# JAVA LAB
# (DMCA222)
# (MCA)


## ACHARYA NAGARJUNA UNIVERSITY

### CENTRE FOR DISTANCE EDUCATION

NAGARJUNA NAGAR,

GUNTUR

ANDHRA PRADESH

# Lab-II

# Object Oriented Programming Lab in JAVA

## CYCLE I: C++ Programming:

## Program 1:

## Aim: Write a C++ program for simple calculator using class and object.

**Procedure:**

1. Create a class calculator.

2. Implement all mathematical functions like addition, subtraction, multiplication, division, square, square root, power etc.

3. Create object for calculator and call the mathematical functions.

**Program:**

```
#include<iostream.h>

#include<math.h>

class calculator

{

public:

double value;

calculator();

double add(double x, double y);

double subtract(double x, double y);

double multiply(double x, double y);

double divide(double x, double y);
```

```
double square(double x);
double square_root(double x);
double power(double x, double y);
};

calculator::calculator()
{
value = 0.0;
}
double calculator::add(double x, double y)
{
value = x + y;
return value;
}
double calculator::subtract(double x, double y)
{
value = x - y;
return value;
}
double calculator::multiply(double x, double y)
{
value = x * y;
return value;
}
```

```
double calculator::divide(double x, double y)
{
value = x / y;
return value;
}
double calculator::square(double x)
{
value = x * x;
return value;
}
double calculator::square_root(double x)
{
value = sqrt(x);
return value;
}
double calculator::power(double x, double psize)
{
value = x;
for(int i = 1; i < psize; i++)
{
value = value * x;
}
return value;
}
```

```
void main()
{
calculator c;
int choice = 0;
double x = 0, y = 0;

while(choice <= 7)
{
cout<<"operations\n";
cout<<"1.Add\n2.Subtract\n3.Multiply\n4.Divide\n5.Square\n6.Square Root\n7.Power\n";
cout<<"\n\nEnter choice: ";
cin>>choice;
switch(choice)
{
case 1:
cout<<"\n\nEnter number 1: ";
cin>>x;
cout<<"\nEnter number 2: ";
cin>>y;
c.add(x,y);
cout<<c.value;
break;
case 2:
cout<<"\n\nEnter number 1: ";
```

```
cin>>x;
cout<<"\nEnter number 2: ";
cin>>y;
c.subtract(x,y);
cout<<c.value;
break;
case 3:
cout<<"\n\nEnter number 1: ";
cin>>x;
cout<<"\nEnter number 2: ";
cin>>y;
c.multiply(x,y);
cout<<c.value;
break;
case 4:
cout<<"\n\nEnter number 1: ";
cin>>x;
cout<<"\nEnter number 2: ";
cin>>y;
c.divide(x,y);
cout<<c.value;
break;
case 5:
cout<<"\n\nEnter number 1: ";
```

```
cin>>x;
c.square(x);
cout<<c.value;
break;
case 6:
cout<<"\nEnter number 1: ";
cin>>y;
c.square_root(x);
cout<<c.value;
break;
case 7:
cout<<"\n\nEnter number 1: ";
cin>>x;
cout<<"\nEnter number 2: ";
cin>>y;
c.power(x,y);
cout<<c.value;
break;
}
}
}
```

**Output:**

Operations

1. .Add

2.Subtract

3.Multiply

4.Divide

5.Square

6.Square Root

7.Power

Enter choice

1

Enter number 1: 3

Enter number 2: 6

9

**Operations**

**1.** .Add

2.Subtract

3.Multiply

4.Divide

5.Square

6.Square Root

7.Power

Enter choice

3

Enter number 1: 3

Enter number 2: 5

15

Operations

1. .Add

2.Subtract

3.Multiply

4.Divide

5.Square

6.Square Root

7.Power

Enter choice

7

Enter number 1: 2

Enter number 2: 4

16

Operations

1. .Add

2.Subtract

3.Multiply

4.Divide

5.Square

6.Square Root

7.Power

Enter choice

8

## Program 2:

## Aim: Write a C++ program to calculate Prime Number Using Constructor

**Procedure:**

1. Declare the class as Prime with data members,Member functions.

2. To cal the function calculate() and do the following steps.

3. For i=2 to a/2 do

4. Check if a%i==0 then set k=0 and break.

5. If it is 1 then display the value is a prime number.

6. Else display the value is not prime.

**Program:**

```
#include<iostream.h>

#include<conio.h>

class prime

{

int a,k,i;

public:

prime(int x)

{

a=x;

}

void calculate()

{

k=1;

for(i=2;i<=a/2;i++)

{
```

```
if(a%i==0)
{
k=0;
break;
}
}
}
void show()
{
if(k==1)
cout<< "\n prime Number ";
else
cout<<"\n  Not prime Number";
}
};
void main()
{
    clrscr();
    int a;
    cout<<"\n\tEnter the Number:";
    cin>>a;
    prime p(a);
    p.calculate();
    p.show();
```

```
    getch();

}
```

**Output:**

Enter the number: 5

Prime number

## Program 3:

## Aim: Write a C++ program on Multiple Inheritance.

**Procedure:**

The class is derived from multiple base classes is called multiple inheritance.

1. In the given example program , **A** and **B** are the base classes and **C** is the derived class.
2. In the derived class , perform the addition operation on the variables **a** and **b** which belongs to class **A** and class **B** and store the result in **c** which belongs to class **C**.
3. Finally the result is stored in class **C** data member **c**.

**Program:**

```
#include<iostream.h>

#include<conio.h>

class A             // Base class 1

{

public:

int a;

void readA()

{

cout<<"\nEnter Number :";

cin>>a;

}

};
```

```
class B        //Base class 2
{
public:
int b;
void readB()
{
cout<<"\nEnter Number :";
cin>>b;
}
};
class C :public A,public B     // C is derived class
{
public:
int c;
void result()
{
c=a+b;
cout<<"\nResult :"<<c;
}
};
void main()
{
clrscr();
C c1;
```

```
c1.readA();

c1.readB();

c1.result();

getch();

}
```

**Output:**

```
Enter Number : 2

Enter Number : 3

Result : 5
```

## Program 4:

## Aim: Write a C++ program on overloading insertion and extraction operators.

**Procedure:**

1.Create class Complex which contains real, imag variables.

2.Overload the istream and ostream operators through operator>>() and operator<<().

3. We use in and out instead of cin and cout for reading and printing data.

Syntax for for overloading the stream extraction operator:

**Program:**

```
#include <iostream.h>

class Complex

{

private:

    int real, imag;

public:
```

```
    Complex(int r = 0, int i =0)
    {
    real = r;
    imag = i;
    }
    friend ostream& operator << (ostream &out, Complex &c);
    friend istream& operator >> (istream &in,  Complex &c);
};
ostream& operator << (ostream &out, Complex &c)
{
    out <<c.real;
    out << "+i"<<c.imag<<endl;
    return out;
}
istream& operator >> (istream &in,  Complex &c)
{
    cout<< "Enter Real Part ";
    in >>c.real;
    cout<< "Enter Imagenory Part ";
    in >>c.imag;
    return in;
}
void main()
{
```

```
   Complex c1;
   cin>> c1;
   cout<< "complex number :";
   cout<< c1;
}
```

**Output:**

Enter Real Part 2

Enter Imagenory Part 3

complex number : 2+i3

Program 5:

## Aim: Write a C++ program for class template

## Procedure:

**Program:**

```
#include<iostream.h>
#include<iomanip.h>
template<class s>
class stack
{
private:
int top;
s item;
s array[5];
public:
stack()
```

```
{
top=-1;
}
void push()
{
if(top==5)
cout<<"overflow"<<endl;
else
{
cout<<"enter item to push in stack"<<endl;
cin>>item;
top++;
array[top]=item;
}
}
void pop()
{
if(top==-1)
cout<<"underflow"<<endl;
else
{
array[top]=NULL;
top--;
}
```

```
}
void traverse()
{
for(int i=0;i<=top;i++)
 {
cout<<array[i]<<"\t";
 }
cout<<endl;
 }
};
void main()
{
stack <int>stacki;
stack <char>stackc;
int  r;
char ch;
do
    {
cout<<"Integer array"<<endl;
cout<<"1.Push"<<endl;
cout<<"2.Pop"<<endl;
cout<<"3.Traverse"<<endl;
cin>>r;
switch (r)
```

```
{
case 1:
stacki.push();
break;
case 2 :
stacki.pop();
break;
case 3:
stacki.traverse();
break;
default:
cout<<"wrong input"<<endl;
break;
            }
cout<<"do you want to continue y/n"<<endl;
cin>>ch;
    }
while(ch!='n');
do
{
cout<<"Character array"<<endl;
cout<<"1.Push"<<endl;
cout<<"2.Pop"<<endl;
cout<<"3.Traverse"<<endl;
```

```
cin>>r;
switch (r)
{
case 1:
stackc.push();
break;
case 2:
stackc.pop();
break;
case 3:
stackc.traverse();
break;
default:
cout<<"wrong input"<<endl;
break;
}
cout<<"do you want to continue y/n"<<endl;
cin>>ch;
}
while(ch!='n');
}
```

**Output:**

Integer array

1.Push

2.Pop

3.Traverse

1

Enter item to push into stack

10

do you want to continue

y

Integer array

1.Push

2.Pop

3.Traverse

2

Enter item to push into stack

20

do you want to continue

y

Integer array

1.Push

2.Pop

3.Traverse

3

10 20

do you want to continue

n

Character array

1.Push

2.Pop

3.Traverse

1

Enter item to push into stack

a

do you want to continue

y

Character array

1.Push

2.Pop

3.Traverse

2

Enter item to push into stack

b

do you want to continue

y

Character array

1.Push

2.Pop

3.Traverse

3

a b

do you want to continue

n

## Program 6

## Aim: Write a C++ program on pure virtual functions for decimal hexadecimal octal values

**Procedure:**

1.create a base class namely number and declare the pure virtual function show().

2.create the derived classes hextype, dectype& octtype from the class number and implement the show().

3. create the objects for the class dectype, hextype and octtype.

4. call the member function show();

5. Corresponding called function change the integer type as hex , oct and decimal .

**Program:**

```
#include<iostream.h>

class number

{

protected:

int value;

public :

void setvalue(int i)

{

value= i;

}

virtual void show() = 0;

};
```

```
class hextype : public number
{
public :
void show()
    {
cout<<"Hexa decimal value :";
cout<<hex<<value<<endl;
    }
};
class dectype : public number
{
public :
void show()
    {
    cout<<"Decimal value :";
cout<<value<<endl;
    }
};
class octtype : public number
{
public :
void show()
 {
```

```
 cout<<"Octal value :";
cout<<oct<<value<<endl;
}
};
void main()
 {
dectype d;
hextype h;
octtype o;
d.setvalue(29);
d.show();
h.setvalue(45);
h.show();
o.setvalue(28);
o.show();
}
```

**Output:**

Decimal value : 29

Hexa decimal value:2d

Octal value:34

## Program 7:

## Aim: Write a C++ program to copy the file in lower case into another file with upper case.

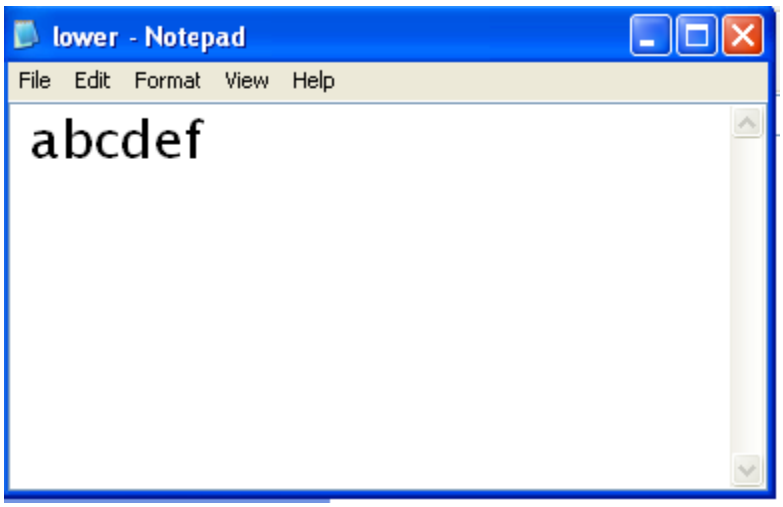
## Procedure:

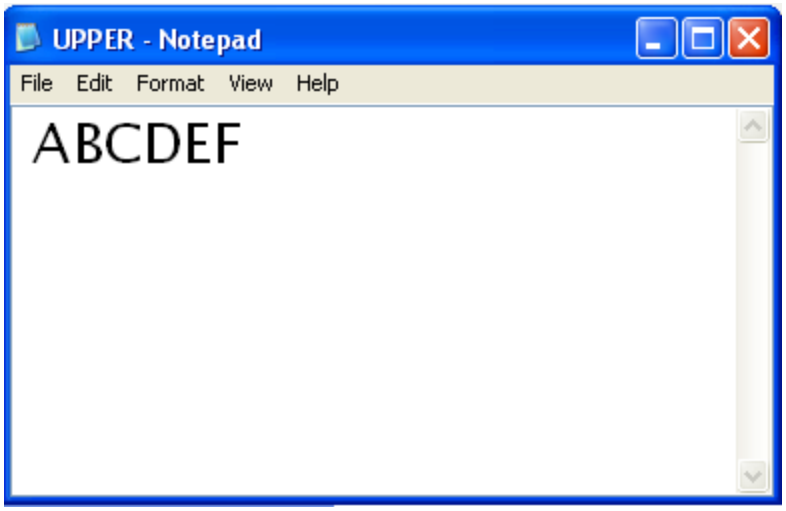
1. Create the input file **lower.dat** and output file **upper.dat**.
   2. Store the lower case letters in **lower.dat**.
   3. The uppercase letters are stored in **upper.dat**.
   4. Read data from **lower.dat** and store the data in **upper.dat**

**Program:**

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#include<ctype.h>
#include<fstream.h>

void main()
{
ofstream out;
ifstream in;
char ch,uch;
clrscr();
in.open("lower.dat");
out.open("upper.dat");
while(!in.eof())
{
 ch=(char)in.get();
 uch=toupper(ch);
 out.put(uch);
 }
 in.close();
out.close();
getch();
}
```

**OUTPUT:**






## CYCLE II: JAVA Programming:

## Program 8

## Aim: Write a java program Find Largest and Smallest Number in an Array

**Procedure:**

1 .**FindLarg** is a class and **numbers** is an array in FindLarg class.

2. Assign first number into **smallest** and **largest** variables.

3. Compare **smallest** with all the remaining numbers in an array and if the number in an array is smaller than **smallest** then assign that number to **smallest**.

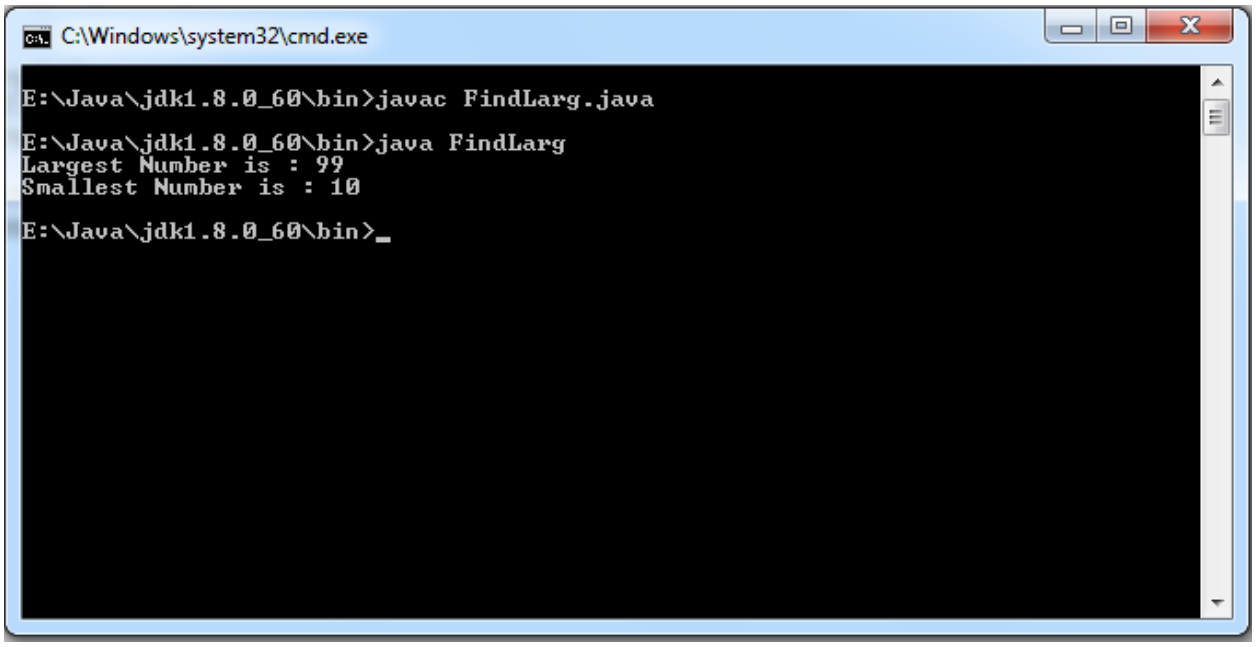
4. Compare **largest** with all the remaining numbers in an array and if the number in an array is larger than the value in the **largest** variable then assign that number to **largest**.

**Program:**

```
public class FindLarg

 {

 public static void main(String[] args)

 {

  int numbers[] = new int[]{20,50,10,70,30,60,90,80,40,99};

int smallest = numbers[0];
```

```
int largetst = numbers[0];

for(int i=1; i< numbers.length; i++)

{

if(numbers[i] > largetst)

        largetst = numbers[i];

else if (numbers[i] < smallest)

        smallest = numbers[i];

}

System.out.println("Largest Number is : " + largetst);

System.out.println("Smallest Number is : " + smallest);

 }

}
```

**Output:**




## Program 9

## Aim: Write a java program for string manipulation operations for string class

**Procedure:**

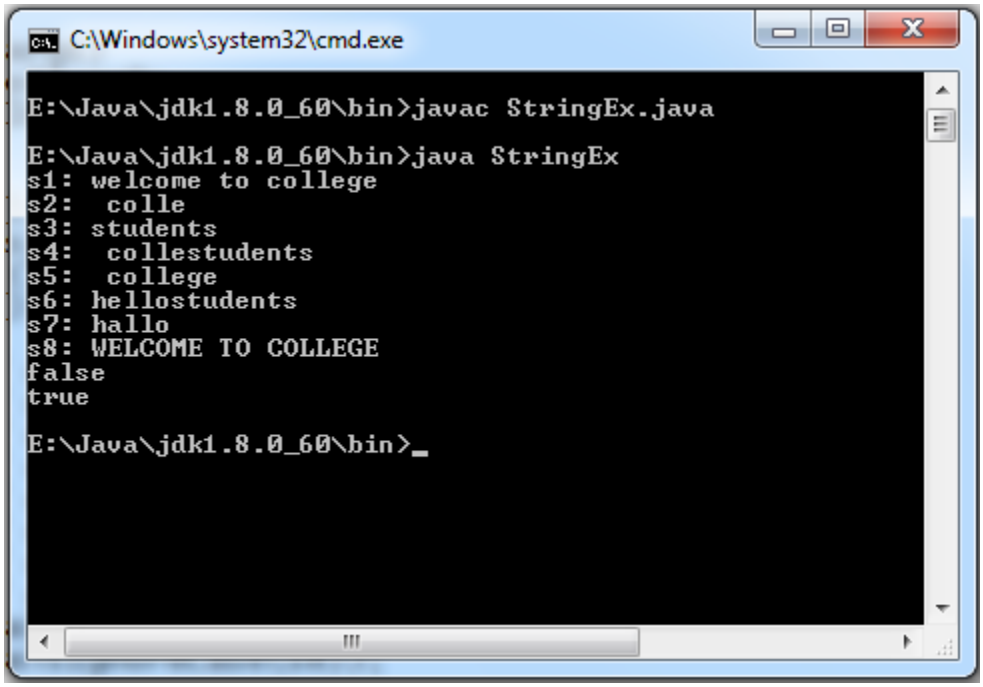
1. **String** is a predefined class and create constructors for **String** class and use different predefined functions like substring(), concat(), replace(), toLowerCase(), toUpperCase() etc in the **String** class.

**Program:**

```
public class StringEx

{       public static void main(String[] args)

        {       String s1 = "welcome to college";

                String s2 = s1.substring(10,16);

                String s3 = "students";

                String s4 = s2 + s3;

                String s5=s1.substring(10);

                String s6="hello".concat(s3);

                String s7="hello".replace("e","a");

                String s8=s1.toUpperCase();

                System.out.println("s1: " + s1);

                System.out.println("s2: " + s2);

                System.out.println("s3: " + s3);

                System.out.println("s4: " + s4);

                System.out.println("s5: " + s5);

                System.out.println("s6: " + s6);

                System.out.println("s7: " + s7);

                System.out.println("s8: " + s8);

                System.out.println(s1.equals(s8));
```

```
                System.out.println(s1.equalsIgnoreCase(s8));

        }

}
```

**Output:**




## Program 10:

## Aim: Write a java program for multiple threads.

**Procedure:**

1. We are creating **additionThread** and it implements Runnable interface and it performs addition of 1 to 5 numbers and store the result in sum.
2. We are creating **factThread** and it implements Runnable interface and it calculates factorial of 1 to 5 numbers and store the result in fact.
3. **yield()** method causes the currently executing thread object to temporarily pause and allow other threads to execute.

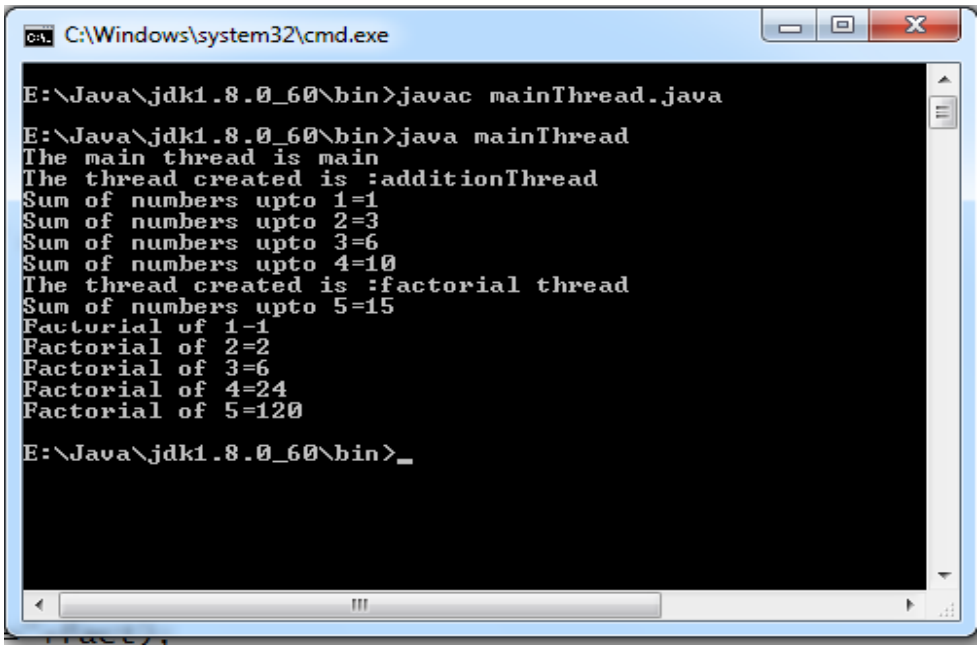
**Program:**

```
class additionThread implements Runnable

{

int i,sum=0;
```

```
public void run()
{
for (i=1;i<=5;i++)
{
sum=sum+i;
System.out.println("Sum of numbers upto " + i +"="+sum);
if (i==4)
                    Thread.yield();
}
}
}
class factThread implements Runnable
{
int i,n, fact=1;
public void run()
{
for( i=1;i<=5;i++)
{
fact=fact*i;
System.out.println("Factorial of " + i +"="+fact);
}
}
}
class mainThread
```

```
{
public static void main(String args[])
{
Thread ct=Thread.currentThread();
System.out.println("The main thread is "+ct.getName());
additionThread at=new additionThread();
factThread ft=new factThread();
Thread add=new Thread(at, "additionThread");
Thread factt=new Thread(ft,"factorial thread");
add.start();
System.out.println("The thread created is :" +add.getName());
factt.start();
System.out.println("The thread created is :" +factt.getName());
}
}
```

**Output:**




## Program 11

## Aim: Write a java program for handling mouse events

**Procedure:**

1. MouseEvents is a class which implements MouseListener and MouseMotionListener interfaces.
2. We must implement all methods in MouseListener and MouseMotionListener interfaces like mousePressed(), mouseReleased(),mouseClicked() etc. and perform all mouse events.

**Program:**

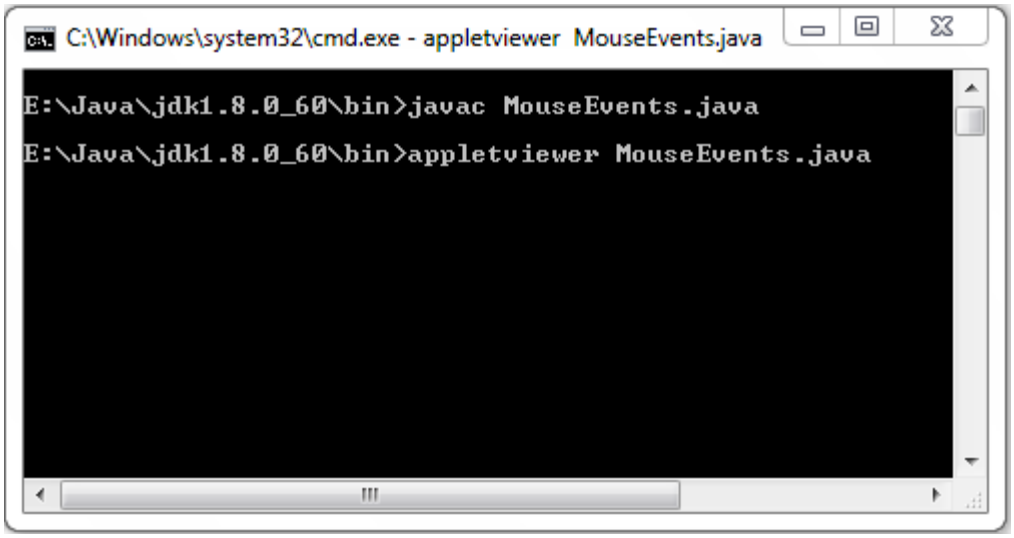
```
import java.awt.*;

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 = "nothing";
```

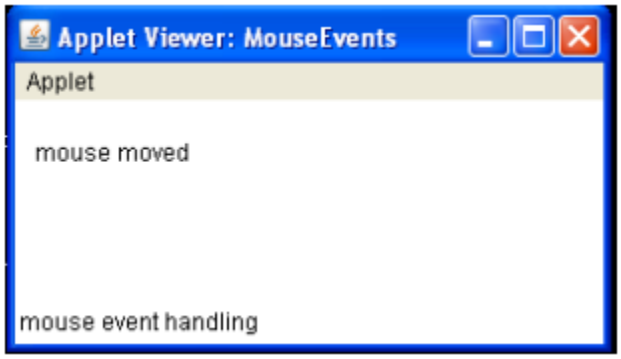
```
int X =10, Y =30; // coordinates of mouse
public void init()
 {
addMouseListener(this);
addMouseMotionListener(this);
}
// Handle mouse clicked.
public void mouseClicked(MouseEvent me)
{
msg = "Mouse clicked.";
repaint();
}
// Handle mouse entered.
public void mouseEntered(MouseEvent me)
 {
msg = "Mouse entered.";
repaint();
}
// Handle mouse exited.
public void mouseExited(MouseEvent me)
{
msg = "Mouse exited.";
repaint();
}
```

```
// Handle button pressed.
public void mousePressed(MouseEvent me)
{
msg = "Mouse pressed";
repaint();
}
// Handle button released.
public void mouseReleased(MouseEvent me)
 {
msg = "Mouse released";
repaint();
}
// Handle mouse dragged.
public void mouseDragged(MouseEvent me)
{
msg = "Mouse dragged";
repaint();
}
// Handle mouse moved.
public void mouseMoved(MouseEvent me)
{
msg="mouse moved";
repaint();
}
```

```
// Display msg in applet window at current X,Y location.

public void paint(Graphics g)

 {

g.drawString(msg, X, Y);

showStatus("mouse event handling");

}

}
```

**Output:**




## Program 12

## Aim: Write an applet program for simple calculator

**Procedure:**

1. The class **calc** is created and create labels, text fields and buttons for all the operations.
2. The class **calc** implements **actionPerformed** interface and we can perform all actions while clicking the button etc.

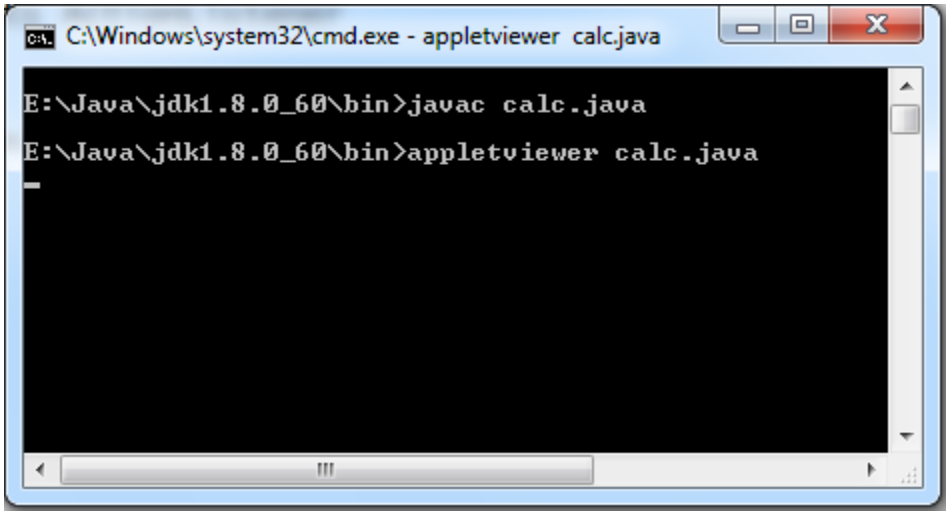
**Program:**

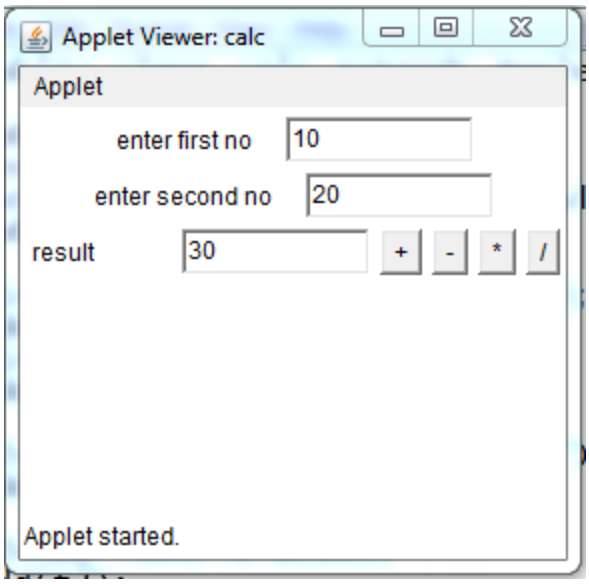
```
import java.awt.*;

import java.applet.*;

import java.awt.event.*;

public class calc extends Applet implements ActionListener

{

Label l1,l2,l3;

TextField t1,t2,t3;

Button addition,subtraction,multiplication,division;

public void init()

{

l1=new Label("enter first no");

add(l1);

t1=new TextField(10);

add(t1);

l2=new Label("enter second no");

add(l2);

t2=new TextField(10);

add(t2);

l3=new Label("result         ");

add(l3);

t3=new TextField(10);
```

```
add(t3);
addition=new Button("+");
add(addition);
addition.addActionListener(this);
subtraction=new Button("-");
add(subtraction);
subtraction.addActionListener(this);
multiplication=new Button("*");
add(multiplication);
multiplication.addActionListener(this);
division=new Button("/");
add(division);
division.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource()==addition)
{
int sum=Integer.parseInt(t1.getText()) + Integer.parseInt(t2.getText());
t3.setText(String.valueOf(sum));
}
if(ae.getSource()==subtraction)
{
int sub=Integer.parseInt(t1.getText()) + Integer.parseInt(t2.getText());
```

```
t3.setText(String.valueOf(sub));

}

if(ae.getSource()==multiplication)

{

int mul=Integer.parseInt(t1.getText()) + Integer.parseInt(t2.getText());

t3.setText(String.valueOf(mul));

}

if(ae.getSource()==division)

{

int div=Integer.parseInt(t1.getText()) + Integer.parseInt(t2.getText());

t3.setText(String.valueOf(div));

}

}

}

/*<applet code="calc" width=200 height=200>

</applet>*/
```

**Output:**






## Program13

## Aim: Write an applet program for GridLayout.

**Procedure:**

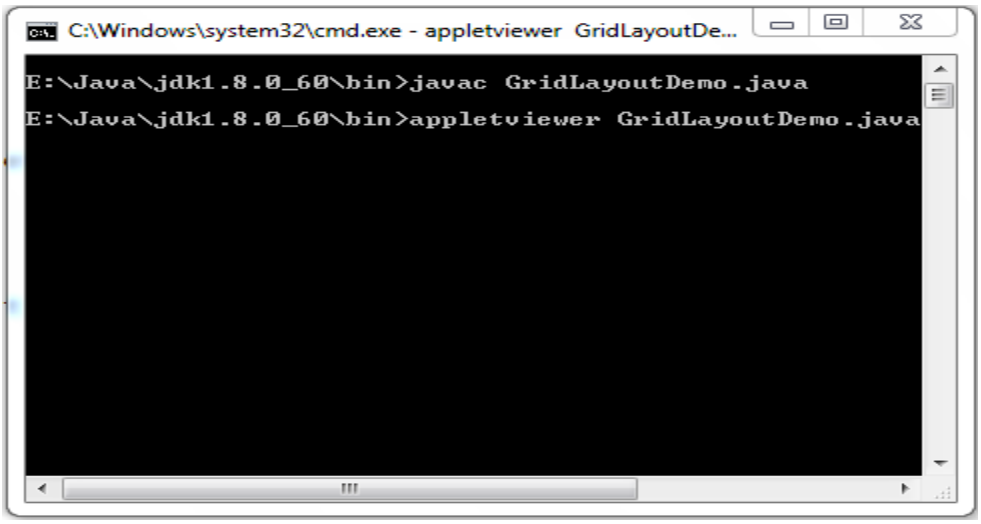
1. **GridLayoutDemo** is a class it creates 6 buttons .
2. A **GridLayout(2,3)** object places components in a grid of cells.
3. The 6 buttons are placed in the form of 2 rows and 3 columns.

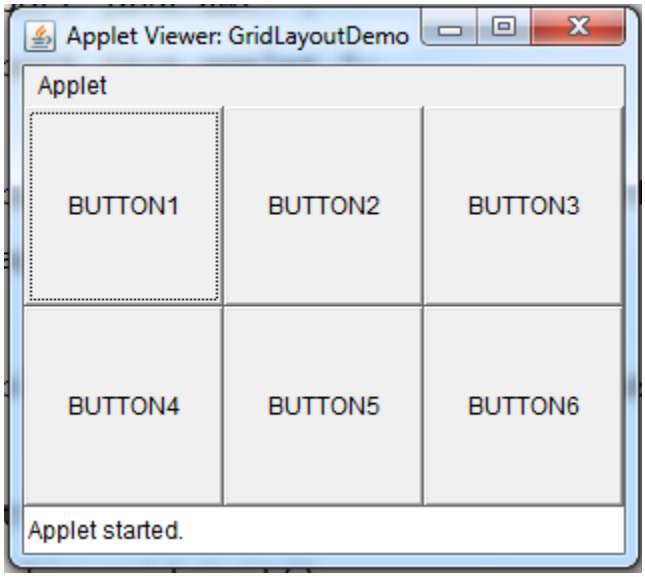
**Program:**

```
import java.awt.*;

import java.applet.*;

/*<applet code="GridLayoutDemo" width=300 height=200>

</applet>*/

public class GridLayoutDemo extends Applet
 {
Button b1,b2,b3,b4,b5,b6;
public void init()
 {
setLayout(new GridLayout(2, 3));
b1=new Button("BUTTON1");
b2=new Button("BUTTON2");
b3=new Button("BUTTON3");
b4=new Button("BUTTON4");
```

```
b5=new Button("BUTTON5");
b6=new Button("BUTTON6");
add(b1);
add(b2);
add(b3);
add(b4);
add(b5);
add(b6);
}
}
```

**Output:**




## Program 14:

## Aim: Write a swing program on JCheckBox

**Procedure:**

**1. JCheckBoxDemo** is a class which implements **ItemListener**.

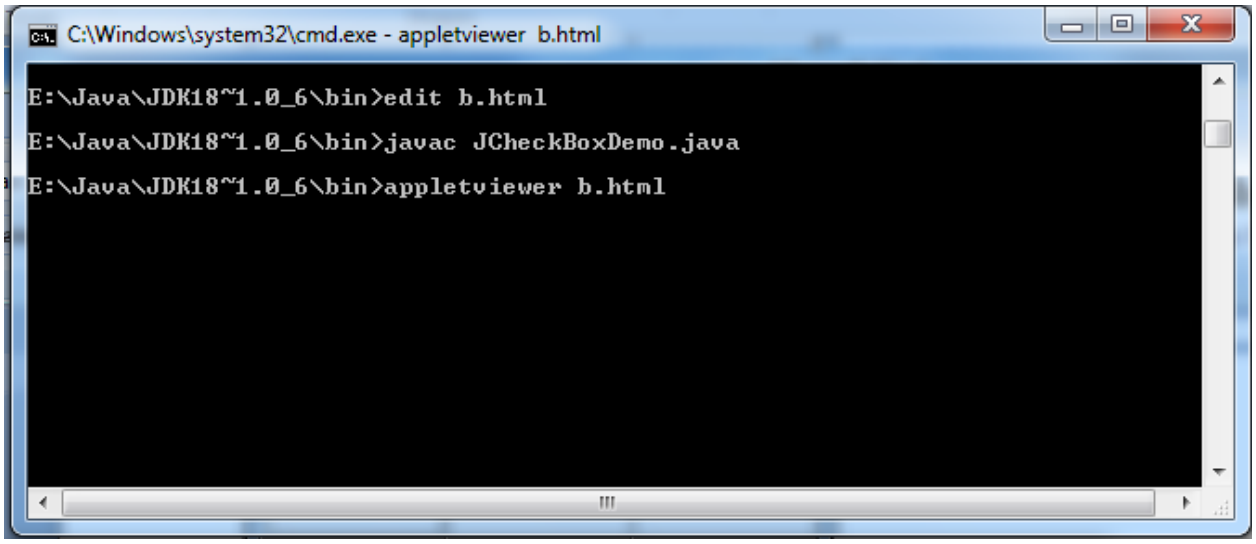
2. We create check boxes and we perform actions on those checkboxes.

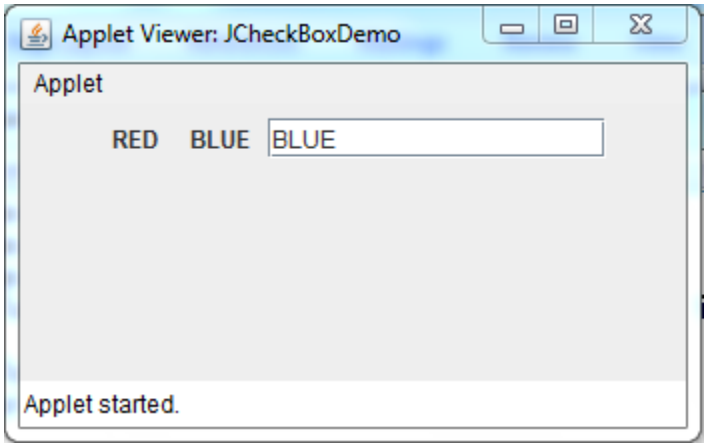
**Program:**

```
//JCheckBoxDemo.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JCheckBoxDemo extends JApplet implements ItemListener
{
JTextField jtf;
public void init()
{
        Container contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());
        ImageIcon img1 = new ImageIcon("b1.jpg");
        ImageIcon img2 = new ImageIcon("b2.jpg");
        ImageIcon img3 = new ImageIcon("b3.jpg");
        JCheckBox cb = new JCheckBox("RED", img1);
        cb.setRolloverIcon(img2);
        cb.setSelectedIcon(img3);
        cb.addItemListener(this);
        contentPane.add(cb);
        cb = new JCheckBox("BLUE", img1);
```

```
        cb.setRolloverIcon(img2);
        cb.setSelectedIcon(img3);
        cb.addItemListener(this);
        contentPane.add(cb);
        jtf = new JTextField(15);
        contentPane.add(jtf);
}
public void itemStateChanged(ItemEvent ie)
 {
        JCheckBox cb = (JCheckBox)ie.getItem();
        jtf.setText(cb.getText());
}
}
//b.html
<html>
<applet code=”JCheckBoxDemo” height=300 width=300>
</applet>
</html>
```

**Output:**




## Program15

## Aim: Write a swing program on JTabbedPane

**Procedure:**

1. With the JTabbedPane class, you can have several components, such as panels, share the same space. The user chooses which component to view by selecting the tab corresponding to the desired component.
2. In the given program we create 3 panels FruitsPanel, ColorsPanel and AnimalsPanel.
3. Based on the tab selection the corresponding panel will be selected.

**Program:**

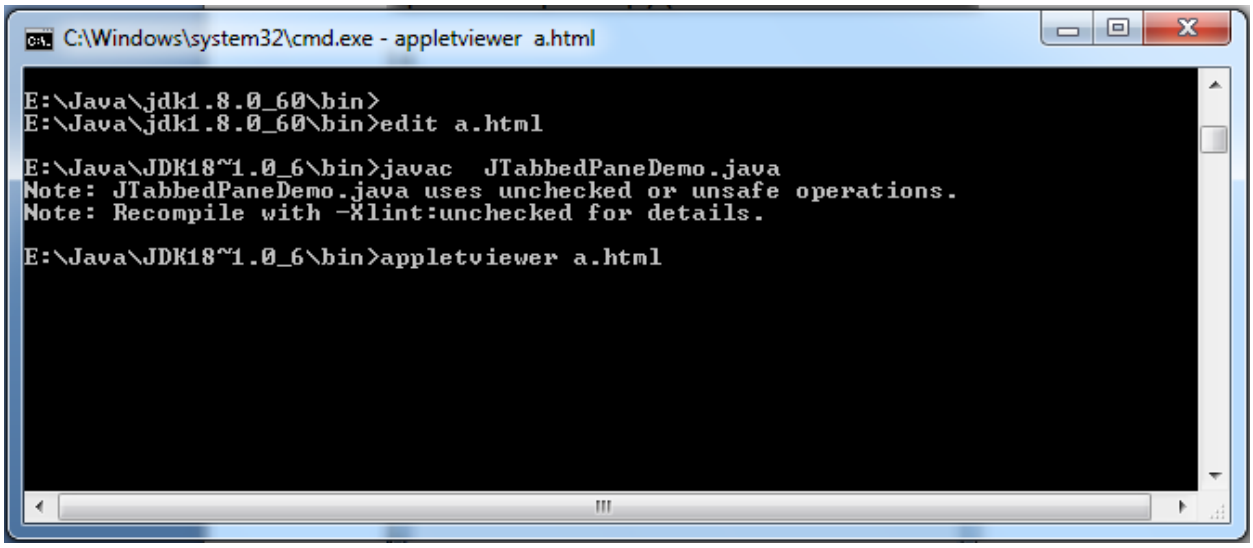
//JTabbedPaneDemo.java

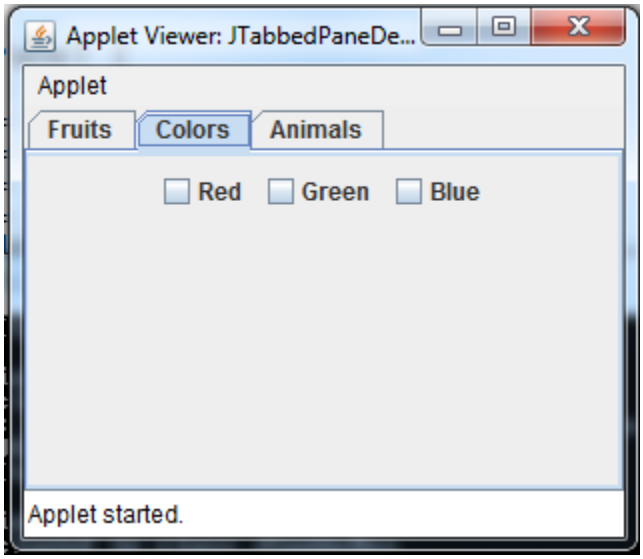
import javax.swing.*;

```
class FruitsPanel extends JPanel {
        public FruitsPanel() {
                JButton b1 = new JButton("Apple");
                JButton b2 = new JButton("Banana");
                JButton b3 = new JButton("Orange");
                JButton b4 = new JButton("Grapes");
                 add(b1); add(b2);
                add(b3); add(b4);
        }
}
class ColorsPanel extends JPanel {
        public ColorsPanel() {
                JCheckBox cb1 = new JCheckBox("Red");
                JCheckBox cb2 = new JCheckBox("Green");
                JCheckBox cb3 = new JCheckBox("Blue");
                add(cb1);
                add(cb2);
                add(cb3);
        }
}
class AnimalsPanel extends JPanel {
        public AnimalsPanel() {
                JComboBox jcb = new JComboBox();
                jcb.addItem("Tiger");
```

```
                jcb.addItem("Lion");
                jcb.addItem("Elephant");
                add(jcb);
        }
}
public class JTabbedPaneDemo extends JApplet {
public void init() {
        JTabbedPane jtp = new JTabbedPane();
        jtp.addTab("Fruits", new FruitsPanel());
        jtp.addTab("Colors", new ColorsPanel());
        jtp.addTab("Animals", new AnimalsPanel());
        getContentPane().add(jtp);
        }
}
//a.html
<html>
<applet code=”JTabbedPaneDemo” height=300 width=300>
</applet>
</html>
```

**Output:**


C:\Windows\system32\cmd.exe - appletviewer a.html
E:\Java\jdk1.8.0_60\bin>
E:\Java\jdk1.8.0_60\bin>edit a.html
E:\Java\JDK18~1.0_6\bin>javac JTabbedPaneDemo.java
Note: JTabbedPaneDemo.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
E:\Java\JDK18~1.0_6\bin>appletviewer a.html


Applet Viewer: JTabbedPaneDe...
Applet
Fruits
Colors
Animals
Red
Green
Blue
Applet started.