



SIDDAGANGA INSTITUTE OF TECHNOLOGY
www.sit.ac.in

Java Programming Lab

LAB MANUAL

Shwetha A N
Asst Professor
Dept of CSE, SIT



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Contents

I PART A	4
1 Classes and Objects	5
2 Overloading constructors and methods	8
3 Inheritance	10
4 Run Time Polymorphism	13
5 Packages	16
6 Dynamic Stack using Interfaces	18
7 Exception Handling	20
II PART B	22
1 Multithreading	23
2 Producer Consumer	26
3 Java Applets	28
3.1 Mouse Events	28
3.2 CardLayout Demo	31
4 Java Servlets	34
4.1 Servlets and Webpages	34
4.2 Server Information	36
5 Swing and JDBC	38
5.1 Login Page	38
5.2 Employee Details	41
6 Client-Server Interaction - Reading a File	44
7 Client-Server Interaction - Area of Circle	46

JAVA PROGRAMMING LABORATORY

Part I
PART A

Chapter 1

Classes and Objects

Write a program in Java with class `Rectangle` with the data fields `width`, `length`, `area` and `color`. The `length`, `width` and `area` are of `double` type and `color` is of `string` type. The methods are `set_length`, `set_width`, `set_color`, and `find_area`. Create two object of `Rectangle` and compare their `area` and `color`. If `area` and `color` both are the same for the objects then display "Matching Rectangles", otherwise display "Non matching Rectangle".

Java Code

```
import java.util.Scanner;
class Rectangle
{
    double length,width,area;
    String colour;
    Scanner s = new Scanner(System.in);

    void SetLength()
    {
        System.out.println("Enter the length of Rectangle");
        length = s.nextDouble();
    }

    void SetWidth()
    {
        System.out.println("Enter the width of Rectangle");
        width = s.nextDouble();
    }

    void SetColour()
    {
        System.out.println("Enter the colour of Rectangle");
        colour = s.next();
    }

    void FindArea()
    {
        area=length*width;
    }

    boolean compare(Rectangle a)
    {
        if(colour.equals(a.colour) && area==a.area)
            return true;
        else
            return false;
    }
}
```

```

class RectDemo
{
    public static void main(String args[])
    {
        Rectangle r1 = new Rectangle();
        Rectangle r2 = new Rectangle();
        boolean matches = false;

        System.out.println("\nRectangle 1 :");
        r1.SetLength();
        r1.SetWidth();
        r1.SetColour();

        System.out.println("\nRectangle 2 :");
        r2.SetLength();
        r2.SetWidth();
        r2.SetColour();

        r1.FindArea();
        r2.FindArea();
        matches = r1.compare(r2);

        if(matches == true)
            System.out.println("\nMatching Rectangle");
        else
            System.out.println("\nNot Matching Rectangle");
    }
}

```

Output

```

asus:1stjavaclass$ javac RectDemo.java
asus:1stjavaclass$ java RectDemo

```

```

Rectangle 1 :
Enter the length of Rectangle
3
Enter the width of Rectangle
4
Enter the colour of Rectangle
blue

```

```

Rectangle 2 :
Enter the length of Rectangle
3
Enter the width of Rectangle
4
Enter the colour of Rectangle
blue

```

Matching Rectangle

```

asus:1stjavaclass$ java RectDemo

```

```

Rectangle 1 :
Enter the length of Rectangle
3
Enter the width of Rectangle

```

4
Enter the colour of Rectangle
red

Rectangle 2 :
Enter the length of Rectangle
4
Enter the width of Rectangle
3
Enter the colour of Rectangle
black

Not Matching Rectangle

Chapter 2

Overloading constructors and methods

Write a java program to overload constructor and method.

Java Code

```
class Shape
{
    double length,breadth,area;

    Shape(double l)
    {
        length=l;
        breadth=l;
    }
    Shape(double l,double b)
    {
        length=l;
        breadth=b;
    }

    void compArea(double l)
    {
        area=l*l;
        System.out.println("Area = " + area + " units");
    }
    void compArea(double l,double b)
    {
        area=l*b;
        System.out.println("Area = " + area + " units");
    }
}
class OverloadEx
{
    public static void main(String args[])
    {
        Shape s1=new Shape(5.000);
        Shape s2=new Shape(5.000,6.000);
        s1.compArea(8.0);
        s2.compArea(s2.length,s2.breadth);

        s2.compArea(s2.length);
    }
}
```


Output

```
asus:1stjavaclaass$ javac OverloadEx.java
asus:1stjavaclaass$ java OverloadEx
Area = 64.0 units
Area = 30.0 units
Area = 25.0 units
asus:1stjavaclaass$
```

Chapter 3

Inheritance

Write a program in Java to create a Player class. Inherit the classes Cricket_Player, Football_Player and Hockey_Player from Player class.

Java Code

```
class Player{
    String name;
    int age,matches,ranking;
    Player(String n,int a,int m,int r){
        name=n;
        age=a;
        matches=m;
        ranking=r;
    }
}

class Cricket_Player extends Player{
    int High_score,Bowl_average,Bat_average;
    Cricket_Player(String a,int b,int c,int d,int e,int f,int g){
        super(a,b,c,d);
        High_score=e;
        Bat_average=f;
        Bowl_average=g;
    }
    void disp(){
        System.out.println("Name: "+name);
        System.out.println("Age: "+age);
        System.out.println("No. of Matches: "+matches);
        System.out.println("Highest Score: "+High_score);
        System.out.println("Batting Average: "+Bat_average);
        System.out.println("Balling Average: "+Bowl_average);
        System.out.println("Player Ranking: "+ranking);
    }
}

class Football_Player extends Player{
    int goals,g_avg,pass;
    Football_Player(String a,int b,int c,int d,int e,int f,int g){
        super(a,b,c,d);
        goals=e;
        g_avg=f;
        pass=g;
    }
    void disp(){
        System.out.println("Name: "+name);
```

```
        System.out.println("Age: "+age);
        System.out.println("No. of Matches: "+matches+"\n");
        System.out.println("No. of Goals: "+goals);
        System.out.println("Goal Average: "+g_avg);
        System.out.println("Passing Efficiency: "+pass+"%");
        System.out.println("Player Ranking: "+ranking);
    }
}

class Hockey_Player extends Player{
    int goals,g_avg,pass;
    Hockey_Player(String a,int b,int c,int d,int e,int f,int g){
        super(a,b,c,d);
        goals=e;
        g_avg=f;
        pass=g;
    }
    void disp(){
        System.out.println("Name: "+name);
        System.out.println("Age: "+age);
        System.out.println("No. of Matches: "+matches);
        System.out.println("No. of Goals: "+goals);
        System.out.println("Goal Average: "+g_avg);
        System.out.println("Passing Efficiency: "+pass+"%");
        System.out.println("Player Ranking: "+ranking);
    }
}

class PlayerDemo{
    public static void main(String args[]){
        Cricket_Player C=new Cricket_Player("Sachin Tendulkar",38,600,8,200,55,60);
        Football_Player F=new Football_Player("Leonel Messi",32,120,90,3,80,94);
        Hockey_Player H=new Hockey_Player("Dhanraj Pillay",32,120,90,3,80,94);
        C.disp();
        F.disp();
        H.disp();
    }
}
```

Output

```
asus:Programs_lab$ javac PlayerDemo.java
```

```
asus:Programs_lab$ java PlayerDemo
```

```
Name: Sachin Tendulkar  
Age: 38  
No. of Matches: 600  
Highest Score: 200  
Batting Average: 55  
Ballling Average: 60  
Player Ranking: 8
```

```
Name: Leonel Messi  
Age: 32  
No. of Matches: 120  
No. of Goals: 3  
Goal Average: 80  
Passing Efficiency: 94%  
Player Ranking: 90
```

```
Name: Dhanraj Pillay  
Age: 32  
No. of Matches: 120  
No. of Goals: 3  
Goal Average: 80  
Passing Efficiency: 94%  
Player Ranking: 90
```

Chapter 4

Run Time Polymorphism

Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.

Java Code

```
class TrunkCall
{
    double duration;
    double charge;
    TrunkCall()
    {
        duration=0;
    }
    TrunkCall(double d)
    {
        duration=d;
    }
    void calcCharge()
    {
        System.out.println("No Policy");
    }
}

class OrdinaryCall extends TrunkCall
{
    double call_rate;
    OrdinaryCall()
    {
        super();
        call_rate=0.60;
    }
    OrdinaryCall(double d)
    {
        super(d);
        call_rate=0.60;
    }
    OrdinaryCall(double d,double f)
    {
        super(d);
        call_rate=f;
    }
    void calcCharge()
    {
        charge=duration*call_rate;
    }
}
```

```
        System.out.println("For OrdinaryCall charge :"+charge);
    }
}

class UrgentCall extends TrunkCall
{
    double call_rate;
    UrgentCall()
    {
        super();
        call_rate=1.0;
    }
    UrgentCall(double d)
    {
        super(d);
        call_rate=1.0;
    }
    UrgentCall(double d,double f)
    {
        super(d);
        call_rate=f;
    }
    void calcCharge()
    {
        charge=duration*call_rate;
        System.out.println("For UrgentCall charge:"+charge);
    }
}

class LightningCall extends TrunkCall
{
    double call_rate;
    LightningCall()
    {
        super();
        call_rate=1.2;
    }
    LightningCall(double d)
    {
        super(d);
        call_rate=1.2;
    }
    LightningCall(double d,double f)
    {
        super(d);
        call_rate=f;
    }
    void calcCharge()
    {
        charge=duration*call_rate;
        System.out.println("For LightningCall charge:"+charge);
    }
}

class Telephone
{
    public static void main(String args[])
    {
        TrunkCall tref;
        OrdinaryCall ordCall=new OrdinaryCall(4);
        UrgentCall urgCall=new UrgentCall(1.0,2.0);
        LightningCall ligCall=new LightningCall(2.0,3.0);
```

```
        tref=ordCall;
        tref.calcCharge();
        tref=urgCall;
        tref.calcCharge();
        tref=ligCall;
        tref.calcCharge();
    }
}
```

Output

```
sus:Programs_lab$ javac Telephone.java
asus:Programs_lab$ java Telephone
For OrdinaryCall charge :2.4
For UrgentCall charge:2.0
For LightningCall charge:6.0
asus:Programs_lab$ javac Telephone.java
asus:Programs_lab$ java Telephone
For OrdinaryCall charge :2.4
For UrgentCall charge:2.0
For LightningCall charge:6.0
```

Chapter 5

Packages

Write a program to make a package Balance in which has Account class with Display_Balance method in it. Import Balance package in another program to access Display_Balance method of Account class.

Java Code

Package

```
package Balance;
public class Account
{
    double principal, rate, balance;
    int time;
    public Account(double pr,int ti,double ra)
    {
        principal = pr;
        time = ti;
        rate = ra;
    }
    public void calcAmount()
    {
        balance = principal * rate * time;
    }
    public void DisplayBalance()
    {
        System.out.println("\n\nPrincipal Amount: "+ principal + "Rs\nTime:" + time +
        "Years\n\nCurrent Balance: " + balance + "Rs");
    }
}
```

Driver Program

```
import Balance.*;
class DemoPackage
{
    public static void main(String args[])
    {
        Account acc = new Account(5000,2,3);
        acc.calcAmount();
        acc.DisplayBalance();
    }
}
```


Output

```
asus:Package$ javac DemoPackage.java
asus:Package$ java DemoPackage
```

```
Principal Amount: 5000.0Rs
Time:2Years
```

```
Current Balance: 30000.0Rs
```

Chapter 6

Dynamic Stack using Interfaces

Create the dynamic stack by implementing the interfaces that defines push() and pop() methods.

Java Code

```
interface IntStack
{
    void push(int item);
    int pop();
}

class DynStack implements IntStack
{
    int stk[];
    int tos;

    DynStack(int size)
    {
        stk = new int[size];
        tos = -1;
    }
    public void push(int item)
    {
        if(tos ==stk.length-1)
        {
            int temp[] = new int[stk.length*2];
            for(int i = 0;i<stk.length;i++)
                temp[i] = stk[i];
            stk = temp;
            stk[++tos] = item;
            System.out.println("Stack size increased");
        }
        else
            stk[++tos] = item;
    }
    public int pop()
    {
        if(tos<0)
        {
            System.out.println("stack underflow");
            return 0;
        }
        else
            return stk[tos--];
    }
}
```

```
class DynStackDemo
{
    public static void main(String args[])
    {
        DynStack mystack1 = new DynStack(5);
        DynStack mystack2 = new DynStack(8);
        for(int i=0;i<20;i++)
            mystack1.push(i);
        for(int i=0;i<20;i++)
            mystack2.push(i);
        System.out.print("\t Elements in stack1 -> ");
        for(int i=0;i<20;i++)
            System.out.print(mystack1.pop()+" ");
        System.out.println();
        System.out.print("\t Elements in stack2 -> ");
        for(int i=0;i<20;i++)
            System.out.print(mystack2.pop()+" ");
        System.out.println();
    }
}
```

Output

```
asus:interface$ javac DynStackDemo.java
asus:interface$ java DynStackDemo
Stack size increased
Stack size increased
Stack size increased
Stack size increased
Elements in stack1 -> 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Elements in stack2 -> 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

Chapter 7

Exception Handling

On a single track two vehicles are running. As vehicles are going in same direction there is no problem. If the vehicles are running in different direction there is a chance of collision. To avoid collisions write a Java program using exception handling. You are free to make necessary assumptions.

Java Code

```
class Collision
{
    String DirectionTrain1, DirectionTrain2;
    Collision(String dir1,String dir2)
    {
        DirectionTrain1 = dir1;
        DirectionTrain2 = dir2;
    }
    void checkCollision()
    {
        try
        {
            if(DirectionTrain1==DirectionTrain2)
            {
                System.out.println("The two vehicles are moving in same direction,
                hence no collision in Pair 1");
            }
            else
            {
                throw new Exception("The two vehicles are moving in opposite directions,
                so collision occurs in Pair 2");
            }
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}

class ExceptionDemo
{
    public static void main(String args[])
    {
        Collision pair1 = new Collision("north","north");
        Collision pair2 = new Collision("north","south");
        pair1.checkCollision();
        System.out.println();
        pair2.checkCollision();
    }
}
```

```
        System.out.println();  
    }  
}
```

Output

```
asus:Programs_lab$ javac ExceptionDemo.java
```

```
asus:Programs_lab$ java ExceptionDemo
```

```
The two vehicles are moving in same direction,hence no collision in Pair 1
```

```
java.lang.Exception: The two vehicles are moving in opposite directions,so collision occurs in Pair 2
```

Part II

PART B

Chapter 1

Multithreading

Thread Priorities

Write a Java program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which thread is long lasting.

Java Code

```
class MulThread implements Runnable
{
    static String last;
    String name;
    Thread t;
    MulThread(String n,int p)
    {
        name=n;
        t=new Thread(this, name);
        t.setPriority(p);
        System.out.println(name+" started");
        System.out.println("new thread: "+t);
        t.start();
    }
    public void run()
    {
        try
        {
            if((t.getPriority()==9)||t.getPriority()==10)
            {
                Thread.sleep(1000);
                System.out.println(t.getName()+" is going to sleep");
            }
            for(int i=0;i<5;i++)
            {
                System.out.println(name+": "+i);
                Thread.sleep(500);
            }
        }
        catch(InterruptedException e)
        {
            System.out.println(name+" thread interrupted");
        }
        last=name;
        System.out.println(name+" exiting");
    }
}
```

```

class NewThread
{
    public static void main(String args[])
    {
        Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
        MulThread m1=new MulThread("one",Thread.NORM_PRIORITY-1);
        MulThread m2=new MulThread("two",Thread.MAX_PRIORITY);
        MulThread m3=new MulThread("three",Thread.NORM_PRIORITY+2);
        MulThread m4=new MulThread("four",Thread.NORM_PRIORITY+4);
        MulThread m5=new MulThread("five",Thread.MIN_PRIORITY+1);
        try
        {
            Thread.sleep(500);
        }
        catch(InterruptedException e)
        {
            System.out.println("main thread interrupted");
        }
        System.out.println("Thread one is:"+m1.t.isAlive());
        System.out.println("Thread two is:"+m2.t.isAlive());
        System.out.println("Thread three is:"+m3.t.isAlive());
        System.out.println("Thread four is:"+m4.t.isAlive());
        System.out.println("Thread five is:"+m5.t.isAlive());
        try
        {
            System.out.println("waiting for thread to finish");
            m1.t.join();
            m2.t.join();
            m3.t.join();
            m4.t.join();
            m5.t.join();
        }
        catch(InterruptedException e)
        {
            System.out.println("main thread interrupted");
        }
        System.out.println("thread one is:"+m1.t.isAlive());
        System.out.println("thread two is:"+m2.t.isAlive());
        System.out.println("thread three is:"+m3.t.isAlive());
        System.out.println("thread four is:"+m4.t.isAlive());
        System.out.println("thread five is:"+m5.t.isAlive());
        System.out.println();
        System.out.println("priority of one:"+m1.t.getPriority());
        System.out.println("priority of two:"+m2.t.getPriority());
        System.out.println("priority of three:"+m3.t.getPriority());
        System.out.println("priority of four:"+m4.t.getPriority());
        System.out.println("priority of five:"+m5.t.getPriority());
        System.out.println();
        System.out.println(MulThread.last+" is long lasting thread");
    }
}

```

Output

```

asus:Programs_lab$ javac ThreadDemo.java
asus:Programs_lab$ java ThreadDemo
one started
new thread: Thread[one,4,main]
two started

```



```
new thread: Thread[two,10,main]
one:0
three started
new thread: Thread[three,7,main]
four started
new thread: Thread[four,9,main]
five started
three:0
new thread: Thread[five,2,main]
five:0
one:1
three:1
Thread one is:true
five:1
Thread two is:true
Thread three is:true
Thread four is:true
Thread five is:true
waiting for thread to finish
two is going to sleep
two:0
one:2
four is going to sleep
three:2
four:0
five:2
two:1
one:3
three:3
four:1
five:3
two:2
one:4
three:4
four:2
five:4
two:3
one exiting
three exiting
five exiting
four:3
two:4
four:4
two exiting
four exiting
thread one is:false
thread two is:false
thread three is:false
thread four is:false
thread five is:false

priority of one:4
priority of two:10
priority of three:7
priority of four:9
priority of five:2

four is long lasting thread
```

Chapter 2

Producer Consumer

Write a multi threaded Java program to implement producer-consumer problem.

Java Code

```
class Q
{
    int n;
    boolean valueSet = false;
    synchronized int get()
    {
        while(!valueSet)
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("InterruptedException caught");
            }
        System.out.println("Got:" + n);
        valueSet = false;
        notify();
        return n;
    }
    synchronized void put(int n)
    {
        while(valueSet)
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("InterruptedException caught");
            }
        this.n = n;
        valueSet = true;
        System.out.println("Put:" + n);
        notify();
    }
}
class Producer implements Runnable
{
    Q q;
    Thread t;
```

```
    Producer(Q q)
    {
        this.q = q;
        t = new Thread(this,"Producer");
        t.start();
    }
    public void run()
    {
        int i = 0;
        while(true)
        {
            q.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    Q q;
    Thread t;
    Consumer(Q q)
    {
        this.q = q;
        t = new Thread(this,"Consumer");
        t.start();
    }
    public void run()
    {
        while(true)
        {
            q.get();
        }
    }
}

class ProducerConsumer
{
    public static void main(String args[])
    {
        System.out.println("Press Control-C to stop");
        Q q = new Q();
        Producer p = new Producer(q);
        Consumer c = new Consumer(q);
    }
}
```

Output

```
Press Control-C to stop
Put:0
Got:0
Put:1
Got:1
Put:2
Got:2
Put:3
Got:3
Put:4
Got:4
Put:5
Got:5
```

Chapter 3

Java Applets

Write a Java Applet program to perform the following

- i Create an applet to handle all mouse events.
- ii Design an applet which uses Card layout with 3 Buttons. When the user clicks on any button, the background layout color must change.

3.1 Mouse Events

Java Code

```
// Demonstrate the mouse event handlers.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="MouseEvents" width=300 height=100>
</applet>
*/
public class MouseEvents extends Applet
implements MouseListener, MouseMotionListener {
    String msg = "";
    int mouseX = 0, mouseY = 0; // coordinates of mouse
    public void init() {
        addMouseListener(this);
        addMouseMotionListener(this);
    }
    // Handle mouse clicked.
    public void mouseClicked(MouseEvent me) {
        // save coordinates
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse clicked.";
        repaint();
    }
    // Handle mouse entered.
    public void mouseEntered(MouseEvent me) {
        // save coordinates
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse entered.";
        repaint();
    }
    // Handle mouse exited.
    public void mouseExited(MouseEvent me) {
        // save coordinates
```

```
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse exited.";
        repaint();
    }
    // Handle button pressed.
    public void mousePressed(MouseEvent me) {
        // save coordinates
        mouseX = me.getX();
        mouseY = me.getY();
        msg = "Down";
        repaint();
    }

    // Handle button released.
    public void mouseReleased(MouseEvent me) {
        // save coordinates
        mouseX = me.getX();
        mouseY = me.getY();
        msg = "Up";
        repaint();
    }

    // Handle mouse dragged.
    public void mouseDragged(MouseEvent me) {
        // save coordinates
        mouseX = me.getX();
        mouseY = me.getY();
        msg = "*";
        showStatus("Dragging mouse at " + mouseX + ", " + mouseY);
        repaint();
    }

    // Handle mouse moved.
    public void mouseMoved(MouseEvent me) {
        // show status
        showStatus("Moving mouse at " + me.getX() + ", " + me.getY());
    }

    // Display msg in applet window at current X,Y location.
    public void paint(Graphics g) {
        g.drawString(msg, mouseX, mouseY);
    }
}
```

Output

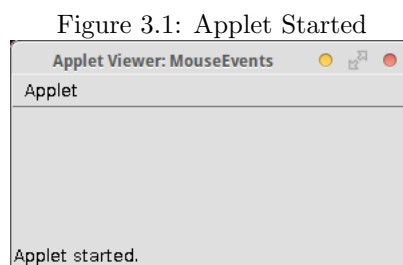


Figure 3.2: Mouse Entered

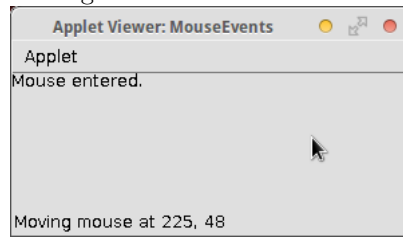


Figure 3.3: Mouse Drag

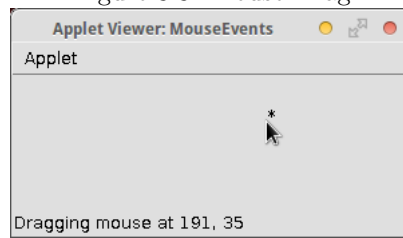


Figure 3.4: Mouse Clicked

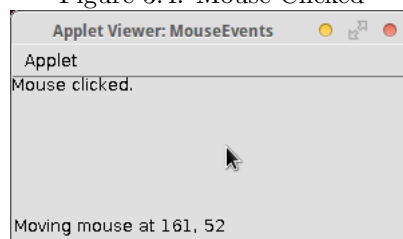
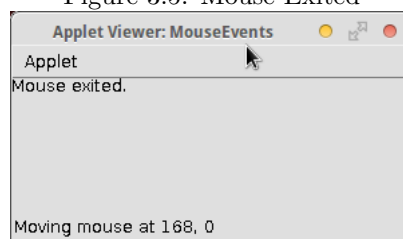


Figure 3.5: Mouse Exited



3.2 CardLayout Demo

Java Code

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//<applet code="CardLayoutDemo" width=450 height=500></applet>
public class CardLayoutDemo extends Applet implements ActionListener {
    String msg;
    Button redButton,greenButton,blueButton;
    Panel mypanel;
    CardLayout cardl;
    Label myLabel;
    public void init() {
        myLabel=new Label("click the button to change background color");
        blueButton=new Button("BLUE");
        redButton=new Button("RED");
        greenButton=new Button("GREEN");
        cardl=new CardLayout();
        mypanel=new Panel();
        mypanel.setLayout(cardl);
        Panel mypanel1=new Panel();
        mypanel1.add(myLabel);
        mypanel1.add(redButton);
        mypanel1.add(blueButton);
        mypanel1.add(greenButton);
        mypanel.add(mypanel1,"Panel");

        add(mypanel);
        redButton.addActionListener(this);
        blueButton.addActionListener(this);
        greenButton.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae) {
        String str=ae.getActionCommand();
        if(str.equals("RED")) {
            showStatus("you pressed red. The background color changes to red");
            setBackground(Color.red);
        }
        if(str.equals("BLUE")) {
            showStatus("you pressed blue. The background color changes to blue");
            setBackground(Color.blue);
        }
        if(str.equals("GREEN")) {
            showStatus("you pressed green. The background color changes to green");
            setBackground(Color.green);
        }
    }
}
```

HTML Code - CardLayoutDemo.html

```
<html>
    <applet code="CardLayoutDemo" width=450 height=500>
    </applet>
</html>
```

Output

Figure 3.6: Applet Started

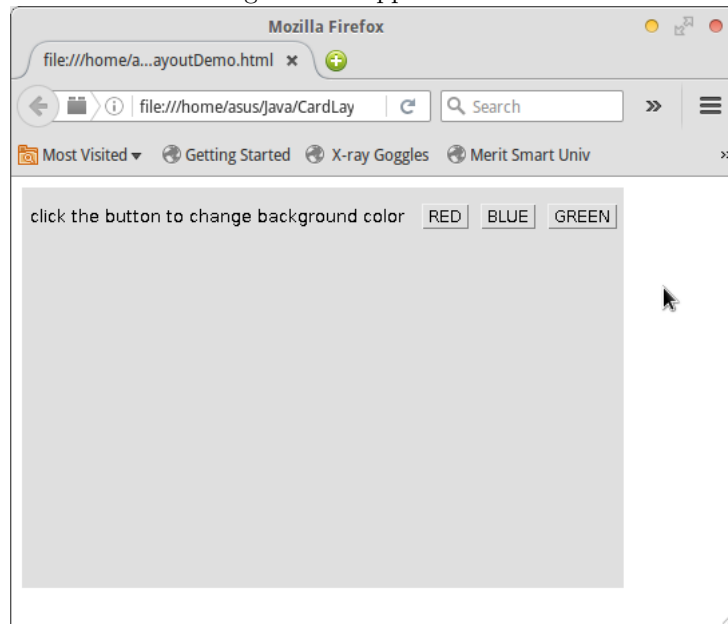


Figure 3.7: Red Button Clicked

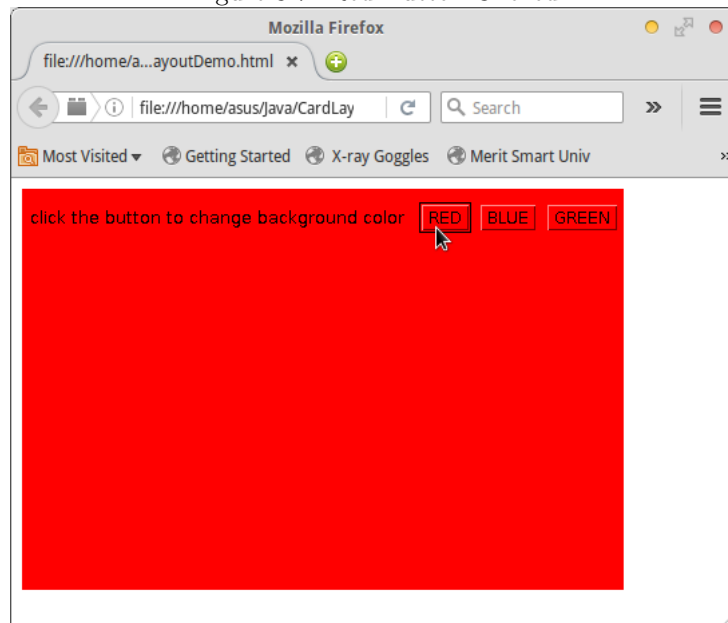


Figure 3.8: Blue Button Clicked

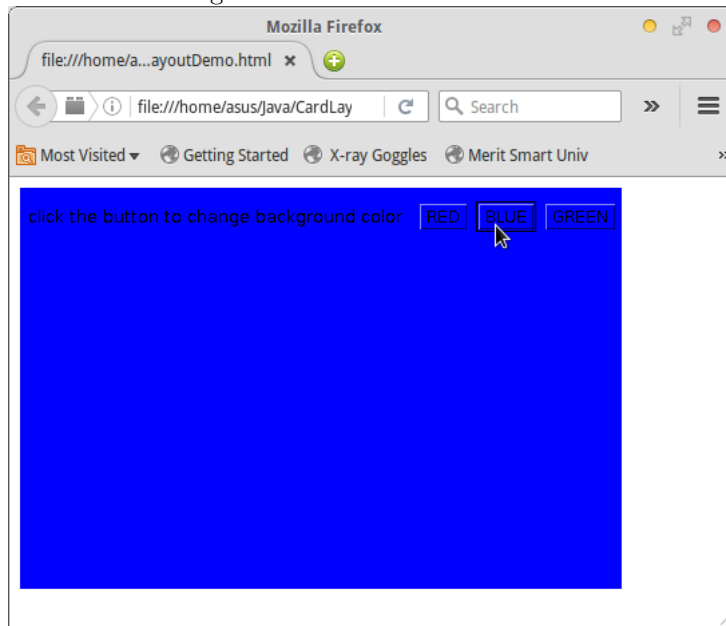
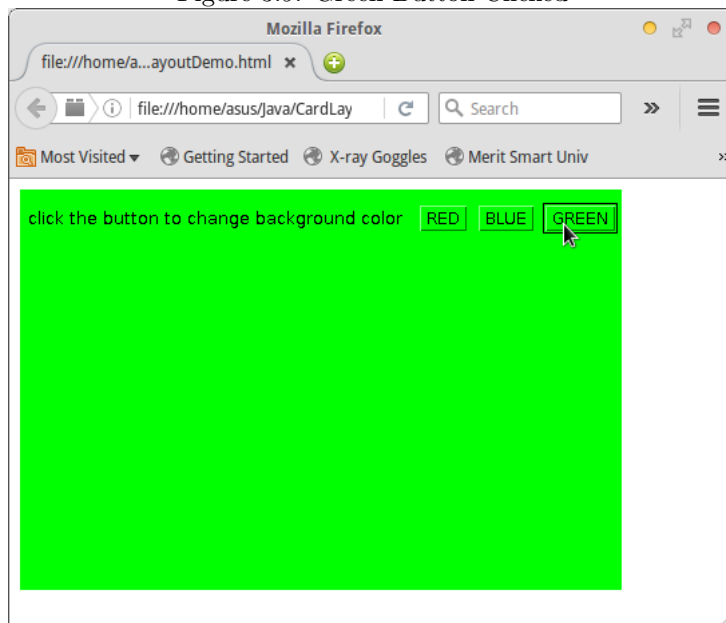


Figure 3.9: Green Button Clicked



Chapter 4

Java Servlets

4.1 Servlets and Webpages

Java Servlet Program to accept username, address and display them in a web page by passing parameters.

HTML Code

```
<html>
  <head>
    <title>Greeting...</title>
  </head>
  <body>
    <h1 align=center>GREETING A USER</h1>
    <hr/>
    <form method=get action="http://localhost:8080/examples/servlets/servlet/WebForm">
      <table>
        <tr>
          <td align="right"><b>NAME:</b></td>
          <td><input type=text name=uname /></td>
        </tr>
        <tr>
          <td><b>ADDRESS:</b></td>
          <td><input type=text name=address /></td>
        </tr>
      </table>
      <input type=submit value=SUBMIT />
      <input type=reset value=CLEAR />
      <hr/>
    </form>
  </body>
</html>
```

Java Code

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class WebForm extends HttpServlet{
  public void doGet(HttpServletRequest request,HttpServletResponse response)
  throws IOException,ServletException{
    String name, addr;
    response.setContentType("text/html");
    name=request.getParameter("uname");
    addr=request.getParameter("address");
    PrintWriter out=response.getWriter();
```

```
        out.println("<html><body>");
        out.println("<h1 align=center> Welcome " + name + "</h1><hr>Address:" + addr);
        out.close();
    }
}
```

Output

Figure 4.1: Input from Webpage

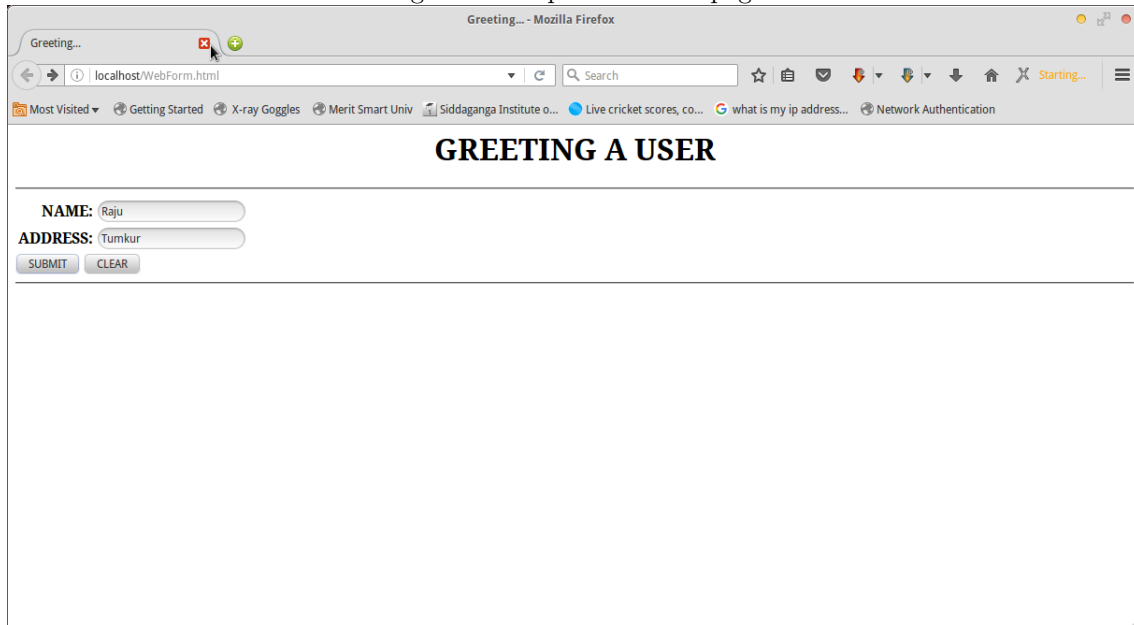
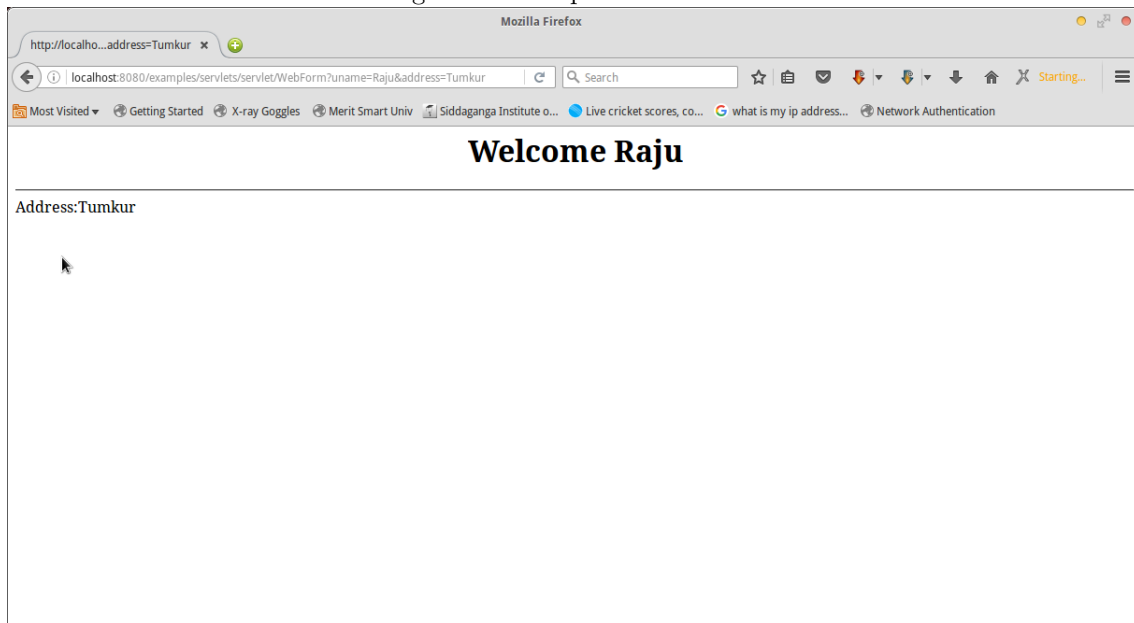


Figure 4.2: Output from Servlet



4.2 Server Information

Java Servlet Program to request server information viz Request Method, URL, Protocol and remote address.

Java Code

```
import javax.servlet.*;
import java.io.*;
import javax.servlet.http.*;
public class ServerInfo extends HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException,ServletException{
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        out.println("<html><head>");
        out.println("<title>Server Information</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1 align=center> SERVER INFORMATTION");
        out.println("<hr><br><center><table border=5><tr>");

        out.println("<td><b>Request Method</b></td>");
        out.println("<td>");
        out.println(request.getMethod());
        out.println("</td></tr>");
        out.println("<tr>");

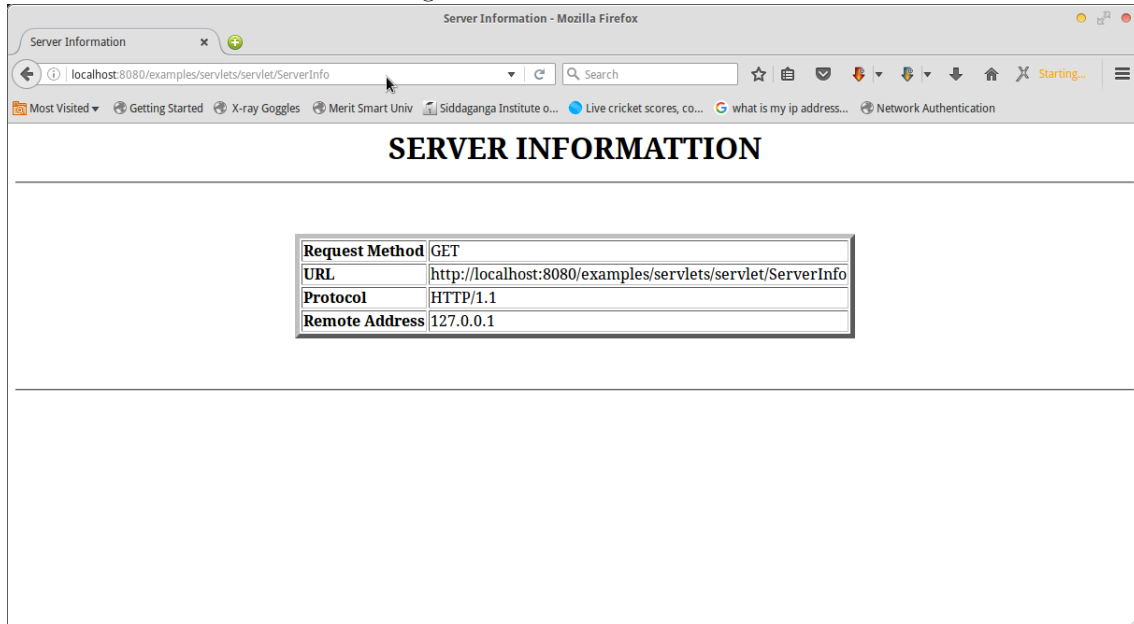
        out.println("<td><b> URL</b></td>");
        out.println("<td>");
        out.println(request.getRequestURL());
        out.println("</td></tr>");
        out.println("<tr>");

        out.println("<td><b>Protocol </b></td>");
        out.println("<td>");
        out.println(request.getProtocol());
        out.println("</td></tr>");
        out.println("<tr>");

        out.println("<td><b>Remote Address<b></td>");
        out.println("<td>");
        out.println(request.getRemoteAddr());
        out.println("</td></tr></table>");
        out.println("<br><hr>");
        out.println("</body></html>");
    }
}
```

Output

Figure 4.3: Server Information



Chapter 5

Swing and JDBC

5.1 Login Page

Create a login page using swing components, after successful authentication , the application should display the valid email-id of the person.

Java Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class login extends JFrame implements ActionListener
{
    JButton SUBMIT;
    JPanel panel;
    JLabel usrLabel, pwdLabel, emailLabel;
    JTextField usrText, pwdText, emailText;
    static final String dbClass = "com.mysql.jdbc.Driver";

    login()
    {
        usrLabel = new JLabel();
        usrLabel.setText("username");
        usrText = new JTextField(25);

        pwdLabel = new JLabel();
        pwdLabel.setText("password");
        pwdText = new JPasswordField(25);

        emailLabel = new JLabel();
        emailLabel.setText("email");
        emailText = new JTextField(25);

        SUBMIT = new JButton("SUBMIT");
        panel = new JPanel(new GridLayout(4,2));

        panel.add(usrLabel);
        panel.add(usrText);
        panel.add(pwdLabel);
        panel.add(pwdText);
        panel.add(emailLabel);
        panel.add(emailText);
        panel.add(SUBMIT);
        add(panel, BorderLayout.CENTER);
        SUBMIT.addActionListener(this);
    }
}
```

```

        setTitle("login form");
    }
    public void actionPerformed(ActionEvent ae)
    {
        String usn = usrText.getText();
        String pwd = pwdText.getText();
        String email = emailText.getText();
        java.sql.Connection conn = null;
        try
        {
            Class.forName(dbClass).newInstance();
            conn = java.sql.DriverManager.getConnection
("jdbc:mysql://localhost/ACCOUNTS?user=root&password=root123");
        }catch(ClassNotFoundException e){
            System.out.println("error in loading driver"+e);
            System.exit(1);
        }
        catch(Exception e)
        {
            System.out.println("error in connection"+e);
            System.exit(0);
        }

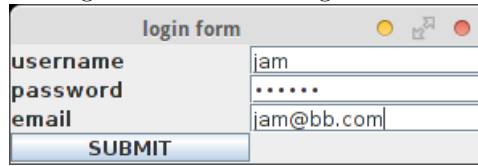
        System.out.println("connection established");
        try{
            java.sql.Statement s = conn.createStatement();
            String query =
            "SELECT * FROM USERS WHERE username = '"+usn+"'and password = '"+pwd+"'";
            java.sql.ResultSet r = s.executeQuery(query);
            r.next();
            int x = r.getRow();
            if (x>0){
                JOptionPane.showMessageDialog(null,"Your Mail id : " + email);
            }
            else{
                JOptionPane.showMessageDialog(this,"incorrect login or password",
                "error",JOptionPane.ERROR_MESSAGE);
            }
        }
        catch(Exception e)
        {
            System.out.println(e);
            System.exit(0);
        }
    }
}

class LoginDemo{
    public static void main(String args[])
    {
        try{
            login frame = new login();
            frame.setSize(300,100);
            frame.setVisible(true);
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(null,e.getMessage());
        }
    }
}

```

Output

Figure 5.1: Correct Login Details

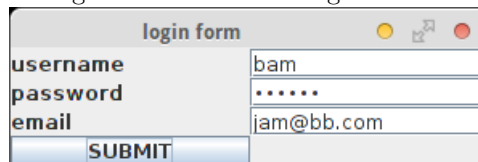


The screenshot shows a Java Swing window titled "login form". It contains three text input fields: "username" with the value "jam", "password" with masked characters "*****", and "email" with the value "jam@bb.com". Below the fields is a "SUBMIT" button.

Figure 5.2: Display Email

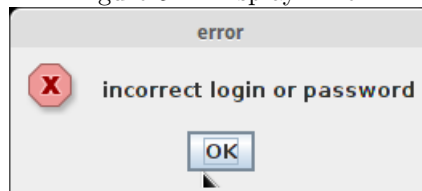


Figure 5.3: Incorrect Login Details



The screenshot shows a Java Swing window titled "login form". It contains three text input fields: "username" with the value "bam", "password" with masked characters "*****", and "email" with the value "jam@bb.com". Below the fields is a "SUBMIT" button.

Figure 5.4: Display Error



5.2 Employee Details

Write an JDBC application displays the employee numbers who are having maximum and minimum salaries from the database.

Java Code

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class EmployeeDB {
    JLabel jlab;

    static final String dbClass = "com.mysql.jdbc.Driver";
    EmployeeDB() {
        JButton jbbtnMin = new JButton("Min Salary");
        JButton jbbtnMax = new JButton("Max Salary");

        // Create a new JFrame container.
        JFrame jfrm = new JFrame("Employee Details");
        // Specify GridLayout for the layout manager.
        jfrm.setLayout(new GridLayout(3,1));
        // Specify FlowLayout for the layout manager.
//        jfrm.setLayout(new FlowLayout(FlowLayout.CENTER));
        // Give the frame an initial size.
        jfrm.setSize(300, 300);
        // Terminate the program when the user closes the application.
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Make two buttons.

        // Add action listener for Alpha.
        jbbtnMin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                fetchDatabase(1);
            }
        });

        // Add action listener for Max.
        jbbtnMax.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                fetchDatabase(2);
            }
        });
        // Add the buttons to the content pane.
        jfrm.add(jbbtnMin);
        jfrm.add(jbbtnMax);
        // Create a text-based label.
        jlab = new JLabel("Press a button.",JLabel.CENTER);
        // Add the label to the content pane.
        jfrm.add(jlab);
    }
}
```

```

        // Display the frame.
        jfrm.setVisible(true);
    }

    void fetchDatabase(int code){
        java.sql.Connection conn = null;
        try
        {
            Class.forName(dbClass).newInstance();
            conn = java.sql.DriverManager.getConnection
                ("jdbc:mysql://localhost/COMPANY?user=root&password=root123");
        }catch(ClassNotFoundException e){
            System.out.println("error in loading driver"+e);
            System.exit(1);
        }
        catch(Exception e){
            System.out.println("error in connection"+e);
            System.exit(0);
        }

        try{
            java.sql.Statement s = conn.createStatement();
            String query1 =
                "SELECT * FROM 'EMPLOYEE' WHERE emp_sal IN (SELECT MIN(emp_sal) FROM EMPLOYEE)";
            String query2 =
                "SELECT * FROM 'EMPLOYEE' WHERE emp_sal IN (SELECT MAX(emp_sal) FROM EMPLOYEE)";

            java.sql.ResultSet r;
            if(code == 1){
                r = s.executeQuery(query1);
            }else{
                r = s.executeQuery(query2);
            }

            if(r.next()){
                jlab.setText(r.getString("emp_id")+ " | " +
                    r.getString("emp_name") + " | Rs " +
                    r.getString("emp_sal") + "/-");
            }
        }
        catch(Exception e)
        {
            System.out.println(e);
            System.exit(0);
        }
    }

    public static void main(String args[]) {
        // Create the frame on the event dispatching thread.
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new EmployeeDB();
            }
        });
    }
}

```

Output

Figure 5.5: Initial Output Window

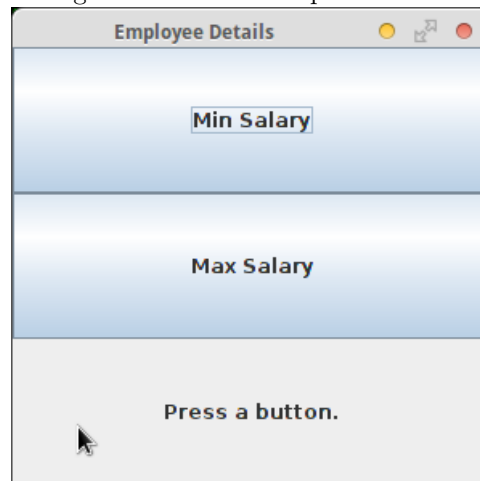
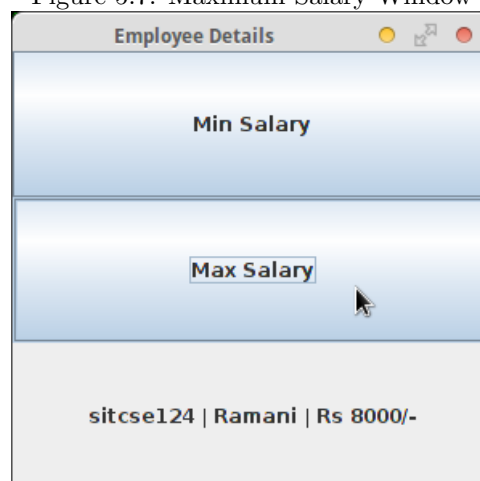


Figure 5.6: Minimum Salary Window



Figure 5.7: Maximum Salary Window



Chapter 6

Client-Server Interaction - Reading a File

Write a JAVA Program to implement Client Server interaction (Client requests a file, Server responds to client with contents of that file which is then displayed on the screen by Client-Socket Programming).

Java Code

Server Program

```
import java.net.*;
import java.io.*;
public class ContentsServer{
    public static void main(String args[]) throws Exception{
        // establishing the connection with the server
        ServerSocket sersock = new ServerSocket(4000);
        System.out.println("Server ready for connection");
        Socket sock = sersock.accept();
        // binding with port: 4000
        System.out.println("Connection is successful and waiting for interaction");
        // reading the file name from client
        InputStream istream = sock.getInputStream();
        BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));
        String fname = fileRead.readLine();
        // reading file contents
        BufferedReader contentRead = new BufferedReader(new FileReader(fname));
        // keeping output stream ready to send the contents
        OutputStream ostream = sock.getOutputStream();
        PrintWriter pwrite = new PrintWriter(ostream, true);
        String str;
        // reading line-by-line from file
        while((str = contentRead.readLine()) != null) {
            pwrite.println(str);
            // sending each line to client
        }
        sock.close();
        sersock.close();
        // closing network sockets
        pwrite.close();
        fileRead.close();
        contentRead.close();
    }
}
```

Client Program

```
import java.net.*;
import java.io.*;
public class ContentsClient{
    public static void main(String args[]) throws Exception {
        Socket sock = new Socket( "127.0.0.1", 4000);
        // reading the file name from keyboard. Uses input stream
        System.out.print("Enter the file name : ");
        BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
        String fname = keyRead.readLine();
        // sending the file name to server. Uses PrintWriter
        OutputStream ostream = sock.getOutputStream( );
        PrintWriter pwrite = new PrintWriter(ostream, true);
        pwrite.println(fname);
        // receiving the contents from server. Uses input stream
        InputStream istream = sock.getInputStream();
        BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));
        String str;
        // reading line-by-line
        while((str = socketRead.readLine()) != null){
            System.out.println(str);
        }
        pwrite.close();
        socketRead.close();
        keyRead.close();
    }
}
```

Output

Server Side

```
asus:server$ javac ContentsServer.java
asus:server$ java ContentsServer
Server ready for connection
Connection is successful and waiting for interaction
asus:server$
```

Client Side

```
asus:client$ javac ContentsClient.java
asus:client$ java ContentsClient
Enter the file name : sample.txt
Don't feed the bats tonight.
asus:client$
```

Chapter 7

Client-Server Interaction - Area of Circle

Write a client-server java program, to find the area of a circle on server side, when a client sends a request along with given radius and then display the result on the client side.

Java Code

Server Program

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

public class CircleServer{
    static final double myPI = 3.1416;
    public static void main(String args[]) throws Exception{
        // establishing the connection with the server
        ServerSocket sersock = new ServerSocket(4000);
        System.out.println("Server ready for connection");
        Socket sock = sersock.accept();
        // binding with port: 4000
        System.out.println("Connection is successful and waiting for interaction");
        // reading the file name from client
        InputStream istream = sock.getInputStream();
        BufferedReader strRead =new BufferedReader(new InputStreamReader(istream));
        String radius = strRead.readLine();

        int iRadius;
        iRadius = Integer.parseInt(radius);
        double dArea;
        dArea = myPI * iRadius * iRadius;

        String result = "Area is " + dArea + " units";
        // reading file contents
        OutputStream ostream = sock.getOutputStream();
        PrintWriter pwrite = new PrintWriter(ostream, true);

        pwrite.println(result);

        System.out.println("Closing the socket");
        sock.close();
        sersock.close();
        // closing network sockets
        pwrite.close();
        strRead.close();
    }
}
```

```
}

```

Client Program

```
import java.net.*;
import java.io.*;
public class CircleClient{
    public static void main(String args[]) throws Exception {
        Socket sock = new Socket( "127.0.0.1", 4000);
        // reading the file name from keyboard. Uses input stream
        System.out.print("Enter the radius : ");
        BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
        String radius = keyRead.readLine();
        // sending the file name to server. Uses PrintWriter
        OutputStream ostream = sock.getOutputStream( );
        PrintWriter pwrite = new PrintWriter(ostream, true);
        pwrite.println(radius);
        // receiving the contents from server. Uses input stream
        InputStream istream = sock.getInputStream();
        BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));
        String str;
        str = socketRead.readLine();
        System.out.println(str);
        pwrite.close();
        socketRead.close();
        keyRead.close();
    }
}

```

Output

Server Side

```
asus:server$ javac CircleServer.java
asus:server$ java CircleServer
Server ready for connection
Connection is successful and waiting for interaction
Closing the socket

```

Client Side

```
asus:client$ javac CircleClient.java
asus:client$ java CircleClient
Enter the radius : 4
Area is 50.2656 units
asus:client$

```