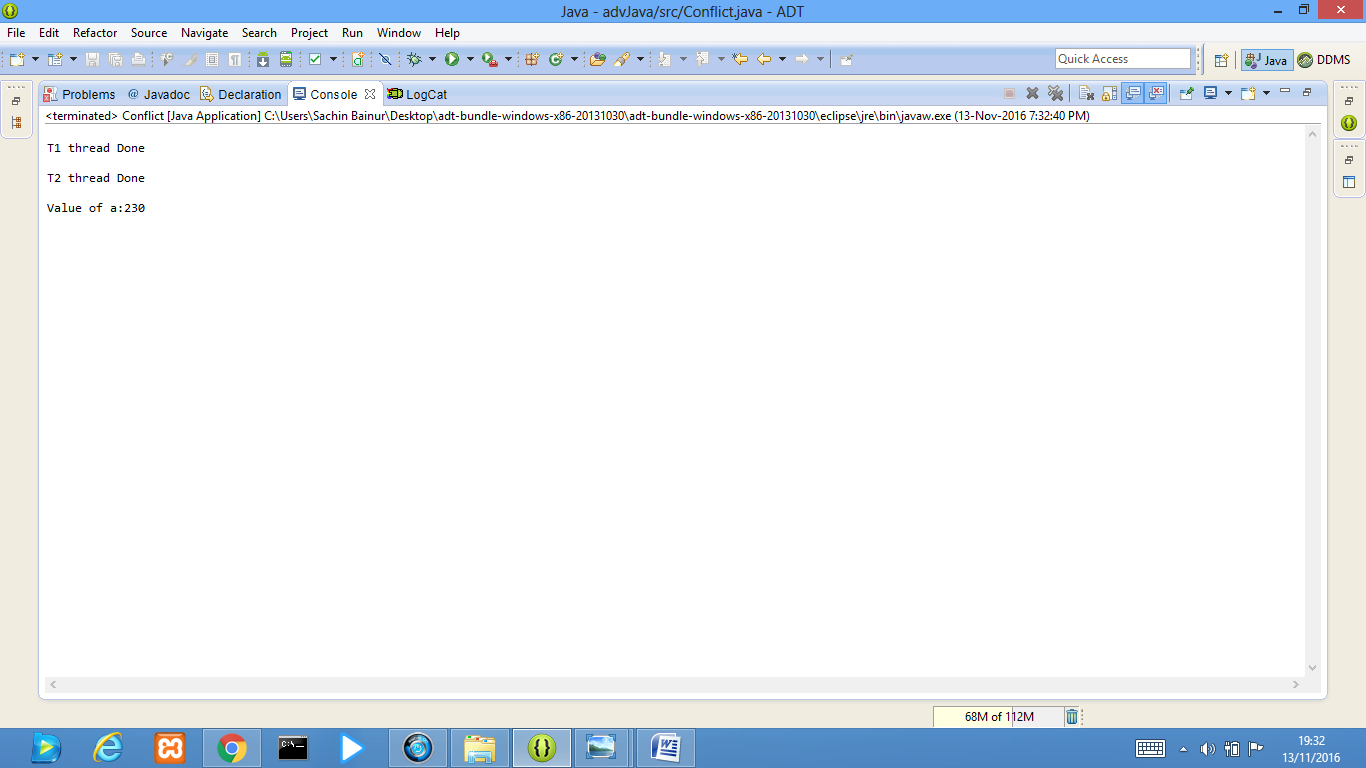
a=x;

}

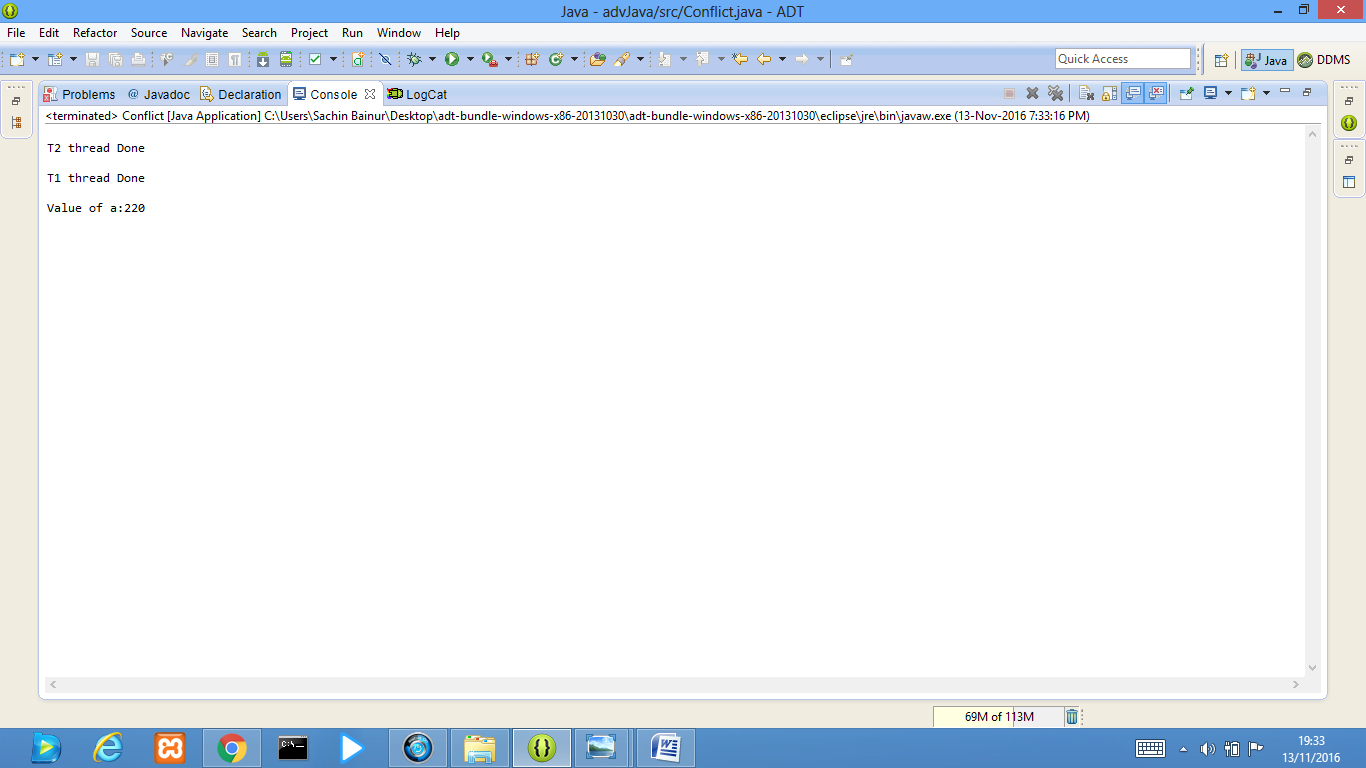
}

Output:

With Synchronisation:

****

Without Synchronisation:

****

**Termwork : 4**

Problem Statement:

|  |  |
| --- | --- |
|  | Design and Implement an APPLET for any computer game of your  choice. Store the user name and the score of each game session in the database (MySQL) |

Java code:

*//SlidePuzzle class*

import java.awt.Button;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Frame;

import java.awt.Label;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

@SuppressWarnings("serial")

public class SlidePuzzle extends Frame implements ActionListener

{ Label l1=new Label("User Name:");

Label l2=new Label("Password:");

Label l3=new Label("\*Enter login details");

Label l4=new Label("\*Arrange the grid in asscending order starting from ONE");

Label l5=new Label("\*Complete the task in minimum time");

Label l6=new Label(" ");

TextField t1=new TextField();

TextField t2=new TextField();

Button b= new Button("Submit");

//Image tImage = new ImageIcon("login.png").getImage();

public SlidePuzzle()

{

setBackground(Color.GRAY);

add(l1);

add(t1);

add(l2);

add(t2);

add(b);

add(l3);

add(l4);

add(l5);

add(l6);

l1.setBounds(20,45,70,20);

t1.setBounds(180,45,200,20);

l2.setBounds(20,95,70,20);

t2.setBounds(180,95,200,20);

b.setBounds(310,145,70,20);

l3.setBounds(20,200,200,20);

l4.setBounds(20,220,400,20);

l5.setBounds(20,240,400,20);

b.addActionListener(this);

t2.setEchoChar('\*');

addWindowListener(new mwa());

}

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==b && t1.getText().compareTo("")!=0){

// l3.setText("Welcome "+t1.getText());

// setVisible(false);

Puzzle p =new Puzzle();

p.setUser(t1.getText(), t2.getText());

p.setTitle("Welcome "+t1.getText());

t1.setText("");

t2.setText("");

}

}

public static void main(String s[])

{ SlidePuzzle l=new SlidePuzzle();

l.setSize(new Dimension(600,600));

l.setTitle("Login");

l.setVisible(true);

}

}

class mwa extends WindowAdapter

{ public mwa(){}

public void windowClosing(WindowEvent e)

{ System.exit(0);

}

}

*//Puzzle class*

**import** java.awt.Button;

**import** java.awt.Frame;

**import** java.awt.Graphics;

**import** java.awt.GridLayout;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.Statement;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.GregorianCalendar;

**import** javax.swing.ImageIcon;

**import** javax.swing.JOptionPane;

@SuppressWarnings("serial")

**public** **class** Puzzle **extends** Frame **implements** ActionListener{

Button b1,b2,b3,b4,b5,b6,b7,b8,b9,x,y;

String u,pwd,ontime,offtime;

**int** sthr,stmin,stsec,edhr,edmin,edsec;

Image tileImage = **new** ImageIcon("bck.jpg").getImage();

String url = "jdbc:mysql://localhost:3306/";

String user = "root";

String password = "";

**void** setUser(String u, String p)

{

**this**.u=u;

pwd=p;

//For login session

Date d=**new** Date();

ontime=d.toString();

}

**public** **void** paint(Graphics g){

g.drawImage(tileImage, 0, 0, **this**);

}

Puzzle(){

//for game session

GregorianCalendar date = **new** GregorianCalendar();

stsec = date.get(Calendar.*SECOND*);

stmin = date.get(Calendar.*MINUTE*);

sthr = date.get(Calendar.*HOUR*);

setResizable(**false**);

setLayout(**new** GridLayout(4,3));

// super("Puzzle - JavaTpoint");

b1=**new** Button("7");

b1.setBounds(50,100,40,40);

b2=**new** Button("2");

b2.setBounds(100,100,40,40);

b3=**new** Button("3");

b3.setBounds(150,100,40,40);

b4=**new** Button("4");

b4.setBounds(50,150,40,40);

b5=**new** Button("");

b5.setBounds(100,150,40,40);

b6=**new** Button("6");

b6.setBounds(150,150,40,40);

b7=**new** Button("1");

b7.setBounds(50,200,40,40);

b8=**new** Button("5");

b8.setBounds(100,200,40,40);

b9=**new** Button("8");

b9.setBounds(150,200,40,40);

x=**new** Button("Exit");

x.setBounds(300, 100, 50, 30);

y=**new** Button("Reset");

y.setBounds(200, 100, 70, 30);

b1.addActionListener(**this**);

b2.addActionListener(**this**);

b3.addActionListener(**this**);

b4.addActionListener(**this**);

b5.addActionListener(**this**);

b6.addActionListener(**this**);

b7.addActionListener(**this**);

b8.addActionListener(**this**);

b9.addActionListener(**this**);

x.addActionListener(**this**);

y.addActionListener(**this**);

add(b1);add(b2);add(b3);add(b4);add(b5);add(b6);add(b7);add(b8);add(b9); add(x) ;add(y);

setSize(400,350);

//setLayout(null);

setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent e){

**if**(e.getSource()==b1){

String label=b1.getLabel();

**if**(b2.getLabel().equals("")){

b2.setLabel(label);

b1.setLabel("");

}

**if**(b4.getLabel().equals("")){

b4.setLabel(label);

b1.setLabel("");

}

}

**if**(e.getSource()==b2){

String label=b2.getLabel();

**if**(b1.getLabel().equals("")){

b1.setLabel(label);

b2.setLabel("");

}

**if**(b3.getLabel().equals("")){

b3.setLabel(label);

b2.setLabel("");

}

**if**(b5.getLabel().equals("")){

b5.setLabel(label);

b2.setLabel("");

}

}

**if**(e.getSource()==b3){

String label=b3.getLabel();

**if**(b2.getLabel().equals("")){

b2.setLabel(label);

b3.setLabel("");

}

**if**(b6.getLabel().equals("")){

b6.setLabel(label);

b3.setLabel("");

}

}

**if**(e.getSource()==b4){

String label=b4.getLabel();

**if**(b1.getLabel().equals("")){

b1.setLabel(label);

b4.setLabel("");

}

**if**(b7.getLabel().equals("")){

b7.setLabel(label);

b4.setLabel("");

}

**if**(b5.getLabel().equals("")){

b5.setLabel(label);

b4.setLabel("");

}

}

**if**(e.getSource()==b5){

String label=b5.getLabel();

**if**(b2.getLabel().equals("")){

b2.setLabel(label);

b5.setLabel("");

}

**if**(b6.getLabel().equals("")){

b6.setLabel(label);

b5.setLabel("");

}

**if**(b4.getLabel().equals("")){

b4.setLabel(label);

b5.setLabel("");

}

**if**(b8.getLabel().equals("")){

b8.setLabel(label);

b5.setLabel("");

}

}

**if**(e.getSource()==b6){

String label=b6.getLabel();

**if**(b9.getLabel().equals("")){

b9.setLabel(label);

b6.setLabel("");

}

**if**(b3.getLabel().equals("")){

b3.setLabel(label);

b6.setLabel("");

}

**if**(b5.getLabel().equals("")){

b5.setLabel(label);

b6.setLabel("");

}

}

**if**(e.getSource()==b7){

String label=b7.getLabel();

**if**(b4.getLabel().equals("")){

b4.setLabel(label);

b7.setLabel("");

}

**if**(b8.getLabel().equals("")){

b8.setLabel(label);

b7.setLabel("");

}

}

**if**(e.getSource()==b8){

String label=b8.getLabel();

**if**(b9.getLabel().equals("")){

b9.setLabel(label);

b8.setLabel("");

}

**if**(b7.getLabel().equals("")){

b7.setLabel(label);

b8.setLabel("");

}

**if**(b5.getLabel().equals("")){

b5.setLabel(label);

b8.setLabel("");

}

}

**if**(e.getSource()==b9){

String label=b9.getLabel();

**if**(b6.getLabel().equals("")){

b6.setLabel(label);

b9.setLabel("");

}

**if**(b8.getLabel().equals("")){

b8.setLabel(label);

b9.setLabel("");

}

}

**if**(e.getSource()==x){

setVisible(**false**);

Store(0);

}

**if**(e.getSource()==y)

{

setVisible(**false**);

**new** Puzzle();

}

//congrats code

**if**(b1.getLabel().equals("1")&&b2.getLabel().equals("2")&&b3.getLabel()

.equals("3")&&b4.getLabel().equals("4")&&b5.getLabel().equals("5")

&&b6.getLabel().equals("6")&&b7.getLabel().equals("7")&&b8.getLabel()

.equals("8")&&b9.getLabel().equals("")){

JOptionPane.*showMessageDialog*(**this**,"Congratulations! You won. Contact Admin to know your score!!");

//for game session

Store(1);

setVisible(**false**);

}

}

**public** **void** Store(**int** flag) {

Date d1=**new** Date();

offtime=d1.toString();

String score="";

**if**(flag==1){

GregorianCalendar date = **new** GregorianCalendar();

edsec = date.get(Calendar.*SECOND*);

edmin = date.get(Calendar.*MINUTE*);

edhr = date.get(Calendar.*HOUR*);

**int** hr=Math.*abs*(sthr-edhr);

**int** min=Math.*abs*(stmin-edmin);

**int** sec=Math.*abs*(stsec-edsec);

score=Integer.*toString*(hr)+":"+Integer.*toString*(min)+":"+Integer.*toString*(sec);

}

**else**{

score="Incomplete";

}

**try**

{

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection con = DriverManager.*getConnection*(url, user, password);

Statement stt = con.createStatement();

stt.execute("USE session");

System.*out*.print(ontime);

stt.execute("INSERT INTO game (User, Score , Login , Logoff) VALUES ('"+u+"', '"+score+"','"+ontime+"','"+offtime+"')");

con.close();

}

**catch** (Exception e2)

{

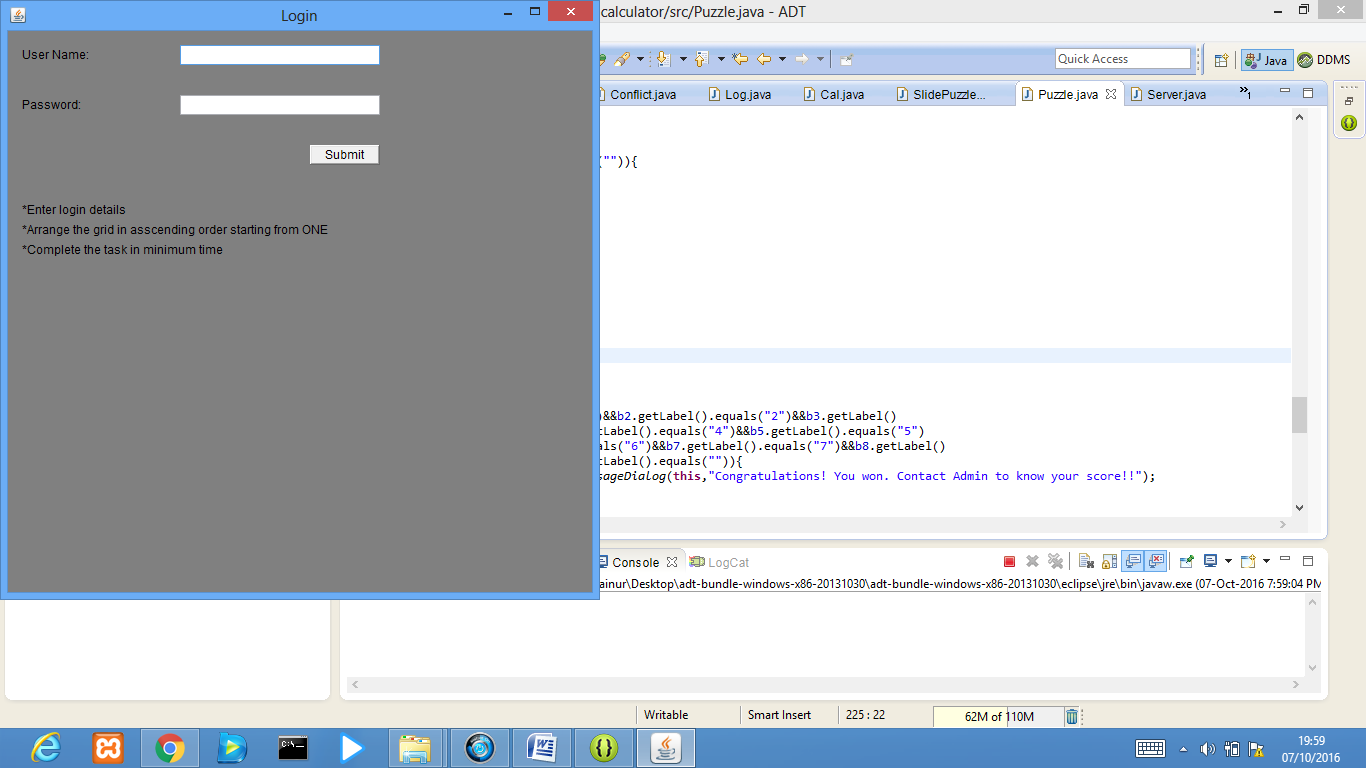
e2.printStackTrace();

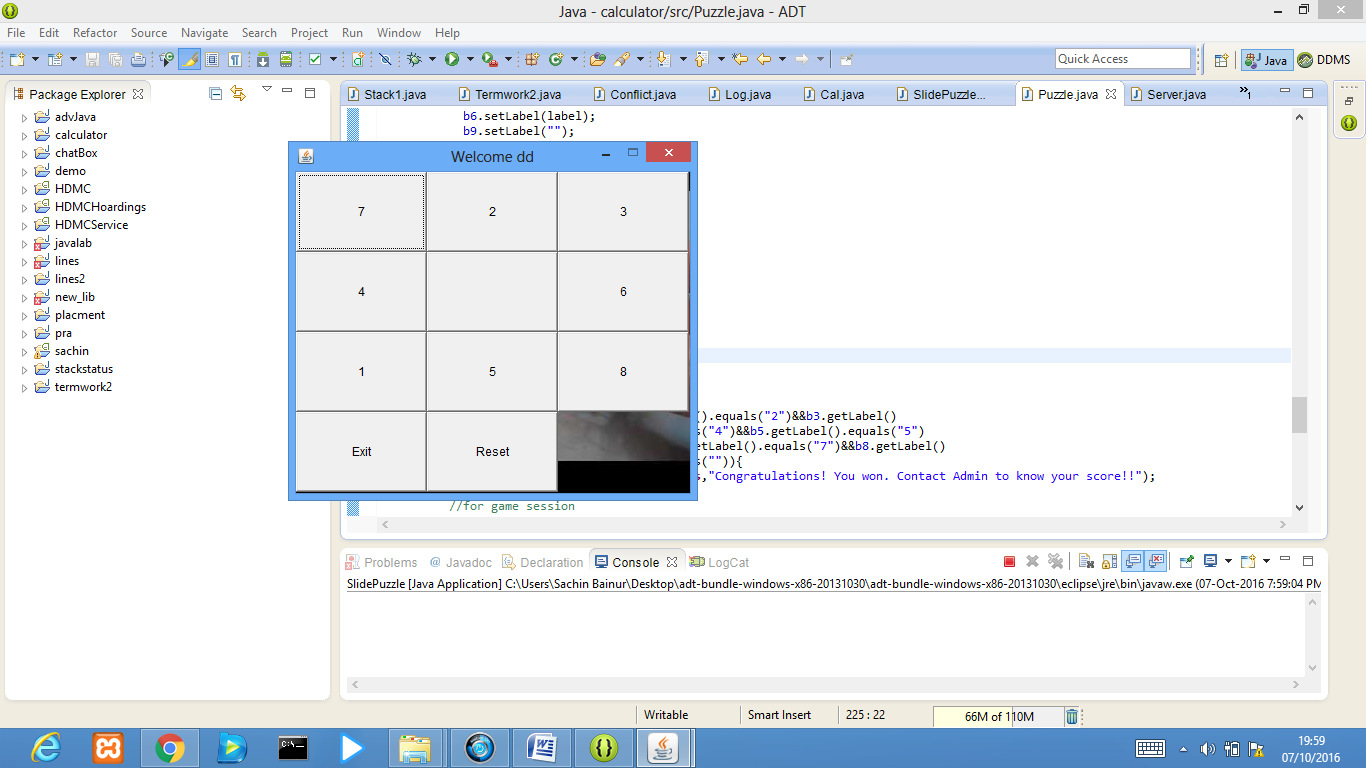
}

}

}

OUTPUT:





**Termwork : 5**

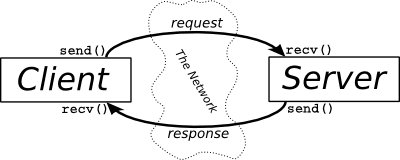
Problem Statement:

Write Java program to implement 1-1 chatting (text) using Networking features.

Design Aspects/State Diagram:

The main aim of this Program is to create a simple server that can add two given numbers. This tutorial has three sections as follows:

* Creating and Running a Server
* Connecting to the server using ip address and port number.
* Creating a Java Client.



Java code:

*server.java:*

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class Server

{

public static void main(String[] args) throws Exception

{

ServerSocket sersock = new ServerSocket(3000);

System.out.println("Server ready for chatting");

Socket sock = sersock.accept( );

// reading from keyboard (keyRead object)

BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

// sending to client (pwrite object)

OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite = new PrintWriter(ostream, true);

// receiving from server ( receiveRead object)

InputStream istream = sock.getInputStream();

BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

String receiveMessage, sendMessage;

while(true)

{

if((receiveMessage = receiveRead.readLine()) != null)

{

System.out.println(receiveMessage);

}

sendMessage = keyRead.readLine();

pwrite.println(sendMessage);

pwrite.flush();

}

}

}

*client.java:*

import java.io.\*;

import java.net.\*;

public class GossipClient

{

public static void main(String[] args) throws Exception

{

Socket sock = new Socket("127.0.0.1", 3000);

// reading from keyboard (keyRead object)

BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

// sending to client (pwrite object)

OutputStream ostream = sock.getOutputStream();

PrintWriter pwrite = new PrintWriter(ostream, true);

// receiving from server ( receiveRead object)

InputStream istream = sock.getInputStream();

BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

System.out.println("Start the chitchat, type and press Enter key");

String receiveMessage, sendMessage;

while(true)

{

sendMessage = keyRead.readLine(); // keyboard reading

pwrite.println(sendMessage); // sending to server

pwrite.flush(); // flush the data

if((receiveMessage = receiveRead.readLine()) != null) //receive from server

{

System.out.println(receiveMessage); // displaying at DOS prompt

}

}

}

}

OUTPUT:

