**PRACTICAL FILE**

**Java Programming**

**(BCA 252)**

**Bachelor of Computer Applications**

**Fourth Semester**

(BATCH: 2018-2021)

**SUBMITTED TO: SUBMITTED BY:**

Ms.Sakshi Khullar Shivankar sharma

Assistant Professor,VIPS 12417702018



**Vivekananda Institute of Professional Studies**

Guru Gobind Singh Indraprastha University,

New Delhi

| **S.No.** | **Program** | **Date** | **Teacher’s Sign** |
| --- | --- | --- | --- |
| **1** | 1. WAP to find out the factorial of a number through recursion. |  |  |
| **2** | 1. WAP to print Fibonacci series. |  |  |
| **3** | 1. WAP to accept Command line arguments & print them. |  |  |
| **4** | 1. WAP to obtain a number by a user & check if it’s prime or not. |  |  |
| **5** | 1. WAP that creates a class Accounts with following details:Instance variables: ac\_no., name, ac\_name, balance .Methods: withdrawal(), deposit(),display().Use constructors to initialize members. |  |  |
| **6** | 1. WAP to implement constructor overloading. |  |  |
| **7** | 1. WAP to count the no. of objects created in a program. |  |  |
| **8** | WAP to show call by value & call by reference. |  |  |
| **9** | 1. WAP to implement method overriding & method overloading. |  |  |
| **10** | 1. WAP that demonstrates all the usages of “super” keyword. |  |  |
| **11** | 1. Create a class box having height, width , depth as the instance variables & calculate its volume. Implement constructor overloading in it. Create a subclass named box\_new that has weight as an instance variable. Use super in the box\_new class to initialize members of the base class. |  |  |
| **12** | 1. WAP that implements multilevel inheritance. |  |  |
| **13** | 1. Consider a university where students who participate in the national games or Olympics are given some grace marks. Therefore, the final marks awarded = Exam\_Marks + Sports\_Grace\_Marks. A class diagram representing this scenario is as follow |  |  |

| **14** | WAP to implement Run time polymorphism. |  |  |
| --- | --- | --- | --- |
| **15** | 1. WAP to implement interface. Create an interface named Shape having area() & perimeter() as its methods. Create three classes circle, rectangle & square that implement this interface. |  |  |
| **16** | 1. WAP to show multiple inheritance. |  |  |
| **17** | 1. WAP to implement exception handling. The program should accept two numbers from the user & divide the first no. by the second. It should throw a Arithmetic Exception if an attempt is made to divide the no. by zero. Use try, catch & finally .Implement multicatch statements also |  |  |
| **18** | 1. Create a user defined exception named “NoMatchException” that is fired when the number entered by the user is not 10.Use the throws & throw keyword. |  |  |
| **19** | 1. WAP that creates three threads which print no.s from 1 to 5 .Set the name & priority of the threads. |  |  |
| **20** | 1. WAP to print even & odd numbers using threads. |  |  |
| **21** | 1. WAP that implements the concept of synchronization in threads. |  |  |
| **22** | 1. WAP that draws different colored shapes on an applet .Set the foreground & background color as red & blue. |  |  |
| **23** | 1. WAP to show moving banner by applet. |  |  |
| **24** | 1. WAP to implement Matrix multiplication by 2D array. |  |  |
| **25** | 1. WAP to implement Vector.[Use : addElement(),elementAt().removeElement(),size(). |  |  |
| **26** | 1. WAP to demonstrate the use of equals(), trim() ,length() , substring(), compareTo() of String class. |  |  |
| **27** | 1. WAP to implement file handling .The program should copy the content from one file to another. |  |  |
| **28** | 1. WAP to implement all mouse events and mouse motion events. |  |  |
| **29** | W WAP to implement keyboard events. |  |  |
| **30** | WAP using AWT to create a simple calculator. |  |  |
| **31** | 1. Write an applet that contains three choices and 30 \* 30 pixel canvas. The three check boxes should be labeled “red” ,”Green” and “Blue” .The selections of the choice should determine the color of the background. |  |  |
| **32** | Create a login form using AWT controls like labels,buttons, textboxes, checkboxes, list, checkboxgroup. The selected checkbox item names should be displayed |  |  |
| **33** | W WAP to display students information such as name,rollno, marks in an applet an applet window. Pass these parameters from an html file. |  |  |
| **34** | 1. WAP to show all Layout managers. (4 Layout managers). |  |  |
| **35** | 1. Create a simple JDBC program that displays & updates(insert or delete) the content of a table named employee having fields (id,name,deptt). |  |  |

1. WAP to find out factorial of a number through recursion

import java.io.\*;

import java.util.\*;

class factorial{

int fact=1;

public int fact(int n){

if(n==0){

return fact;

}

else{

fact\*=n;

return fact(n-1);

}

}

}

class first{

public static void main(String[] args){

int number;

factorial f=new factorial();

Scanner sc=new Scanner(System.in);

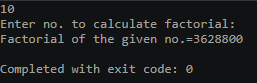
System.out.println("Enter no. to calculate factorial:");

number=sc.nextInt();

System.out.println("Factorial of the given no.="+f.fact(number));

}

}



2. WAP to print Fibonacci series.

import java.io.\*;

import java.util.\*;

class second{

public static void main(String[] args){

int t1=0,t2=1,t3,upto;

String series;

series=t1+" "+t2;

Scanner sc=new Scanner(System.in);

System.out.println("Enter range upto which fibonacci series is to be printed:");

upto=sc.nextInt();

t3=t2+t1;

while(t3<=upto){

series=series+" "+t3;

t1=t2;

t2=t3;

t3=t2+t1;

}

System.out.println("Series->"+series);

}

}



3. WAP to accept Command line arguments & print them.

import java.io.\*;

import java.util.\*;

class third{

public static void main(String[] args) {

int i=0;

System.out.println("Arguments entered:");

while(i<args.length){

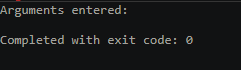
System.out.println(args[i]);

i++;

}

}

}



4. WAP to obtain a number by a user & check if it’s prime or not.

import java.io.\*;

import java.util.\*;

class Accounts{

private int ac\_no;

private String name,ac\_name;

private double balance;

Accounts(int ac,String n,String an,double bal){

ac\_no=ac;

name=n;

ac\_name=an;

balance=bal;

}

public void display(){

System.out.println("Account No.->"+ac\_no+"\nName->"+name+"\nAccount Type->"+ac\_name+"\nBalance->"+balance);

}

public void deposit(double amt){

balance+=amt;

display();

}

public void withdrawal(double amt){

balance-=amt;

display();

}

}

class fifth{

public static void main(String[] args) {

int choice;

double amount;

Scanner sc=new Scanner(System.in);

Accounts a1=new Accounts(7,"Cristiano Ronaldo","Savings",7000000);

a1.display();

System.out.println("Enter amount to be deposited:");

amount=sc.nextDouble();

a1.deposit(amount);

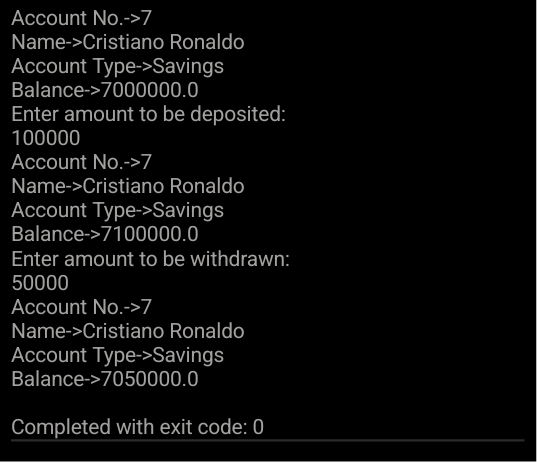
System.out.println("Enter amount to be withdrawn:");

amount=sc.nextDouble();

a1.withdrawal(amount);

}

}



5. WAP that creates a class Accounts with following details: Instance variables: ac\_no., name, ac\_name, balance .Methods: withdrawal(), deposit(),display().Use constructors to initialize members.

import java.io.\*;

import java.util.\*;

class Accounts{

private int ac\_no;

private String name,ac\_name;

private double balance;

Accounts(int ac,String n,String an,double bal){

ac\_no=ac;

name=n;

ac\_name=an;

balance=bal;

}

public void display(){

System.out.println("Account No.->"+ac\_no+"\nName->"+name+"\nAccount Type->"+ac\_name+"\nBalance->"+balance);

}

public void deposit(double amt){

balance+=amt;

display();

}

public void withdrawal(double amt){

balance-=amt;

display();

}

}

class fifth{

public static void main(String[] args) {

int choice;

double amount;

Scanner sc=new Scanner(System.in);

Accounts a1=new Accounts(7,"Cristiano Ronaldo","Savings",7000000);

a1.display();

System.out.println("Enter amount to be deposited:");

amount=sc.nextDouble();

a1.deposit(amount);

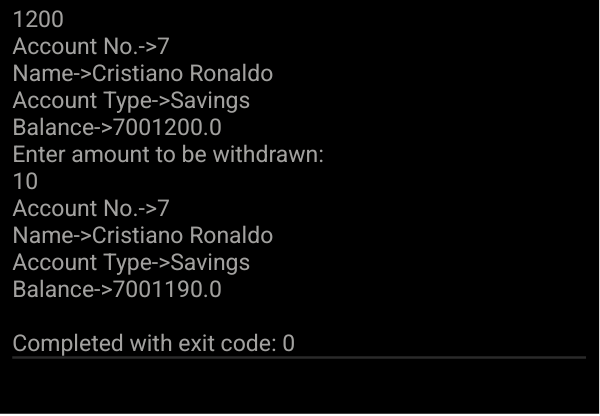
System.out.println("Enter amount to be withdrawn:");

amount=sc.nextDouble();

a1.withdrawal(amount);

}

}



6. WAP to implement constructor overloading.

import java.io.\*;

import java.util.\*;

class vehicle{

int no\_of\_wheels;

String color;

vehicle(String c){

no\_of\_wheels=2;

color=c;

}

vehicle(int n,String c){

no\_of\_wheels=n;

color=c;

}

void display(){

System.out.println("The vehicle has "+no\_of\_wheels+" wheels and is of "+color+" colour.");

}

}

class sixth{

public static void main(String[] args) {

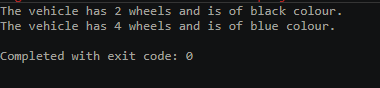
vehicle bike=new vehicle("black");

vehicle car=new vehicle(4,"blue");

bike.display();

car.display();

}}



7. WAP to count the no. of objects created in a program.

import java.io.\*;

import java.util.\*;

class count{

static int c=0;

count(){

c=c+1;

}

static void display(){

System.out.println("No. of objects created="+c);

}

}

class seventh{

public static void main(String[] args) {

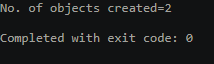
count c1=new count();

count c2=new count();

count.display();

}

}



8. WAP to show call by value & call by reference.

import java.io.\*;

import java.util.\*;

class pair{

int a,b;

pair(){a=10;b=20;}

}

class swap{

void swapval(int x,int y){

int temp=x;

x=y;

y=temp;

}

void swapref(pair ob){

int temp=ob.a;

ob.a=ob.b;

ob.b=temp;

}

}

class eighth{

public static void main(String[] args) {

pair p=new pair();

swap s=new swap();

System.out.println("Values of a and b initially:"+p.a+" "+p.b);

s.swapval(p.a,p.b);

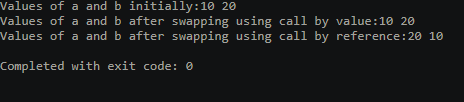
System.out.println("Values of a and b after swapping using call by value:"+p.a+" "+p.b);

s.swapref(p);

System.out.println("Values of a and b after swapping using call by reference:"+p.a+" "+p.b);

}

}



9. WAP to implement method over ridding & method overloading.

import java.io.\*;

import java.util.\*;

class shape{

void display(){

System.out.println("Class Shape.");

}

}

class polygon extends shape{

void display(){

System.out.println("Class Polygon");

}

void area(int side){

System.out.println("Area of the square="+(side\*side));

}

void area(int l,int b){

System.out.println("Area of the rectangle="+(l\*b));

}

}

class ninth{

public static void main(String[] args) {

shape s;

s=new shape();

s.display();

s=new polygon();

s.display();

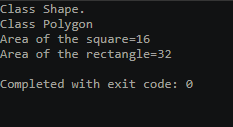
polygon p=new polygon();

p.area(4);

p.area(4,8);

}

}



10. WAP that demonstrates all the usages of “super” keyword.

import java.io.\*;

import java.util.\*;

class parent{

int agep;

parent(){

System.out.println("Parent Constructor");

}

}

class child extends parent{

int agec;

child(){

super();

System.out.println("Child constructor");

}

void getinpt(){

Scanner sc=new Scanner(System.in);

System.out.println("Enter Parent's age:");

super.agep=sc.nextInt();

System.out.println("Enter Child's age:");

agec=sc.nextInt();

}

void outpt(){

System.out.println("Parent's Age:"+super.agep+"\nChild's Age:"+agec);

}

}

class tenth{

public static void main(String[] args) {

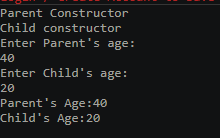
child c=new child();

c.getinpt();

c.outpt();

}

}



11. Create a class box having height, width , depth as the instance variables & calculate its volume. Implement constructor overloading in it. Create a subclass named box\_new that has weight as an instance variable. Use super in the box\_new class to initialize members of the base class.

class box{

private int height,width,depth;

box(int depth) {

this.height = 11;

this.width = 8;

this.depth = depth;

}

box(int height, int width) {

this.height = height;

this.width = width;

this.depth = 8;

}

box(int height, int width, int depth) {

this.height = height;

this.width = width;

this.depth = depth;

}

int volume (){

return (height\*width\*depth);}}

class box\_new extends box{

int depth;

box\_new(int depth) {

super(depth);

this.depth = depth;}}

public class Q11 {

public static void main(String[] args) {

box\_new ob1 = new box\_new(5);

box ob2 = new box(4,5,6);

box ob3 = new box(3,7);

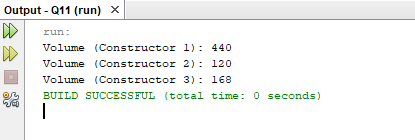
System.out.println("Volume (Constructor 1): "+ob1.volume());

System.out.println("Volume (Constructor 2): "+ob2.volume());

System.out.println("Volume (Constructor 3): "+ob3.volume());

}}

OUTPUT



12. WAP that implements multilevel inheritance.

import java.util.Scanner;

class College{

String CollegeName;

int building;

College() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter College Name: ");

CollegeName = sc.next();

System.out.print("Enter No. of Buildings: ");

building=sc.nextInt();

}

void DisplayCollege() {

System.out.println("\nCollege Name: "+CollegeName+"\nNo. Of Buildings: "+building);

}

}

class Course extends College{

String course;

int StudentCount;

Course() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Course Name: ");

course = scan.next();

System.out.print("Enter No. of students: ");

StudentCount = scan.nextInt();

}

void DisplayClasses() {

DisplayCollege();

System.out.println("Course Name:"+course+"\nStudent count:"+StudentCount);

}

}

class Student extends Course{

String Name;

int RollNum;

Student() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Student Name: ");

Name = scan.next();

System.out.print("Enter Roll Number: ");

RollNum = scan.nextInt();

}

void DisplayStudent() {

DisplayClasses();

System.out.println("Student Name:"+Name+"\nRoll Num:"+RollNum);

}

}

public class Q12 {

public static void main(String[] args) {

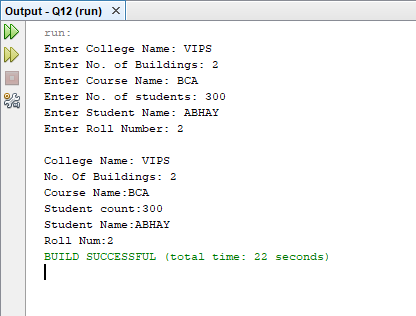
Student std = new Student();

std.DisplayStudent();

}

}

OUTPUT



Q13. Consider a university where students who participate in the national games or Olympics are given some grace marks. Therefore, the final marks awarded = Exam\_Marks + Sports\_Grace\_Marks.

import java.util.Scanner;

class Student {

String SName;

int Num;

Student() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Student name: ");

this.SName = scan.next();

System.out.print("Enter Student Roll Number: ");

Num = scan.nextInt();

}

void DisplayStudent(){

System.out.println("\nStudent Name: "+SName);

System.out.println("Student Number: "+Num);

}

}

class Exam extends Student{

int Total;

int Subject;

Exam() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Total Marks: ");

Total = scan.nextInt();

System.out.print("Enter No of Subjects: : ");

Subject = scan.nextInt();

}

void DisplayExam(){

DisplayStudent();

System.out.println("Total marks: "+Total+"\nTotal Subject: "+Subject);

}

void grace(int m){

Total += m;

}

}

interface Sports{

int gracemark = 50;

}

public class Result extends Exam implements Sports {

String sports = "None";

public static void main(String[] args) {

Exam ob = new Exam();

System.out.print("Plays Sports(true/false): ");

boolean ans = new Scanner(System.in).nextBoolean();

if(ans){

ob.grace(gracemark);

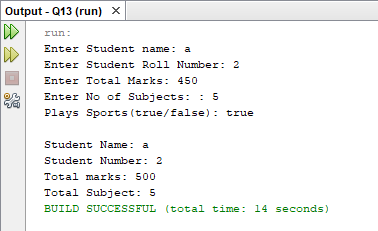
}

ob.DisplayExam();

}

}

OUTPUT



14. WAP to implement Run time polymorphism.

class A {

void callme() {

System.out.println("Inside A's callme method");}}

class B extends A {

void callme() {

System.out.println("Inside B's callme method");}}

class C extends B {

void callme() {

System.out.println("Inside C's callme method");}}

class Q14{

public static void main(String args[]) {

A a = new A(); // object of type A

B b = new B(); // object of type B

C c = new C(); // object of type C

A r; // obtain a reference of type A

r = a; // r refers to an A object

r.callme();

r = b;

r.callme();

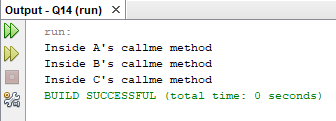
r = c;

r.callme();

}

}

OUTPUT



15. WAP to implement interface. Create an interface named Shape having area() & perimeter() as its methods. Create three classes circle, rectangle & square that implement this interface.

import java.util.Scanner;

interface Shape{

void area();

void perimeter();

}

class Circle implements Shape{

int radius,area,perimeter;

Circle() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Radius: ");

radius = scan.nextInt();

}

public void area() {

System.out.println("Area: "+Math.PI \* Math.pow(radius,2) );

}

public void perimeter() {

System.out.println("Perimeter: "+2 \* Math.PI \* radius );

}

}

class Rectangle implements Shape{

int Length,Width;

Rectangle() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Length: ");

Length = scan.nextInt();

System.out.print("Enter Width: ");

Width = scan.nextInt();;

}

public void area() {

System.out.println("Area: "+ Length \* Width );

}

public void perimeter() {

System.out.println("Perimeter: "+ ((2\*Length)+(2\*Width)) );

}

}

class Square implements Shape{

int Side;

Square() {

Scanner scan = new Scanner(System.in);

System.out.print("Enter Side: ");

Side = scan.nextInt();

}

public void area() {

System.out.println("Area: "+ 4 \* Side );

}

public void perimeter() {

System.out.println("Perimeter: "+ Math.pow(Side,2) );

}

}

public class Q15 {

public static void main(String[] args) {

System.out.println("Circle");

Circle circle = new Circle();

circle.area();

circle.perimeter();

System.out.println("\n\nRectangle");

Rectangle rectangle = new Rectangle();

rectangle.area();

rectangle.perimeter();

System.out.println("\n\nSquare");

Square square = new Square();

square.area();

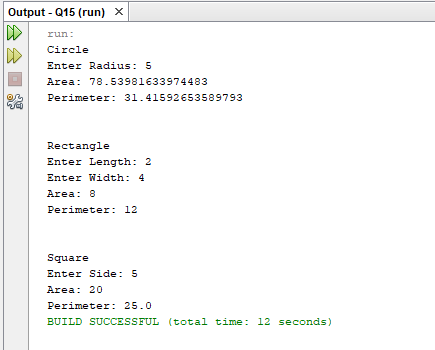
square.perimeter();

}

}

}

OUTPUT



1. WAP to show multiple inheritance.

interface adder

{

void add();

void sub();

}

interface multiplier

{void mul();

void div();}

class Mycal implements adder,multiplier

{

int a,b;

Mycal( int x, int y)

{ a=x; b=y;}

public void add ()

{ System.out.println("Sum of no.s is:" +(a+b));

}

public void sub()

{System.out.println("Subtraction is:" +(a-b));

}

public void mul()

{System.out.println("Multiplication is:" +(a\*b));

}

public void div()

{System.out.println("Division is:" +(a/b));

}}}}

class Q16 {

public static void main(String args[]){

Mycal m =new Mycal (20,10);

m.add();

m.sub();

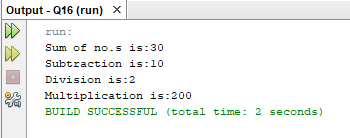
m.div();

m.mul():

}

}

OUTPUT



Q17. WAP to implement exception handling. The program should accept two numbers from the user & divide the first no. by the second. It should throw a Arithmetic Exception if an attempt is made to divide the no. by zero. Use try, catch & finally .Implement multicatch statements also.

import java.util.Scanner;

public class Q17 {

public static void main(String[] args) {

int x,y,z;

Scanner scan = new Scanner(System.in);

System.out.print("Enter x: ");

x = scan.nextInt();

System.out.print("Enter y: ");

y = scan.nextInt();

try {

z = x / y;

System.out.println("x/y = "+z);

}

catch (ArithmeticException ex){

System.out.println(ex);

}

catch (Exception ex) {

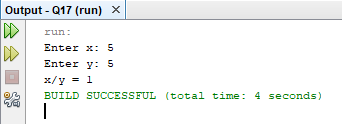
System.out.println(ex);

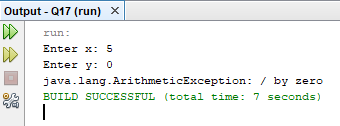
}

}

}

OUTPUT





Q18. Create a user defined exception named “NoMatchException” that is fired when the number entered by the user is not 10.Use the throws & throw keyword.

import java.util.Scanner;

class NoMatchException extends Exception{

public String toString(){

return ("Input is not equal to 10") ;

}

}

public class Q18 {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int inp;

try{

System.out.print("Enter a number: ");

inp = scan.nextInt();

if (inp != 10){

throw new NoMatchException();

}

} catch (NoMatchException e) {

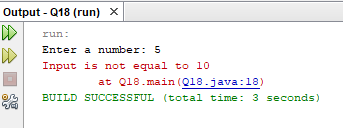
e.printStackTrace();

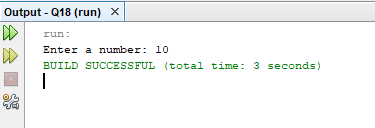
}

}

}

OUTPUT





Q19. WAP that creates three threads which print no.s from 1 to 5 .Set the name & priority of the threads.

class loop extends Thread{

String Name;

public loop(String name) {

Name = name;}

public void run() {

for(int i=0;i<5;i++){

System.out.println(Name+" : "+(i+1));

try {

Thread.sleep(1000);}

catch (InterruptedException e) {

e.printStackTrace();}}}}

public class Q19 {

public static void main(String[] args) {

loop ob = new loop("T1");

ob.setName("Thread 1");

ob.setPriority(1);

System.out.println(ob.getName()+" Priority: "+ob.getPriority());

ob.start();

loop ob2 = new loop("T2");

ob2.setName("Thread 2");

ob2.setPriority(5);

System.out.println(ob2.getName()+" Priority: "+ob2.getPriority());

ob2.start();

loop ob3 = new loop("T3");

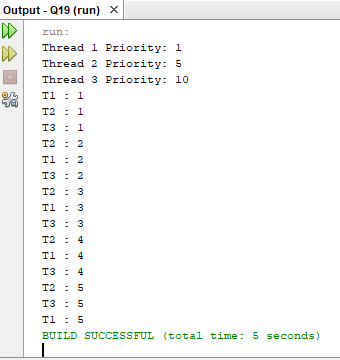
ob3.setName("Thread 3");

ob3.setPriority(10);

System.out.println(ob3.getName()+" Priority: "+ob3.getPriority());

ob3.start();}}

OUTPUT



Q20. WAP to print even & odd numbers using threads.

class Even extends Thread{

int limit;

Even(int limit){

this.limit = limit;}

public void run() {

for(int i=2;i<limit;i+=2){

System.out.println("Even: "+i);

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();}}}}

class Odd extends Thread{

int limit;

Odd(int limit){

this.limit = limit;}

public void run() {

for(int i=1;i<limit;i+=2){

System.out.println("Odd: "+(i));

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class Q20 {

public static void main(String[] args) {

Odd odd = new Odd(20);

Even even = new Even(20);

System.out.println("Odd Thread");

odd.start();

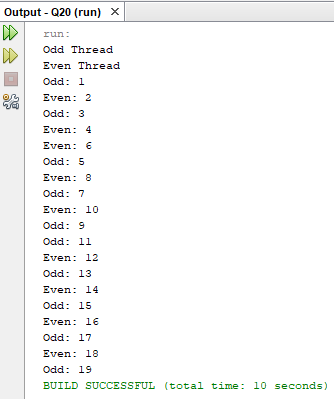
System.out.println("Even Thread");

even.start();

}

}

OUTPUT



Q21. WAP that implements the concept of synchronization in threads.

class Buying implements Runnable{

int item\_avail=5;

synchronized public void run(){

{

System.out.println("Items available:"+item\_avail);

if(item\_avail!=0){

System.out.println("order placed successfully");

try{

Thread.sleep(1000);

item\_avail=0;

}

catch(InterruptedException e){}

}

else

System.out.println("Sorry, out of stock");}}}

class Synch\_demo{

public static void main (String []args){

Buying ob=new Buying();

Thread t1= new Thread(ob);

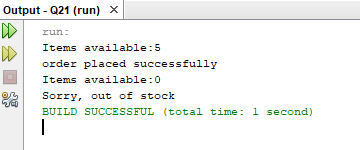
Thread t2= new Thread(ob);

t1.start();

t2.start();

}}

OUTPUT



Q22. WAP that draws different colored shapes on an applet .Set the foreground & background color as red & blue.

import java.applet.Applet;

import java.awt.\*;

public class Q22 extends Applet {

public void paint(Graphics g) {

setBackground(Color.BLUE);

setForeground(Color.RED);

g.fillOval(20,250,50,50);

g.fillRect(20,150,60,50);

g.drawLine(20,50,100,90);

g.fillArc(20,350,70,70,0,180);

}

}

HTML

<html>

<head>

<title>Q22</title>

</head>

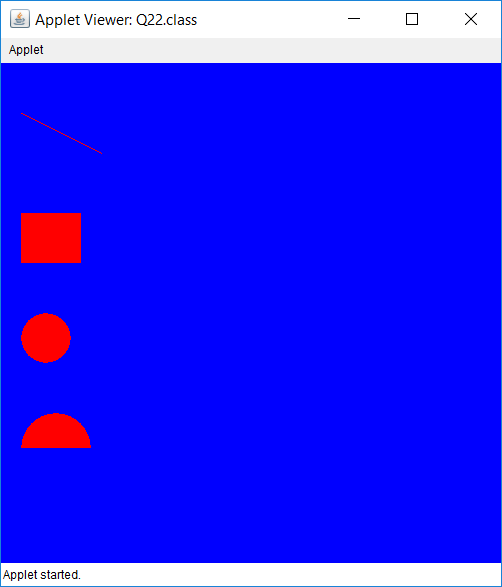
<body>

<applet code="Q22.class" width=500 height=500 ></applet>

</body>

</html>

OUTPUT



Q23. WAP to show moving banner by applet.

import java.applet.Applet;

import java.awt.\*;

public class Q23 extends Applet implements Runnable{

String text = "Moving Banner ";

boolean run = true;

public void init() {

setBackground(Color.black);

setForeground(Color.white);

}

public void start() {

Thread t1 = new Thread(this);

t1.start();

}

public void stop() {

run = false;

}

public void paint(Graphics g) {

g.drawString(text,250,250);

}

public void run() {

while (run){

char temp = text.charAt(0);

text = text.substring(1);

text += temp;

repaint();

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}}}}

HTML

<html>

<head>

<title>Q23</title>

</head>

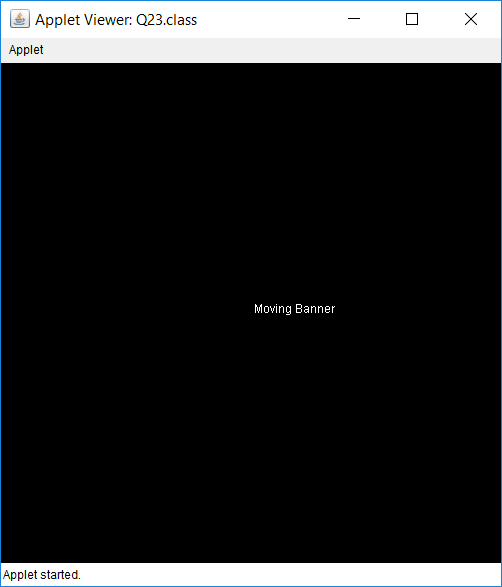
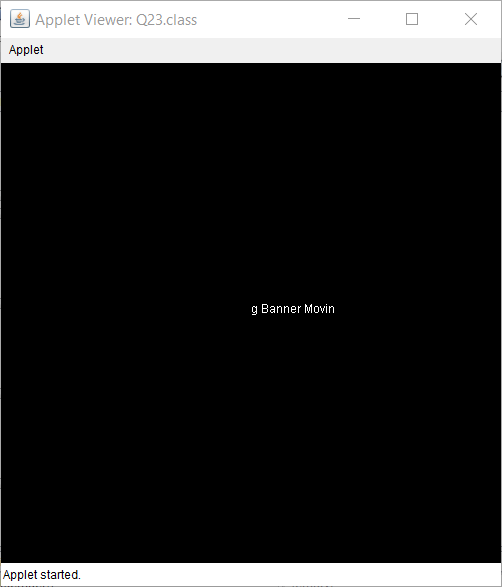
<body>

<applet code="Q23.class" width=500 height=500 ></applet>

</body>

</html>

OUTPUT



Q24. WAP to implement Matrix multiplication by 2D array.

public class Q24{

public static void main(String args[]){

int a[][]={{1,1,1},{2,2,2},{3,3,3}};

int b[][]={{1,1,1},{2,2,2},{3,3,3}};

int c[][]=new int[3][3];

System.out.println("\nMatrix 1");

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(a[i][j]+" ");}

System.out.println();}

System.out.println("\nMatrix 2");

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(b[i][j]+" ");}

System.out.println();}

System.out.println("\nResult");

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

c[i][j]=0;

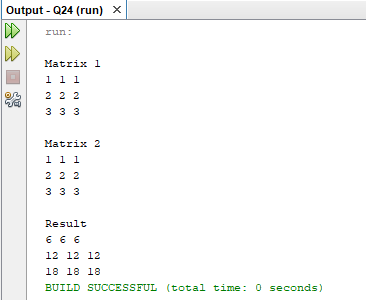
for(int k=0;k<3;k++)

{c[i][j]+=a[i][k]\*b[k][j];}

System.out.print(c[i][j]+" ");}

System.out.println();}}}

OUTPUT



Q25. WAP to implement Vector.[Use : addElement(), elementAt(), removeElement(), size().

import java.util.Scanner;

import java.util.Vector;

public class Q25 {

public static void main(String[] args) {

Vector list = new Vector();

int inp = 0;

Scanner scanner = new Scanner(System.in);

Outer : while (true){

System.out.print(" \nVector \n1.Add Element\n2.Element At\n3.Remove At\n4.Size\n5.Exit");

System.out.print("\nOption:");

inp = scanner.nextInt();

switch (inp)

{

case 1: System.out.print("Enter new Element: ");

list.add(scanner.next());

break;

case 2: System.out.print("Enter Index: ");

System.out.println(list.elementAt(scanner.nextInt()));

break;

case 3: System.out.print("Enter Index to Remove Element: ");

System.out.println(list.remove(scanner.nextInt()));

break;

case 4: System.out.println("Size="+list.size());

break;

case 5: break Outer;

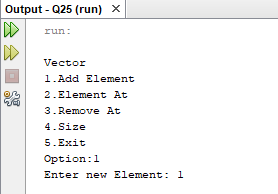
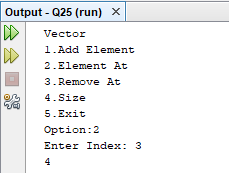
}

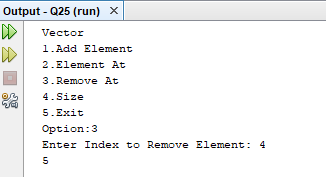
}

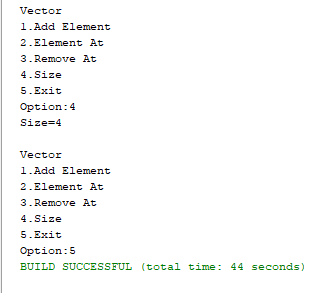
}

}

OUTPUT





Q26. WAP to demonstrate the use of equals(), trim() ,length() , substring(), compareTo() of String class.

public class Q26 {

public static void main(String[] args) {

String str1 = "Avengers: Endgame";

String str2 = "Avengers: Infinity War";

System.out.println("String 1: "+str1);

System.out.println("String 2: "+str2);

if (str1.equals(str2)){

System.out.println("Strings are Equal");

}

else {

System.out.println("Strings are Un-Equal");

}

str1 = str1.trim();

System.out.println("Trim String 1: "+str1);

System.out.println("String 1 Length: "+str1.length());

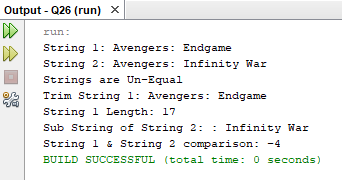
System.out.println("Sub String of String 2: "+str2.substring(8));

System.out.println("String 1 & String 2 comparison: "+str1.compareTo(str2));

}

}

OUTPUT



Q27. WAP to implement file handling .The program should copy the content from one file to another.

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class Q27 {

public static void main(String[] args) throws IOException {

FileReader fr = null;

try {

fr = new FileReader("D:\\old.txt");

} catch (FileNotFoundException e) {

e.printStackTrace();

}

int i;

FileWriter fw=new FileWriter("D:\\new.txt");

while ((i=fr.read()) != -1){

fw.write(i);

}

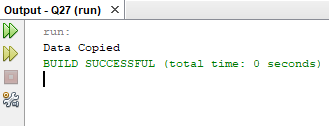
System.out.println("Data Copied");

fw.close();

}

}

OUTPUT



Q29. WAP to implement keyboard events.

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

public class Q29 extends Applet implements KeyListener {

String Str = "String: ";

String Key;

public void init() {

addKeyListener(this);

}

public void paint(Graphics g) {

g.drawString(Str,0,50);

g.drawString(Key,0,100);

}

public void keyTyped(KeyEvent e) {

Key = "Typed: " + e.getKeyChar();

repaint();

}

public void keyPressed(KeyEvent e) {

Key = "Pressed: " + e.getKeyChar();

repaint();

}

public void keyReleased(KeyEvent e) {

Key = "Released" + e.getKeyChar();

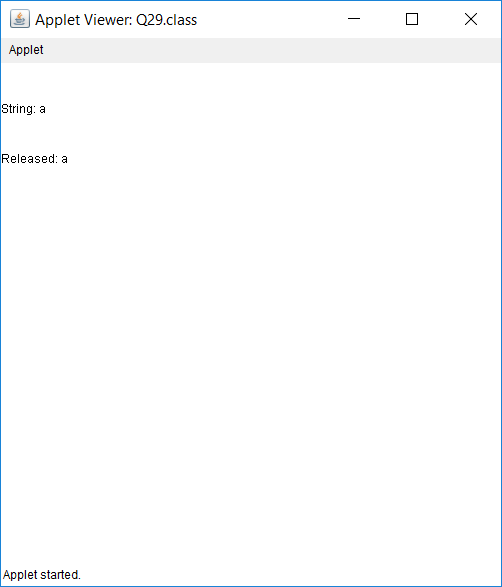
Str += e.getKeyChar();

repaint();

}

}

OUTPUT



Q30. WAP using AWT to create a simple calculator.

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Q30 extends Applet implements ActionListener {

TextField t1,t2;

Label result;

Button Add,Sub,Mul,Div;

public void init() {

t1 = new TextField(10);

add(t1);

t2 = new TextField(10);

add(t2);

result = new Label("Answer is: ");

Add = new Button("+");

Sub = new Button("-");

Mul = new Button("\*");

Div = new Button("/");

Add.addActionListener(this);

add(Add);

Sub.addActionListener(this);

add(Sub);

Mul.addActionListener(this);

add(Mul);

Div.addActionListener(this);

add(Div);

add(result);

}

public void actionPerformed(ActionEvent e) {

String temp = "Answer is: ";

if(e.getActionCommand().equals("+")) {

temp += String.valueOf(Integer.parseInt(t1.getText()) + Integer.parseInt(t2.getText()));

}

else if(e.getActionCommand().equals("-")) {

temp += String.valueOf(Integer.parseInt(t1.getText()) - Integer.parseInt(t2.getText()));

}

else if(e.getActionCommand().equals("\*")) {

temp += String.valueOf(Integer.parseInt(t1.getText()) \* Integer.parseInt(t2.getText()));

}

else if(e.getActionCommand().equals("/")) {

temp += String.valueOf(Integer.parseInt(t1.getText()) / Integer.parseInt(t2.getText()));

}

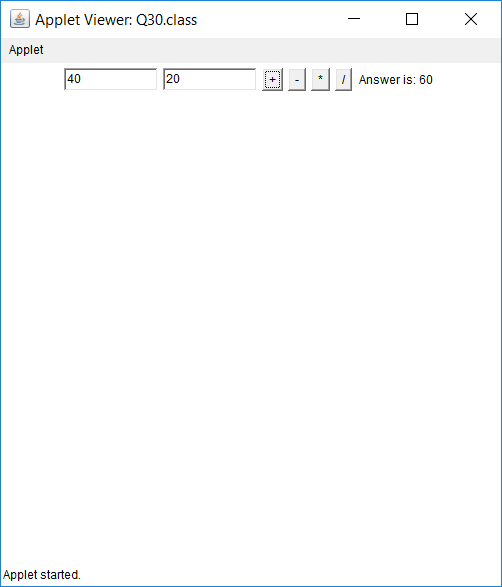
result.setText(temp);

repaint();

}

}

OUTPUT



Q31. Write an applet that contains three choices and 30 \* 30 pixel canvas. The three check boxes should be labeled “red” ,”Green” and “Blue” .The selections of the choice should determine the color of the background.

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

public class Q31 extends Applet implements ItemListener {

Canvas c1;

CheckboxGroup checkboxGroup;

@Override

public void init() {

checkboxGroup = new CheckboxGroup();

Checkbox Red = new Checkbox("Red",checkboxGroup,true);

Checkbox Blue = new Checkbox("Blue",checkboxGroup,false);

Checkbox Green = new Checkbox("Green",checkboxGroup,false);

c1 = new Canvas();

c1.setSize(400,400)

c1.setBackground(Color.RED);

Red.addItemListener(this);

Blue.addItemListener(this);

Green.addItemListener(this);

add(c1);

add(Red);

add(Blue);

add(Green);

}

public void itemStateChanged(ItemEvent e) {

if (checkboxGroup.getSelectedCheckbox().getLabel().equals("Red")){

c1.setBackground(Color.RED);

}

else if (checkboxGroup.getSelectedCheckbox().getLabel().equals("Blue")){

c1.setBackground(Color.BLUE);

}

else if (checkboxGroup.getSelectedCheckbox().getLabel().equals("Green")){

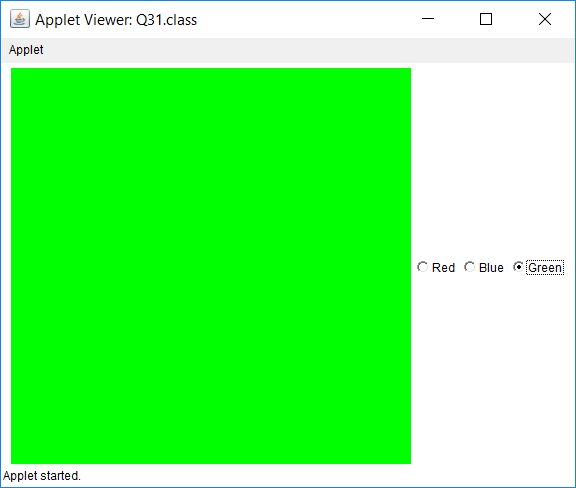
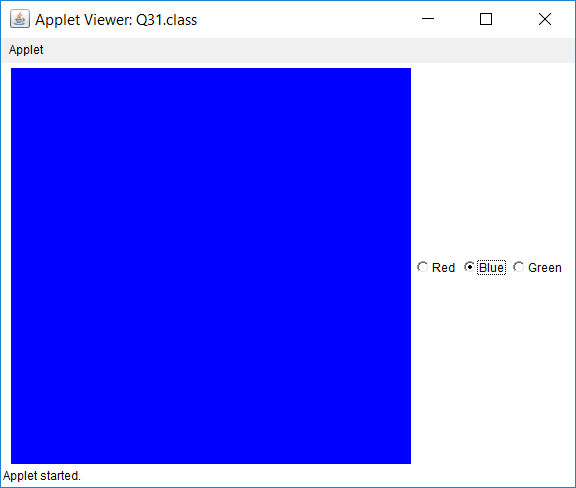
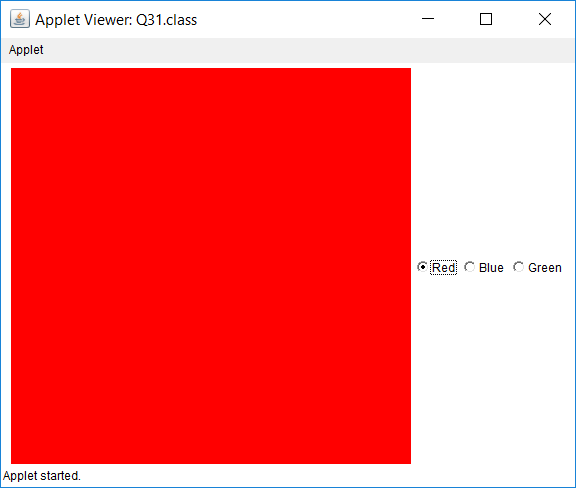
c1.setBackground(Color.GREEN);

}

}

}

OUTPUT



Q32. Create a login form using AWT controls like labels,buttons, textboxes, checkboxes, list, checkboxgroup. The selected checkbox item names should be displayed.

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

public class Q32 extends Applet implements ItemListener, ActionListener {

CheckboxGroup Gender;

Label LabelGender,LabelTier,LabelLimit;

Choice Sub;

List Limit;

public void init() {

Label Name = new Label("Enter Name: ");

add(Name);

TextField NameInp = new TextField();

add(NameInp);

Gender = new CheckboxGroup();

Checkbox Male = new Checkbox("Male",Gender,false);

Checkbox Female = new Checkbox("Female",Gender,false);

add(Male);

Male.addItemListener(this);

add(Female);

Female.addItemListener(this);

Sub = new Choice();

Sub.add("Free Tier");

Sub.add("Paid Tier");

Sub.add("Deluxe TIer");

add(Sub);

Limit = new List(5);

Limit.add("1 Year");

Limit.add("2 Year");

Limit.add("3 Year");

Limit.add("4 Year");

Limit.add("5 Year");

add(Limit);

Limit.addItemListener(this);

LabelGender = new Label("Gender");

LabelTier = new Label("Free Tier");

LabelLimit = new Label("No Limit");

add(LabelGender);

add(LabelTier);

add(LabelLimit);

Button Refresh = new Button("Refresh");

Refresh.addActionListener(this);

add(Refresh);

setLayout(new GridLayout(5,2));

setSize(250,200);

}

public void itemStateChanged(ItemEvent e) {

LabelGender.setText(Gender.getSelectedCheckbox().getLabel());

}

public void actionPerformed(ActionEvent e) {

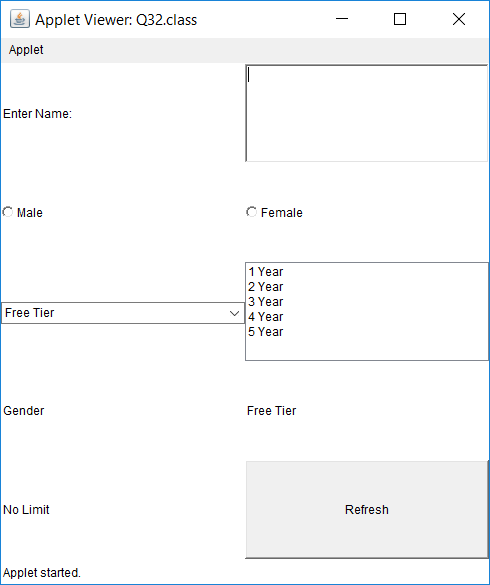
LabelTier.setText(Sub.getSelectedItem());

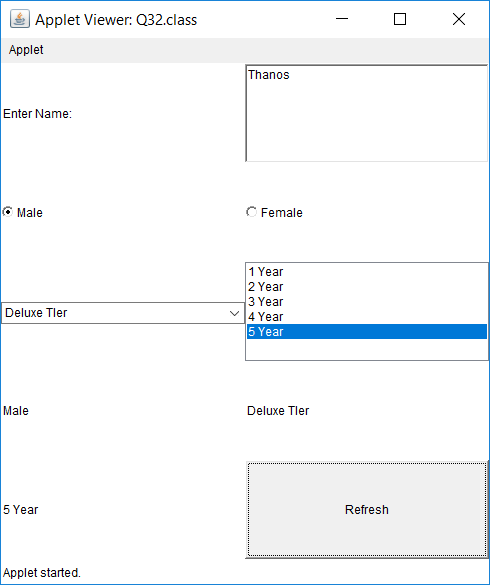
LabelLimit.setText(Limit.getSelectedItem());

}

}

OUTPUT





Q34. WAP to show all Layout managers. (4 Layout managers).

import javax.swing.\*;

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Q34 extends Applet implements ActionListener {

CardLayout CardLay = new CardLayout();

JPanel Card;

@Override

public void init() {

JPanel Flow =new JPanel();

Flow.setLayout(new FlowLayout());

Button b1 = new Button("1");

Button b2 = new Button("2");

Button b3 = new Button("3");

Button b4 = new Button("4");

Flow.add(b1);

Flow.add(b2);

Flow.add(b3);

Flow.add(b4);

JPanel Grid = new JPanel();

Grid.setLayout(new GridLayout(3,2));

Button Gridb1 = new Button("1");

Button Gridb2 = new Button("2");

Button Gridb3 = new Button("3");

Button Gridb4 = new Button("4");

Grid.add(Gridb1);

Grid.add(Gridb2);

Grid.add(Gridb3);

Grid.add(Gridb4);

Card = new JPanel();

Card.setLayout(CardLay);

Button Cardb1 = new Button("1");

Button Cardb2 = new Button("2");

Button Cardb3 = new Button("3");

Button Cardb4 = new Button("4");

Cardb1.addActionListener(this);

Cardb2.addActionListener(this);

Cardb3.addActionListener(this);

Cardb4.addActionListener(this);

Card.add("a",Cardb1);

Card.add("b",Cardb2);

Card.add("c",Cardb3);

Card.add("d",Cardb4);

JPanel Border = new JPanel();

Border.setLayout(new BorderLayout());

Button Borderb1 = new Button("1");

Button Borderb2 = new Button("2");

Button Borderb3 = new Button("3");

Button Borderb4 = new Button("4");

Border.add(Borderb1, BorderLayout.EAST);

Border.add(Borderb2, BorderLayout.CENTER);

Border.add(Borderb3, BorderLayout.WEST);

Border.add(Borderb4, BorderLayout.NORTH);

add(new Label("FlowLayout"));

add(Flow);

add(new Label("GridLayout"));

add(Grid);

add(new Label("CardLayout"));

add(Card);

add(new Label("BorderLayout"));

add(Border);

setLayout(new GridLayout(8,1));

}

public void actionPerformed(ActionEvent e) {

CardLay.next(Card);

}

}

OUTPUT

