

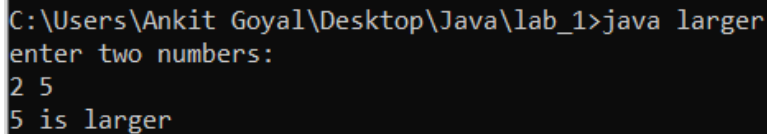
Practical No. 1

1. Write an application that ask the user to enter 2 integers, obtain them from user and displays a large number followed by the work “is larger”. If the number are equal, print the message “These numbers are equal”.

```
package lab_1;
import java.util.Scanner;

public class larger{
    public static void main(String[] args) {
        System.out.println("enter two numbers: ");
        Scanner in = new Scanner(System.in);
        int num1 = in.nextInt();
        int num2 = in.nextInt();
        if(num1>num2)
            System.out.println(num1 + " is larger");
        else if(num2>num1)
            System.out.println(num2 + " is larger");
        else
            System.out.println("These numbers are equal");
    }
}
```

Output:



```
C:\Users\Ankit Goyal\Desktop\Java\lab_1>java larger
enter two numbers:
2 5
5 is larger
```

2. Write an application in which user passes Command Line arguments and the output shows the count of the arguments passed on command line and also print them all.

```
public class count {
    public static void main(String[] args){
        int cnt=args.length;
        System.out.println("no. of arguments is " + cnt + "\n");
        for(int i=0;i<cnt;i++){
            System.out.println(args[i] + " ");
        }
    }
}
```

```

        }
    }
}

```

Output:

```

C:\Users\Ankit Goyal\Desktop\Java\lab_1>java Count 4 5 7 ankit
no. of arguments is 4

4
5
7
ankit

```

3. Write an application to implement Pascal Triangle.

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

.....

.....

.....

```

package lab_1;
import java.util.Scanner;
public class PascalTriangle {
    public static int fact(int n)
    {
        if(n==0)
            return 1;
        else
            return n*fact(n-1);
    }
    public static int ncr(int n, int r)
    {
        int res;
        res=fact(n)/(fact(r)*fact(n-r));
        return res;
    }
}

```

```

public static void main(String args[])
{
    System.out.print("enter no. of rows: ");
    Scanner in=new Scanner(System.in);
    int row=in.nextInt();
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<row-i-1;j++)
            System.out.print(" ");
        for(int j=0;j<=i;j++)
        {
            System.out.print(ncr(i,j)+ " ");
        }
        System.out.println();
    }
}

```

Output:

```

C:\Users\Ankit Goyal\Desktop\Java\lab_1>java PascalTriangle
enter no. of rows: 5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1

```

4. Write a menu driven application using switch-case statements in which case 1 : print a box, case 2 : an oval, case 3 : an arrow and case 4 : a diamond using loop and control statements.

1. *****	2. ***	3. *	4. *
* *	* *	***	* *
* *	* *	*****	* *
* *	* *	*	* *
* *	* *	*	* *
* *	* *	*	* *
*****	***	*	*

```
package lab_1;
import java.util.Scanner;
public class Pattern1 {
    public static void printBox(int n)
    {
        for(int i=1;i<=n;i++)
        {
            if(i==1 || i==n)
            {
                for(int j=1;j<=n;j++)
                    System.out.print("*");
            }
            else
            {
                System.out.print("*");
                for(int j=1;j<=n-2;j++)
                    System.out.print(" ");
                System.out.print("*");
            }
            System.out.println();
        }
    }
    public static void printOval(int n)
    {
        int t=(n-3)/4;
        for(int i=1;i<=n;i++)
        {
            if(i==1 || i==n)
            {
                for(int j=1;j<=t+1;j++)
                    System.out.print(" ");
                for(int j=1;j<=2*t+1;j++)
                    System.out.print("*");
            }
            else if(i>1 && i<=t+1)
            {
                for(int j=1;j<=t-i+2;j++)
                    System.out.print(" ");
                System.out.print("*");
                for(int j=1;j<=2*t+1+2*(i-2);j++)
                    System.out.print(" ");
                System.out.print("*");
            }
            else if(i>t+1 && i<(3*t+3))
            {

```

```

        System.out.print("*");
        for(int j=1;j<=4*t+1;j++)
            System.out.print(" ");
        System.out.print("*");
    }
    else
    {
        int k=n-i+1;
        for(int j=1;j<=t-k+2;j++)
            System.out.print(" ");
        System.out.print("*");
        for(int j=1;j<=2*t+1+2*(k-2);j++)
            System.out.print(" ");
        System.out.print("*");
    }
    System.out.println();
}

}

public static void printArrow(int n)
{
    for(int i=0;i<n/2+1;i++)
    {
        for(int j=0;j<n/2-i;j++)
        {
            System.out.print(" ");
        }
        for(int j=0;j<2*i+1;j++)
        {
            System.out.print("*");
        }
        System.out.print("\n");
    }
    for(int i=0;i<n/2+2;i++)
    {
        for(int j=0;j<n/2;j++)
        {
            System.out.print(" ");
        }
        System.out.print("*\n");
    }
}

static void printDiamond(int n)
{
    for(int i=0;i<n/2+1;i++)
    {
        for(int j=0;j<n/2-i;j++)

```

```
        {
            System.out.print(" ");
        }
        System.out.print("*");
        for(int j=0;j<2*i-1;j++)
        {
            System.out.print(" ");
        }
        if(i>0)
            System.out.print("*");

        System.out.print("\n");
    }
    for(int i=n/2-1;i>=0;i--)
    {
        for(int j=0;j<n/2-i;j++)
        {
            System.out.print(" ");
        }
        System.out.print("*");

        for(int j=0;j<2*i-1;j++)
        {
            System.out.print(" ");
        }

        if(i>0)

            System.out.print("*");
        System.out.print("\n");
    }
}

public static void main(String args[])
{
    System.out.println("1.print a box");
    System.out.println("2.print a oval");
    System.out.println("3.print a arrow");
    System.out.println("4.print a diamond");
    System.out.println("5.exit");
    int t=1;
    while(t==1)
    {
        System.out.print("enter your choice");
        Scanner in = new Scanner(System.in);
        int ch=in.nextInt();
        switch(ch)
```

```
{
case 1:
    System.out.print("enter no .of rows");
    int n=in.nextInt();
    printBox(n);
    break;
case 2:
    System.out.print("enter no .of rows in 4n+3");
    int m=in.nextInt();
    printOval(m);
    break;
case 3:
    System.out.print("enter no .of rows in 2n+1");
    int r=in.nextInt();
    printArrow(r);
    break;
case 4:
    System.out.print("enter no .of rows in 2n+1");
    int d=in.nextInt();
    printDiamond(d);
    break;
case 5:
    t=0;
    break;
}
}
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_1>java Pattern1
1.print a box
2.print a oval
3.print a arrow
4.print a diamond
5.exit
enter your choice1
enter no .of rows4
****
*  *
*  *
****
enter your choice2
enter no .of rows in 4n+37
***
*  *
*  *
*  *
*  *
*  *
***
enter your choice4
enter no .of rows in 2n+15
*
*  *
*  *
*  *
*
enter your choice5
```


Practical No. 2

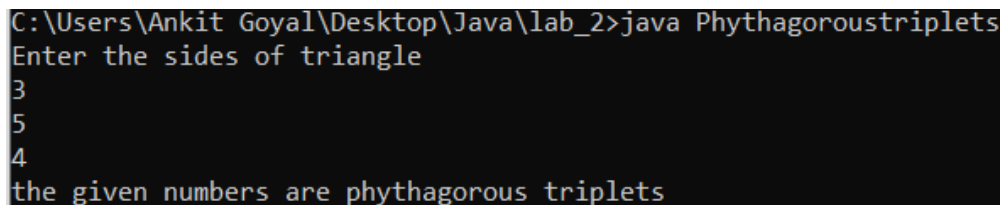
1. Write an application to that asks the user to enter three integer values($2 < n < 100$), and then displays that these values are Pythagorean triplets or not. The application should try all the combinations of the values.

```
package lab_2;
import java.util.*;
public class Phythagoroustriplets {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a , b , c;
        System.out.println("Enter the sides of triangle");
        a = in.nextInt();
        b = in.nextInt();
        c = in.nextInt();
        a = a*a;
        b = b*b;
        c = c*c;
        if(((a == b+c) || (b == a+c) ) || (c == a+b) )
            System.out.println("the given numbers are phythagorous triplets");
        else System.out.println("the given numbrs are not phythagorous triplets");

    }

}
```



```
C:\Users\Ankit Goyal\Desktop\Java\lab_2>java Phythagoroustriplets
Enter the sides of triangle
3
5
4
the given numbers are phythagorous triplets
```

2. Write an application that inputs from the user the radius of the circle as a floating point and prints the diameter, circumference and area. Use the following formulas :

Diameter = $2r$

Circumference = $2 * PI * r$

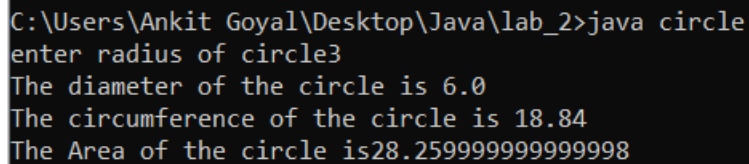
Area = $PI * r^2$

Take $PI = 3.14159$, should be declared as final

```
package lab_2;
import java.util.*;
public class circle {

    static final double pi = 3.14;
    public static void main(String[] args)
    {
        double r;
        System.out.print("enter radius of circle");
        Scanner in = new Scanner(System.in);
        r = in.nextFloat();
        in.close();
        System.out.println("The diameter of the circle is " + (2*r));
        System.out.println("The circum. of the circle is" + (2*pi*r));
        System.out.println("The Area of the circle is" + (pi * r* r ));

    }
}
```



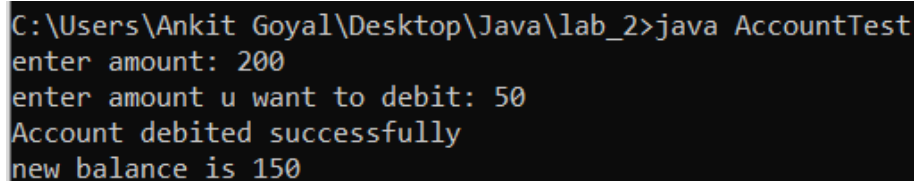
```
C:\Users\Ankit Goyal\Desktop\Java\lab_2>java circle
enter radius of circle3
The diameter of the circle is 6.0
The circumference of the circle is 18.84
The Area of the circle is28.259999999999998
```

3. Write an application which contains a class named “Account” that maintains the balance of a bank account. The Account class has method debit() that withdraws money from the account. Ensure that the debit amount does not exceed the account’s balance. If it does, the balance should remain unchanged with a message indicating “Debit amount exceeded account balance”.

```
package lab_2;
import java.util.Scanner;

class Account
{
    public int balance;
    Account(int a)
    {
        this.balance = a;
    }
}
```

```
public class AccountTest {  
  
    public static void main(String[] args) {  
        Scanner in = new Scanner(System.in);  
        int amt;  
        System.out.print("enter amount: ");  
        amt = in.nextInt();  
        Account obj = new Account(amt);  
        int debit ;  
        System.out.print("enter amount u want to debit: ");  
        debit = in.nextInt();  
        if(debit > obj.balance)  
        {  
            System.out.println("Debit balance exceeded from the current balance");  
            System.out.println("new balance is " + obj.balance);  
        }  
        else  
        {  
            obj.balance = obj.balance - debit;  
            System.out.println("Account debited successfully");  
            System.out.println("new balance is "+ obj.balance);  
        }  
        in.close();  
    }  
}
```



```
C:\Users\Ankit Goyal\Desktop\Java\lab_2>java AccountTest  
enter amount: 200  
enter amount u want to debit: 50  
Account debited successfully  
new balance is 150
```

4. Write an application to determine the gross pay for 3 employees. The company pays straight for first 40 hours and half after that for overtime. Salary per hour and number of hours of each employee should be input by user.

```
package lab_2;  
import java.util.Scanner;
```

```
class Employee  
{
```

```

        public int whour;
        public int salary;
        Employee()
        {
            this.whour =0;
            this.salary =0;
        }
    }
    public class Grosspay {

        public static void main(String[] args) {
            Employee[] arr = new Employee[3];
            Scanner in = new Scanner(System.in);
            System.out.println("Enter the salary per hour and working hours for the three
employees");
            for(int i=0;i<3;i++)
            {
                arr[i] = new Employee();
                arr[i].salary = in.nextInt();
                arr[i].whour = in.nextInt();
            }
            System.out.println("The amount paid to each employee at the end of the month is
");
            for(int i=0; i<3;i++)
            {
                if(arr[i].whour>40)
                {
                    System.out.println("the amount paid to "+ i + "th employee is " +(
(40*arr[i].salary) +( (arr[i].whour - 40 )*( arr[i].salary / 2))) );
                }
                else
                    System.out.println("the amount paid to "+ i + "th employee is " +
arr[i].whour *arr[i].salary);
            }
            in.close();
        }
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_2>java Grosspay
Enter the salary per hour and working hours for the three employees
100 50
200 60
300 40
The amount paid to each employee at the end of the month is
the amount paid to 0th employee is 4500
the amount paid to 1th employee is 10000
the amount paid to 2th employee is 12000

```

Practical No. 3

1. Write an application to store all the addition combination of the two dices in 2D array and display as follows :

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

```
package lab_3;
```

```
public class Dices {
```

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

```
        int[][] arr = new int[6][6];
```

```
        for(int i=0;i<6;i++)
```

```
        {
```

```
            for(int j=0;j<6;j++)
```

```
            {
```

```
                arr[i][j]=i+j+2;
```

```
            }
```

```
        }
```

```
        for(int i=0;i<6;i++)
```

```
        {
```

```
            for(int j=0;j<6;j++)
```

```
            {
```

```
                System.out.print(arr[i][j]+" ");
```

```
            }
```

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

```
        }
```

```
    }
```

```
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_3>java Dices
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10
6 7 8 9 10 11
7 8 9 10 11 12
```

2. Write a program to implement all the functions of the String class.

i. To convert the string to lowercase letters

ii. To convert the string to uppercase letters

iii. Replace the character at the given index

iv. Check if two strings are equal

v. To find the length of the string

vi. Find the character at the given position

vii. Concatenate two strings

viii. Check if the given string is a substring of another given string

```
package lab_3;
import java.util.*;
```

```
public class Stringimplement {
```

```
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("You can perform the following operation ");
        System.out.println("1. to convert the string to lower case");
        System.out.println("2. To convert the string to upper case");
        System.out.println("3. To replace a character in entire string");
        System.out.println("4. To check if two strings are equal");
        System.out.println("5. To check the length of string ");
        System.out.println("6. To find a character at given poistion");
        System.out.println("7. To concate two strings");
        System.out.println("8. To return  substring of a string from a specified position");
        while(true){
            int choice ;
            System.out.print("enter your choice: ");
            choice = in.nextInt();
            //System.out.println();
            switch(choice)
            {
```

```
case 1 :
{
    String a ;
    System.out.println("enter string: ");
    a = in.next();
    a = a.toLowerCase();
    System.out.println("in lower case: "+ a);
    break;
}
case 2 :
{
    String a;
    System.out.print("enter string: ");
    a = in.next();
    a = a.toUpperCase();
    System.out.println("in upper case: " + a);
    break;
}
case 3 :
{
    String a;
    System.out.print("enter string: ");
    a = in.next();
    char b,c;
    System.out.println("enter the character to replace with another
character");
    b = in.next().charAt(0);
    c = in.next().charAt(0);
    a = a.replace(b, c);
    System.out.println("the new string is: "+ a);
    break;
}
case 4 :
{
    String a , b;
    System.out.print("enter two string: ");
    a = in.next();
    b = in.next();
    if(a.equals(b))
    {
        System.out.println("these strings are equal");
    }
    else System.out.println("these strings are not equal");
    break;
}
case 5 :
```

```
        {
            String a;
            System.out.print("enter string: ");
            a = in.next();
            System.out.println("the length of the string is "+ a.length());
            break;
        }
    case 6 :
    {
        String a;
        System.out.print("enter string: ");
        a = in.next();
        int b;
        System.out.print("enter a position to find the character: ");
        b = in.nextInt();
        if(b<0||b>a.length() -1)
        {
            System.out.println("enter a valid position");
        }
        else System.out.println("the character at the "+ (b+1)+ "th position is "+
a.charAt(b));
        break;
    }
    case 7 :
    {
        System.out.print("enter two string: ");
        String a = in.next();
        String b = in.next();
        System.out.println("the concatenated string is "+ a.concat(b));
        break;
    }
    case 8 :
    {
        int b;
        System.out.print("enter string: ");
        String a = in.next();
        System.out.print("enter a positoin to find the substring: ");
        b = in.nextInt();
        if(b<0 || b> a.length()-1)
        {
            System.out.println("enter a valid positoin");
        }
        else
        {
            System.out.println("the required substring is "+ a.substring(b));
```



```

        }
        break;
    }
    default :
    {
        System.out.println("enter a valid choice");
        in.close();
        return ;
    }
}
}
}
}
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_3>java Stringimplement
You can perform the following operation
1. to convert the string to lower case
2. To convert the string to upper case
3. To replace a character in entire string
4. To check if two strings are equal
5. To check the length of string
6. To find a character at given poistion
7. To concats two strings
8. To return substring of a string from a specified position
enter your choice: 1
enter string:
ANKit
in lower case: ankit
enter your choice: 5
enter string: ankitgoyal
the length of the string is 10
enter your choice: 3
enter string: ankit
enter the character to replace with another character
k
a
the new string is: anait

```

3. A string is a double string if it has even length and half of the string is equal to the next half of the string. Write a program to check if the given string is a double string.

```

package lab_3;
import java.util.Scanner;

public class Doublestring {

    public static void main(String[] args) {
        String a;
        Scanner in = new Scanner(System.in);
    }
}

```

```

        System.out.println("enter a string");
        a = in.nextLine();
        if(a.length()%2 !=0)
        {
            System.out.println("this string is not a double string");
            in.close();
            return ;
        }
        else
        {
            for(int i=0;i<a.length()/2;i++)
            {
                if(a.charAt(i)!=a.charAt(a.length()-1-i))
                {
                    System.out.println("this strng is not a double string ");
                    in.close();
                    return ;
                }
            }
            System.out.println("this string is double string");
            in.close();
            return ;
        }
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_3>java Doublestring
enter a string
ankitgoyal
this strng is not a double string

```

4. Write a program in which you have an array of integers. Implement copy paste which allows you to copy any continuous subsequences of array and paste it into any position of the array.

```
import java.util.*;
```

```

class CopyPaste{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter size of array: ");
        int size=sc.nextInt();
        int[] arr= new int[size];
        int i,j,length;
        System.out.println("Enter elements of array: ");
    }
}

```

```

        for(i=0;i<size;i++){
            arr[i]=sc.nextInt();
        }

        int sCopy,eCopy,sPaste;
        int choice=1;
        while(choice==1){
            System.out.println("Enter starting and ending index for copying");
            sCopy=sc.nextInt();
            eCopy=sc.nextInt();
            if(sCopy<0 || eCopy>=size){
                System.out.println("Error: Indices out of bound ");
            } else {
                length=eCopy-sCopy+1;
                int[] copiedArray= new int[length];
                for(i=sCopy,j=0;i<=eCopy;i++,j++){
                    copiedArray[j]=arr[i];
                }
                System.out.println("Enter starting index for pasting");
                sPaste=sc.nextInt();
                sPaste++;
                if(sPaste+length<size){
                    for(i=sPaste-1,j=0;i<sPaste+length-1;i++,j++){
                        arr[i]=copiedArray[j];
                    }
                    for(i=0;i<size;i++){
                        System.out.print(arr[i]+" ");
                    }
                }
            }
            System.out.println("\nDo you want to continue: ");
            choice = sc.nextInt();
        }
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_3>java CopyPaste
Enter size of array:
8
Enter elements of array:
8 7 4 5 6 2 3 1
Enter starting and ending index for copying
3 5
Enter starting index for pasting
0
5 6 2 5 6 2 3 1
Do you want to continue:
0

```

Practical No. 4

1. Create a package package1 which contains three classes, one is super class and others are subclasses. Make some of the components private, some of them protected and some public, show access control using the above description. Then, create another package p2 in which the class can access public components of package package1. Now import the classes of package package1 and package2 for the class of the main() method.

```
package lab_4.p1;
import java.util.*;
class mainClass {
    public static void main(String[] args) {

        System.out.println("\nAccessing the problem1b(which extends the problem1)
variables and functions.");
        problem1b p1b = new problem1b();
        System.out.println("\nWow! I can access my public members here: " + p1b.stuff);
        System.out.println("\nAlso, I can use my parents function : ");
        p1b.CallMe();
    }
}

package lab_4.p1;
public class problem1 {

    public int variable1 = 1234; // accessible to other classes and other packages also
    protected String variable2 = " Java is Beautiful"; // only accessible to the base classes
of class Problem1

    private float variable3 = 2.345f; // not accessible outside this class
    static public int variable4 = 4321;
    void CallMe() {
        System.out.println("Calling a function of the problem1 class\n");
        System.out.println("We have called you. And you answered, so nice of you! I
think you are default or public(may be protected if I am your child:)");
    }
    // this function acts as public within this package and private outside.
}

package lab_4.p1;
class problem1a extends problem1 {
    private int variable = 1234;
}
package lab_4.p1;
class problem1b extends problem1 {
    public int stuff = 1000000;
    private float a = 0.00023495320f;
```

```

}

package lab_4.p2;
import lab_4.p1.*;
class anotherClass {

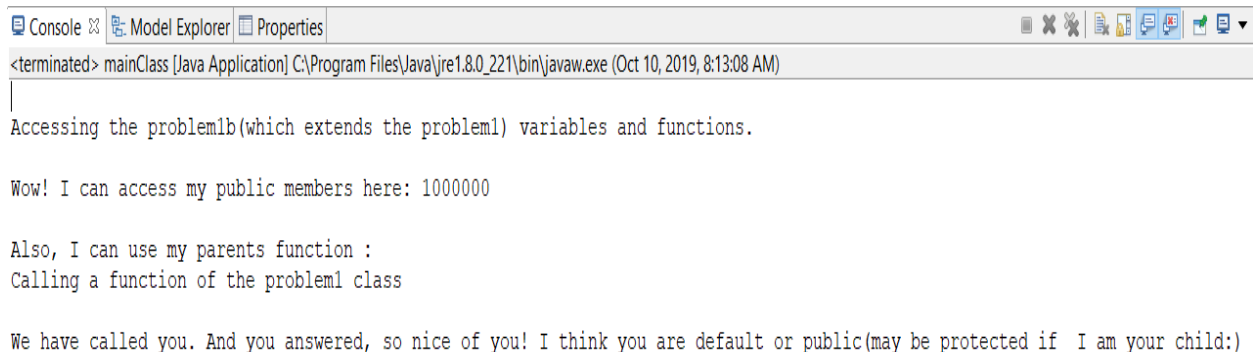
    public String variable = "This is my public Variable";
    void Public() {
        System.out.println("\nThe called function is my public.\n\n");
    }
    void CallingProblem1() {
        System.out.println("\nThis is the public member of the problem1 class of
package1 : " + lab_4.p1.problem1.variable4);
    }
}

package lab_4;
import lab_4.p1.*;
class checkingPackage {

    public static void main(String[] args) {

        problem1 p = new problem1();
        p.variable1 = 2;
        System.out.println(p.variable1);
    }
}

```



The screenshot shows a Java IDE with a console window. The console output is as follows:

```

<terminated> mainClass [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Oct 10, 2019, 8:13:08 AM)
Accessing the problem1b(which extends the problem1) variables and functions.

Wow! I can access my public members here: 1000000

Also, I can use my parents function :
Calling a function of the problem1 class

We have called you. And you answered, so nice of you! I think you are default or public(may be protected if I am your child:)

```

2. Create a class student which takes the roll number of the students, another class Test extends Student and add up marks of two tests. Now an interface Sports declares the sports marks and has method putMarks. Implement multiple inheritance in Class Result using above classes and interface.

```
import java.util.*;
class Student {

    int rollNumber;
    void SetRollNumber(int rollNumber) {
        this.rollNumber = rollNumber;
    }
    int GetRollNumber() {
        return this.rollNumber;
    }
}
class Test extends Student {
    int marks1, marks2;
    // int sportsMarks = 0;
    public void setMarks(int marks1, int marks2) {
        this.marks1 = marks1;
        this.marks2 = marks2;
    }
    int AddMarksUp() {
        return this.marks1 + this.marks2;
    }
}
interface Sports {
    public void setSportsMarks(int sM);
    public int putMarks();
}
class Result extends Test implements Sports {

    int sportsMarks;
    public void setSportsMarks(int sM) {
        sportsMarks = sM;
    }
    public int putMarks() {
        int totalMarks = sportsMarks + marks1 + marks2;
        return totalMarks;
    }
}
class MarksCalculation {

    public static void main(String[] args) {

        Result R = new Result();
        Test T = new Test();
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the marks in the two tests: ");
```

```

        int marks1 = input.nextInt();
        int marks2 = input.nextInt();
        T.setMarks(marks1, marks2);
        System.out.println("Enter the marks in the sports test: ");
        int sportsMarks = input.nextInt();
        R.setSportsMarks(sportsMarks);
        R.marks1 = T.marks1;
        R.marks2 = T.marks2;
        System.out.println("\nThe result is finally out : " + R.putMarks());
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_4>java MarksCalculation
Enter the marks in the two tests:
6 8
Enter the marks in the sports test:
5

The result is finally out : 19

```

3. Write an application which takes a 1D array of the numbers $0 \leq n \leq 9$ only as input and displays the number of times n repeated most consecutively. For example :

Input : 1 2 1 1 1 5 5 2 2 3 3 5 5 5 5 4 6

Output : 4

```

package lab_4;
import java.util.Scanner;
public class MaxFreq {

    public static void main(String[] args) {

        System.out.println("enter no. of elements");
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int []v = new int[n];
        for(int i=0;i<n;i++)
        {
            v[i]=in.nextInt();
        }
        int maxcount=0;
        int i=0;
        while(i<n-1)
        {
            int count=1;
            while(i<n-1 && v[i]==v[i+1])

```

```

        {
            count++;
            i++;
        }
        if(maxcount<count)
            maxcount=count;
        i++;
    }
    System.out.println("output is "+ maxcount);
}
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_3>java MaxFreq
enter no. of elements
6
2 5 4 1 1 8
output is 2

```

4. Write an application to store all the addition combination of the two dices in 2D array and display as follows :

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

```
package lab_4;
```

```
public class Dices {
```

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

```

        int[][] arr = new int[6][6];
        for(int i=0;i<6;i++)
        {
            for(int j=0;j<6;j++)
            {
                arr[i][j]=i+j+2;
            }
        }
    }
}

```



```
    }  
    for(int i=0;i<6;i++)  
    {  
        for(int j=0;j<6;j++)  
        {  
            System.out.print(arr[i][j]+" ");  
        }  
        System.out.println();  
    }  
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_3>java Dices  
2 3 4 5 6 7  
3 4 5 6 7 8  
4 5 6 7 8 9  
5 6 7 8 9 10  
6 7 8 9 10 11  
7 8 9 10 11 12
```

Practical No. 5

1. Write an application to show access protection in user defined packages. Create a package p1 which has three classes: Protection, Derived and SamePackage. First class contains variables of all the access types. Derived is subclass of Protection but SamePackage is not. Other package p2 consisting of two class: one deriving p1.Protection and other not. Import the package considering all the cases of access protection.

```
package lab_5.p1;
```

```
public class Protection {
```

```
    public int publicVariable = 2323;
    private float floatVariable = 23.223f;
    protected String stringVariable = "Protected";
}
```

```
package lab_5.p1;
```

```
public class Derived extends Protection {
```

```
    public int stuff = 23421;
    public void print() {
        System.out.println("I am a function of Derived class");
    }
    public void printProtected() {
        System.out.println("I can access the protected variable of Protection class. Its value is: " +
stringVariable);
    }
}
```

```
package lab_5.p1;
```

```
public class SamePackage {
    int variable = 23;
    protected String is = "Hello";
}
```

```
package lab_5.p2;
```

```
import lab_5.p1.*;
```

```
public class first extends Protection {
```

```
    int anyNumber = 23422;
    public String anyName = "Henry Ford";
}
```

```
package lab_5.p2;
```

```
public class second {

    double doubleStuff = 2.3324;
    Integer integer = new Integer(23);
    public void print() {
        System.out.println("I am a function of second class.");
        System.out.println("I have only two variable : " + doubleStuff + "and " + integer);
    }
}

package lab_5;
import lab_5.p1.*;
import lab_5.p2.*;
class Problem1 {

    public static void main(String[] args) {
        Protection p = new Protection();
        System.out.println("Public Variable's value : "+ p.publicVariable);
        System.out.println("\n\nThe variables and functions of Protection class are: ");
        // System.out.println("Protected Variable: " + stringVariable);
        System.out.println("Can't access Private and Protected variables\n\n");

        Derived d = new Derived();
        System.out.println("The variables and functions of Derived class are: ");
        System.out.println("Public variable's value : " + d.stuff);
        d.print();
        d.printProtected();
        System.out.println("\n\n");

        SamePackage s = new SamePackage();
        System.out.println("The variables and functions of SamePackage class are: ");
        System.out.println("The default variable of this class is not visible.");
        System.out.println("The protected variable of this class is not visible.\n\n");

        first f = new first();
        System.out.println("The variables and functions of first class are: ");
        System.out.println("The default variable of the class is not accessible.");
        System.out.println("Public variable's value is : " + f.anyName + "\n\n");

        second sec = new second();
        System.out.println("The default variables of the class are not accessible.");
        System.out.println("But there is a public function. Lets call that : ");
        sec.print();
        System.out.println("\n\n");
    }
}
```

```
}
```

```
<terminated> Problem1 [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (Oct 10, 2019, 7:21:32 AM)
```

```
Public Variable's value : 2323
```

```
The variables and functions of Protection class are:  
Can't access Private and Protected variables
```

```
The variables and functions of Derived class are:  
Public variable's value : 23421  
I am a function of Derived class  
I can access the protected variable of Protection class. Its value is: Protected
```

```
The variables and functions of SamePackage class are:  
The default variable of this class is not visible.  
The protected variable of this class is not visible.
```

```
The variables and functions of first class are:  
The default variable of the class is not accessible.  
Public variable's value is : Henry Ford
```

```
The default variables of the class are not accessible.  
But there is a public function. Lets call that :  
I am a function of second class.  
I have only two variable : 2.3324and 23
```

2. Multiple Inheritance in java is when a class implements more than one interface. Write an application showing Multiple Inheritance and Dynamic Polymorphism (Method Overriding) using Interface.

```
class Game {  
    public void type() {  
        System.out.println("We are talking about Indoor and Outdoor games.");  
    }  
    public void numberOfPlayers() {  
        System.out.println("Number of player is a relative term, depends on the game.");  
    }  
}  
class Football extends Game {
```

```
    public void type() {
        System.out.println("Football is an outdoor game.");
    }
    public void numberOfPlayers() {
        System.out.println("In Football, we have 11 players in each team.");
    }
}
class Kabaddi extends Game {
    public void type() {
        System.out.println("Kabaddi is an outdoor game.");
    }
    public void numberOfPlayers() {
        System.out.println("In Kabaddi, we have 12 players in each team.");
    }
}
import java.util.*;
class Problem2 {
    public static void main(String[] args) {
        System.out.println("\n\n");
        Game g = new Game();
        System.out.println("Game class functions : ");
        g.type();
        g.numberOfPlayers();
        System.out.println("\n\n");
        System.out.println("Football class functions : ");
        g = new Football();
        g.type();
        g.numberOfPlayers();
        System.out.println("\n\n");
        System.out.println("Kabaddi class functions : ");
        g = new Kabaddi();
        g.type();
        g.numberOfPlayers();
        System.out.println("\n\n");
    }
}
```

```
<terminated> Problem2 [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe
```

```
Game class functions :  
We are talking about Indoor and Outdoor games.  
Number of player is a relative term, depends on the game.
```

```
Football class functions :  
Football is an outdoor game.  
In Football, we have 11 players in each team.
```

```
Kabaddi class functions :  
Kabaddi is an outdoor game.  
In Kabaddi, we have 12 players in each team.
```

3. Define a package named area which contains classes : Square, Rectangle, Circle, Ellipse and Triangle. Write an application in which the main class, FindArea outside the package area imports the package area and calculate the area of different shapes. Output of your application should be menu driven.

```
package lab_5.area;
```

```
public class Circle {  
    double radius;  
    public void putrad(double radius){  
        this.radius=radius;  
    }  
    public double get_area()  
    {  
        return 3.14*radius*radius;  
    }  
}
```

```
package lab_5.area;
```

```
public class Rectangle {  
    double length, breadth;  
    public void putsides(double length, double breadth){  
        this.length=length;  
        this.breadth=breadth;  
    }  
    public double get_area()  
    {
```

```
        return length*breadth;
    }
}

package lab_5.area;
public class Elipse {
    double major, minor;
    public void putaxes(double major, double minor){
        this.major=major;
        this.minor=minor;
    }
    public double get_area()
    {
        return 3.14*major*minor;
    }
}

package lab_5.area;
public class Square {
    double side;
    public void putside(double side){
        this.side=side;
    }
    public double get_area()
    {
        return side*side;
    }
}

package lab_5.area;
public class Triangle {
    double base, height;
    public void putdims(double base, double height){
        this.base=base;
        this.height=height;
    }
    public double get_area()
    {
        return 0.5*base*height;
    }
}

package lab_5;
import java.util.Scanner;
```

```
import lab_5.area.*;
public class FindArea {
    public static void main(String []args){
        System.out.println("You can find the area for ");
        System.out.println("1. square");
        System.out.println("2. rectangle");
        System.out.println("3. circle");
        System.out.println("4. ellipse");
        System.out.println("5. triangle");
        while(true){
            System.out.println("enter choice");
            int choice ;
            Scanner in = new Scanner(System.in);
            choice = in.nextInt();
            switch(choice)
            {
                case 1 :
                {
                    Square s = new Square();
                    System.out.println("enter side");
                    double side = in.nextDouble();
                    s.putside(side);
                    System.out.println("area of square = " + s.get_area());
                    break;
                }
                case 2 :
                {
                    Rectangle r = new Rectangle();
                    System.out.println("enter length and breadth");
                    double length = in.nextDouble();
                    double breadth = in.nextDouble();
                    r.putside(length,breadth);
                    System.out.println("area of rectangle = " + r.get_area());
                    break;
                }
                case 3 :
                {
                    Circle c = new Circle();
                    System.out.println("enter radius");
                    double rad = in.nextDouble();
                    c.putrad(rad);
                    System.out.println("area of circle = " + c.get_area());
                }
            }
        }
    }
}
```



```
        break;
    }
    case 4 :
    {
        Elipse e = new Elipse();
        System.out.println("enter major and minor axes");
        double major = in.nextDouble();
        double minor = in.nextDouble();
        e.putaxes(major,minor);
        System.out.println("area of rectangle = " + e.get_area());
        break;
    }
    case 5 :
    {
        Triangle t = new Triangle();
        System.out.println("enter base and height");
        double base = in.nextDouble();
        double height = in.nextDouble();
        t.putdims(base, height);
        System.out.println("area of rectangle = " + t.get_area());
        break;
    }

    default :
    {
        System.out.println("enter a valid choice");
        in.close();
        return ;
    }
}
}
```

```
FindArea [Java Application] C:\Program Files\Ja
You can find the area for
1. square
2. rectangle
3. circle
4. ellipse
5. triangle
enter choice
1
enter side
5
area of square = 25.0
enter choice
2
enter length and breadth
8 5
area of rectangle = 40.0
enter choice
3
enter radius
5
area of circle = 78.5
enter choice
4
enter major and minor axes
2 3
area of rectangle = 18.84
enter choice
5
enter base and height
2 4
area of rectangle = 4.0
```

3. Define an interface Volume having a final variable P1 and a method to calculate volumes of different shapes. Implement the interface Volume in class Cube, Cylinder, Cone and Sphere to calculate the volume of these different shapes. Calculate the volume using the main class FindVolume. Output of your application should be menu driven.

```
public interface Volume {

    final double pi = 3.14;
    public double volume();
}

package lab_5.Volume;
public class Sphere implements Volume{
    public int radius;
    public Sphere(int radius){
        this.radius = radius;
    }
}
```

```
public double volume(){
    double v = (4 * pi * radius * radius * radius ) / 3;
    return v;
}
}
```

```
package lab_5.Volume;
public class Cone implements Volume{
    public int radius,height;
```

```
    public Cone(int radius,int height){
        this.radius=radius;
        this.height=height;
    }
    public double volume(){
        return (pi*radius*radius*height)/3;
    }
}
```

```
package lab_5.Volume;
public class Cube implements Volume{
    public int side;
    public Cube(int side){
        this.side=side;
    }
    public double volume(){
        return side*side*side;
    }
}
```

```
package lab_5.Volume;
public class Cylinder implements Volume {
    public int radius,h;
```

```
    public Cylinder(int radius, int h){
        this.radius=radius;
        this.h=h;
    }
    public double volume(){
        return pi*radius*h;
    }
}
import java.util.Scanner;
```

```
import volume.Sphere;
import volume.Cube;
import volume.Cylinder;
import volume.Cone;
public class Problem4 {
    public static void main(String[] args) {

        int option;
        Scanner input = new Scanner(System.in);
        do {
            System.out.println("Choose the solid you want the volume for: ");
            System.out.println("1. Sphere");
            System.out.println("2. Cylinder");
            System.out.println("3. Cube");
            System.out.println("4. Cone");
            System.out.println("0. Exit");
            option = input.nextInt();
            switch(option) {
                case 1:
                    System.out.println("Enter radius : ");
                    int r = input.nextInt();
                    Sphere s = new Sphere(r);
                    System.out.println("Volume of the sphere is : " + s.volume() + "\n");
                    break;

                case 2:
                    System.out.println("Enter radius and height of the cylinder : ");
                    int radius = input.nextInt();
                    int height = input.nextInt();
                    Cylinder c = new Cylinder(radius, height);
                    System.out.println("Volume of the cylinder : " + c.volume() + "\n");
                    break;

                case 3:
                    System.out.println("Enter the side of the cube : ");
                    int side = input.nextInt();
                    Cube cB = new Cube(side);
                    System.out.println("Volume of cone is : " + cB.volume() + "\n");
                    break;

                case 4:
                    System.out.println("Enter the side of the Cone : ");
                    radius = input.nextInt();
```

```
        height = input.nextInt();
        Cone cN = new Cone(radius, height);
        System.out.println("Volume of cone is : " + cN.volume() + "\n");
        break;
    }
}while(option != 0);
}
```

```
Choose the solid you want the volume for:
```

- 1. Sphere
- 2. Cylinder
- 3. Cube
- 4. Cone
- 0. Exit

```
3
```

```
Enter the side of the cube :
```

```
10
```

```
Volume of cone is : 1000.0
```

```
Choose the solid you want the volume for:
```

- 1. Sphere
- 2. Cylinder
- 3. Cube
- 4. Cone
- 0. Exit

Practical No. 6

1. Use inheritance to create an exception superclass (called ExceptionA) and exception subclasses ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a program to demonstrate that the catch block for type ExceptionA catches exceptions of type ExceptionB and ExceptionC.

```
package Lab6;
import java.util.Scanner;
class ExceptionA extends Exception
{
    public String toString()
    {
        return "Exception A";
    }
}
class ExceptionB extends ExceptionA
{
    public String toString()
    {
        return "Exception B";
    }
}
class ExceptionC extends ExceptionB
{
    public String toString()
    {
        return "Exception C";
    }
}
public class ExceptionInheritance {

    static int val;
    public static void main(String args[])
    {
        System.out.println("Enter 1 to throw ExceptionA, 2 for ExcpetionB, 3 for
ExceptionC");
        Scanner in = new Scanner(System.in);
        val = in.nextInt();
        try
        {
            switch(val)
```

```

        {
            case 1:
                throw new ExceptionA();
            case 2:
                throw new ExceptionB();
            case 3:
                throw new ExceptionC();
        }
    }
    catch(ExceptionA e)
    {
        System.out.println("Caught Exception is "+ e+" and is caught in
ExceptionA catch.");
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_6>java ExceptionInheritance
Enter 1 to throw ExceptionA, 2 for ExcpetionB, 3 for ExceptionC
1
Caught Exception is Exception A

```

2. Write a program that demonstrates how various exceptions are caught with catch(Exception exception).

This time, define classes ExceptionA (which inherits from class Exception) and ExceptionB (which inherits from class ExceptionA). In your program, create try blocks that throw exceptions of types ExceptionA, ExceptionB, NullPointerException and IOException. All exceptions should be caught with catch blocks specifying type Exception.

```

package Lab6;
import java.io.IOException;
import java.util.Scanner;

class ExceptionA extends Exception
{
    public String toString()
    {
        return "Exception A";
    }
}
class ExceptionB extends ExceptionA
{

```

```
        public String toString()
        {
            return "Exception B";
        }
    }

    public class ExceptionClass {

        static int val;
        public static void main(String args[])
        {
            System.out.println("Enter 1 to throw ExceptionA, 2 for ExcpetionB, 3 for
NullPointerException, 4 for IOException");
            Scanner in = new Scanner(System.in);
            val = in.nextInt();
            try
            {
                switch(val)
                {
                    case 1:
                        throw new ExceptionA();
                    case 2:
                        throw new ExceptionB();
                    case 3:
                        throw new NullPointerException();
                    case 4:
                        throw new IOException();
                }
            }
            catch(Exception e)
            {
                System.out.println("Caught Exception is "+ e);
            }
        }
    }
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_6>java ExceptionClass
Enter 1 to throw ExceptionA, 2 for ExcpetionB, 3 for NullPointerException, 4 for IOException
3
Caught Exception is java.lang.NullPointerException
```

2. Write a program that shows that the order of catch blocks is important. If you try a catch a superclass exception type before a subclass type, the compiler should generate errors.


```

package Lab6;

import java.io.IOException;
import java.util.Scanner;
public class CatchOrder {

    static int val;
    public static void main(String args[])
    {
        System.out.println("Enter 1 to throw ExceptionA, 2 for ExcpetionB, 3 for
        NullPointerException, 4 for IOException");
        Scanner in = new Scanner(System.in);
        val = in.nextInt();
        try
        {
            switch(val)
            {
                case 1:
                    throw new ExceptionA();
                case 2:
                    throw new ExceptionB();
                case 3:
                    throw new NullPointerException();
                case 4:
                    throw new IOException();
            }
        }
        catch(Exception e)
        {
            System.out.println("Caught Exception is "+ e);
        }
        catch(ExceptionA e)
        {
            System.out.println("Caught Subclass Exception is "+ e);
        }
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_6>javac CatchOrder.java
CatchOrder.java:39: error: exception ExceptionA has already been caught
        catch(ExceptionA e)
            ^
1 error

```

3. Write a program that illustrates rethrowing an exception. Define methods someMethod and someMethod2. Method someMethod2 should initially throw an exception. Method someMethod should call someMethod2, catch the exception and rethrow it. Call someMethod from method main(), and catch the rethrow exception. Print the stack trace of this exception.

```
package Lab6;
import java.io.IOException;
public class ExceptionInMethod {
    public static void methodB()
    {
        System.out.println("Hello You are in method B");
        try
        {
            throw new IOException();
        }
        catch(Exception e)
        {
            e.printStackTrace();
            throw new NullPointerException();
        }
    }
    public static void methodA()
    {
        System.out.println("Hello You are in method A");
        try
        {
            methodB();
        }
        catch(Exception e)
        {
            e.printStackTrace();
            throw new ArithmeticException();
        }
    }
    public static void main(String args[])
    {
        System.out.println("you are in Main");
        try
        {
            methodA();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

    }
}
C:\Users\Ankit Goyal\Desktop\Java\lab_6>java ExceptionInMethod
you are in Main
Hello You are in method A
Hello You are in method B
java.io.IOException
    at ExceptionInMethod.methodB(ExceptionInMethod.java:12)
    at ExceptionInMethod.methodA(ExceptionInMethod.java:26)
    at ExceptionInMethod.main(ExceptionInMethod.java:41)
java.lang.NullPointerException
    at ExceptionInMethod.methodB(ExceptionInMethod.java:17)
    at ExceptionInMethod.methodA(ExceptionInMethod.java:26)
    at ExceptionInMethod.main(ExceptionInMethod.java:41)
java.lang.ArithmeticException
    at ExceptionInMethod.methodA(ExceptionInMethod.java:31)
    at ExceptionInMethod.main(ExceptionInMethod.java:41)

```

4. Write an application to define two different Exceptions. Enter an issue_date and a return_date and define two different exceptions. First, if difference of issue-date and return_date is more than 15 days, throw an exception displaying calculated fine(Rs. 50 per day) after 15 days. Second, if return date is before issue_date, throw an exception displaying “Return date < Issue date”.

```

package Lab6;
import java.util.Date;
import java.util.Scanner;

class FineException extends Exception
{
    int fine ;
    public FineException(int fine)
    {
        this.fine = fine;
    }
    public String toString()
    {
        return "Fine for 15 days :" + fine;
    }
}
class InvalidDateException extends Exception
{
    public String toString()
    {

```

```

        return "Return Date < Issue Date" ;
    }
}
public class DateException {

    static Date issue_date,return_date;
    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        System.out.println("\nEnter the Year/Month/Date for the issuing date: ");
        int year, month, date;
        year = input.nextInt();
        month = input.nextInt();
        date = input.nextInt();
        issue_date = new Date(year,month,date);
        System.out.println("\nEnter the Year/Month/Date for the returning date: ");
        year = input.nextInt();
        month = input.nextInt();
        date = input.nextInt();
        return_date = new Date(year,month,date);
        try
        {
            if(return_date.before(issue_date))
            {
                throw new InvalidDateException();
            }
            if(return_date.getTime()-issue_date.getTime())>15*24*60*60*1000)
            {
                int fine = (int)(return_date.getTime()-
issue_date.getTime())/(24*60*60*1000)*50;
                throw new FineException(fine);
            }
        }
        catch(FineException e)
        {
            System.out.print(e);
        }
    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_6>java DateException
Fine for 15 days :850
C:\Users\Ankit Goyal\Desktop\Java\lab_6>

```

Practical No. 7

1. Write an application of multithreading. Create three different classes(threads) that inherit Thread class. Each class consists of a for loop that prints identity of the class with a number series in increasing order. Start all three threads together. Now run the application 2 or 3 times and show the output.

```
package lab_7;
class newThread1 extends Thread {
    newThread1()
    {
        start();
    }
    public void run() {
        try {
            for(int i = 1; i < 10; i++) {
                System.out.println("Child Thread 1: " + i);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child Thread 1 interrupted.");
        }
        System.out.println("Exiting child thread 1.");
    }
}
class newThread2 extends Thread {
    newThread2()
    {
        start();
    }
    public void run() {
        try {
            for(int i = 1; i < 10; i++) {
                System.out.println("Child Thread 2: " + i);
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Child Thread 2 interrupted.");
        }
        System.out.println("Exiting child thread 2.");
    }
}
class newThread3 extends Thread {
    newThread3()
    {
        start();
    }
}
```

```
        public void run() {
            try {
                for(int i = 1; i < 10; i++) {
                    System.out.println("Child Thread 3: " + i);
                    Thread.sleep(500);
                }
            } catch (InterruptedException e) {
                System.out.println("Child Thread 3 interrupted.");
            }
            System.out.println("Exiting child thread 3.");
        }
    }
    public class MultiThread {
        public static void main(String[] args) {
            new newThread1();
            new newThread2();
            new newThread3();
        }
    }
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_7>java MultiThread
Child Thread 1: 1
Child Thread 2: 1
Child Thread 3: 1
Child Thread 1: 2
Child Thread 2: 2
Child Thread 3: 2
Child Thread 1: 3
Child Thread 2: 3
Child Thread 3: 3
Child Thread 1: 4
Child Thread 2: 4
Child Thread 3: 4
Child Thread 1: 5
Child Thread 2: 5
Child Thread 3: 5
Child Thread 1: 6
Child Thread 2: 6
Child Thread 3: 6
Child Thread 1: 7
Child Thread 2: 7
Child Thread 3: 7
Child Thread 1: 8
Child Thread 2: 8
Child Thread 3: 8
Child Thread 1: 9
Child Thread 2: 9
Child Thread 3: 9
Exiting child thread 1.
Exiting child thread 2.
Exiting child thread 3.
```

2. Write an application of multithreading to create three different threads namely: One, Two and Three. Create the threads implementing Runnable Interface. Start all three threads together and print the thread with increasing number till the thread exit. Make sure that main thread exits at last.

NOTE : Use Thread.sleep().

```
package lab_7;

class newThread implements Runnable {
    String name;
    Thread t;

    newThread(String nameOfThread)
    {
        name = nameOfThread;
        t = new Thread(this, name);
        System.out.println("New Thread: " + t);
        t.start();
    }
    public void run()
    {
        try {
            for(int i = 1; i < 10; i++) {
                System.out.println(name + ": " + i);
                Thread.sleep(500);
            }
            catch (InterruptedException e) {
                System.out.println(name + "Interrupted");
            }
            System.out.println(name + " exiting.");
        }
    }
}

public class MultiTreadUsingRunnable {

    public static void main(String[] args) {
        new newThread("one");
        new newThread("two");
        new newThread("three");
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            System.out.println("main interrupted");
        }
        System.out.println("main exiting");
    }
}
```

```
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_7>java MultiTreadUsingRunnable
New Thread: Thread[one,5,main]
New Thread: Thread[two,5,main]
one: 1
two: 1
New Thread: Thread[three,5,main]
three: 1
one: 2
two: 2
three: 2
one: 3
two: 3
three: 3
one: 4
two: 4
three: 4
one: 5
two: 5
three: 5
one: 6
two: 6
three: 6
one: 7
two: 7
three: 7
one: 8
two: 8
three: 8
one: 9
two: 9
three: 9
one exiting.
two exiting.
three exiting.
main exiting
```

3. Write an application of multithreading with priority. Create three different threads of min, max and norm priority each. Now run each thread together for 10 seconds. Each thread should consist of a variable count that increases itself with time(click++). Stop all threads after 10 seconds and print the value of click for each and show the use priority in multithreading.

```
package lab_7;
import java.util.*;
```

```
class NewThread4 implements Runnable {
```



```
String name;
Thread t;
long click=0;
private volatile boolean running=true;
NewThread4(String threadname,int p){

    name=threadname;
    t=new Thread(this,name);
    System.out.println("New thread: "+t);
    t.setPriority(p);
    t.start();
}

public void run()
{
    while(running)
        click++;
}
public void stop()
{
    running=false;
}
public void frequency()
{
    System.out.println("no. of clicks "+name+" "+click);
}
}

public class MultiThreadWithPriority{

    public static void main(String args[]){
        NewThread4 one= new NewThread4("One",Thread.MIN_PRIORITY);
        NewThread4 two= new NewThread4("Two",Thread.NORM_PRIORITY);
        NewThread4 three= new NewThread4("Three",Thread.MAX_PRIORITY);
        try{
            Thread.sleep(100);
        }
        catch(InterruptedException e)
        {
            System.out.println("InterruptedException caught");
        }
        one.stop();
        two.stop();
        three.stop();
        one.frequency();
        two.frequency();
    }
}
```

```

    three.frequency();
    System.out.println("main thread exiting");    }
}

```

```

C:\Users\Ankit Goyal\Desktop\Java\lab_7>java MultiThreadWithPriority
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
no. of clicks One 71953474
no. of clicks Two 71823076
no. of clicks Three 71397994
main thread exiting

```

4. Inherit a class from Thread and override the run() method. Inside run(), print a message, and then call sleep(). Repeat this three times, then return from run(). Put a start-up message in the constructor and override finalize() to print a shut down message. Make a separate thread class that calls System.gc() and System.runFinalization() inside run(), printing a message as it does so. Make several thread objects of both types and run them to see what happens.

```

package lab_7;
class Sync extends Thread
{
    Sync()
    {
        System.out.println("Object created\n");
    }
    public synchronized void run(String s)
    {
        for(int i = 0 ; i< 3 ; i++)
        {
            System.out.println("Sending " + s);
            try {
                Thread.sleep(600);
            }
            catch (InterruptedException e) {
                System.out.println("Thread interrupted");
            }
            System.out.println("\n\n" + s + " sent");
        }
    }
    public void finalise()
    {
        System.out.println("Object is to be destroyed");
    }
}

```

```
}  
class Test extends Thread  
{  
    private String s;  
    Sync sy ;  
    Test(String a , Sync b)  
    {  
        s = a;  
        sy = b;  
    }  
    public void run()  
    {  
        System.out.println("Before running the finalise method");  
        sy.run(s);  
        sy.finalise();  
        System.out.println("calling the Garbage collector");  
        System.gc();  
    }  
}  
public class SynchronisedThread  
{  
    public static void main(String[] args)  
    {  
        Sync a ;  
        a = new Sync();  
        Test b = new Test("HI" , a);  
        b.run();  
    }  
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_7>java SynchronisedThread  
Object created  
  
Before running the finalise method  
Sending HI  
  
HI sent  
Sending HI  
  
HI sent  
Sending HI  
  
HI sent  
Object is to be destroyed  
calling the Garbage collector
```

5. There are two processes, a producer and a consumer, that share a common buffer with a limited size. The producer “produces” data and stores it in the buffer, and the consumer “consumes” the data, removing it from the buffer. Having two processes that run in parallel, we need to make sure that the producer won’t put new data in the buffer when the buffer is full and the consumer won’t try to remove data from the buffer if the buffer is empty.

```
package lab_7;
```

```
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;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0, j = 0;
        while(j < 5) {
```

```
        q.put(i++);
        j++;
    }
}
}
class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i=0;
        while(i<5) {
            q.get();
            i++;
        }
    }
}
}
public class ProducerConsumer {

    public static void main(String[] args) {

        Q q = new Q();
        new Producer(q);
        new Consumer(q);
    }
}
```

```
C:\Users\Ankit Goyal\Desktop\Java\lab_7>java ProducerConsumer
Put: 0
Got: 0
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
```

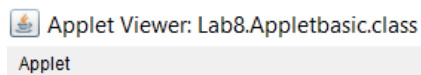
Practical No. 8

1. Write an applet program to draw a string, “This is my first Applet” on given coordinates and run the applet in both the browser and applet viewer.

```
package Lab8;
```

```
import java.applet.Applet;  
import java.awt.Graphics;
```

```
public class Appletbasic extends Applet  
{  
    public void paint(Graphics g)  
    {  
        g.drawString("This is my First Applet", 100, 200);  
    }  
}
```



This is my First Applet

2. Write an applet program that asks the user to enter two floating point numbers and draws their sum, difference, product and division.

Note : Use PARAM tag for user input.

```
package lab_8;  
import java.util.Scanner;  
import java.applet.Applet;  
import java.awt.Graphics;  
import java.awt.*;  
import java.awt.event.*;  
class AdapterExample extends WindowAdapter{  
    Frame f;  
    AdapterExample(){  
        f=new Frame();
```

```

        f.addWindowListener(this);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void windowClosing(WindowEvent e) {
        f.dispose();
    }
}
public class CalculatorUsingApplet extends Applet
{

    public void paint(Graphics g)
    {
        Applet ap = new Applet();
        Scanner in = new Scanner(System.in);
        int a , b;
        System.out.println("enter two integers");
        a = in.nextInt();
        b = in.nextInt();
        g.drawString("Addition " + (a+b) , 0 , 10);
        g.drawString("Substraction " + (a-b) , 0 , 20);
        g.drawString("Multiplicatoin " + (a*b) ,0 , 30);
        g.drawString("division " + (a /b) , 0 , 40);
        //ap.destroy();
        //new AdapterExample();
    }
}

```

CalculatorUsingApplet [Java Applet]

enter two integers

5 8

|



Applet

Addition 13
Substraction -3
Multiplicatoin 40
division 0

Applet started.

3. Write an applet that draws a checkerboard pattern as follows:

```
package Lab8;
```

```
import java.applet.Applet;
```

```
import java.awt.Graphics;
```

```
public class Pattern extends Applet
```

```
{
```

```
    public void paint(Graphics g)
```

```
    {
```

```
        String s = "****";
```

```
        g.drawString(s,0 , 10);
```

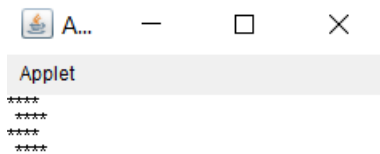
```
        g.drawString(s, 5, 20);
```

```
        g.drawString(s, 0, 30);
```

```
        g.drawString(s, 5, 40);
```

```
    }
```

```
}
```



Applet started.

4. Write an applet in which background is Red and foreground (text color) is Blue. Also show the message “Background is Red and Foreground is Blue” in the status window.

```
package Lab8;
```

```
import java.applet.Applet;
```

```
import java.awt.Color;
```

```
import java.awt.Graphics;
```



```

public class ColorChangeinApplet extends Applet {
    public void init() {
        // Here we change the default grey color background of an
        // applet to yellow background.
        setBackground(Color.RED);
    }

    public void paint(Graphics g) {
        g.setColor(Color.BLUE);
        g.drawString("Background is RED and Foreground is BLUE", 0, 50);
    }
}

```



5. Write an applet program showing URL of code base and document vase in the applet window. NOTE : Use `getCodeBase()` and `getDocumentBase()`.

```

package lab_8;
import java.awt.*;
import java.applet.*;
import java.net.*;

/*
<applet code="Bases" width=300 height=50>
</applet>
*/
public class URlshowing extends Applet {

    public void paint(Graphics g) {

```

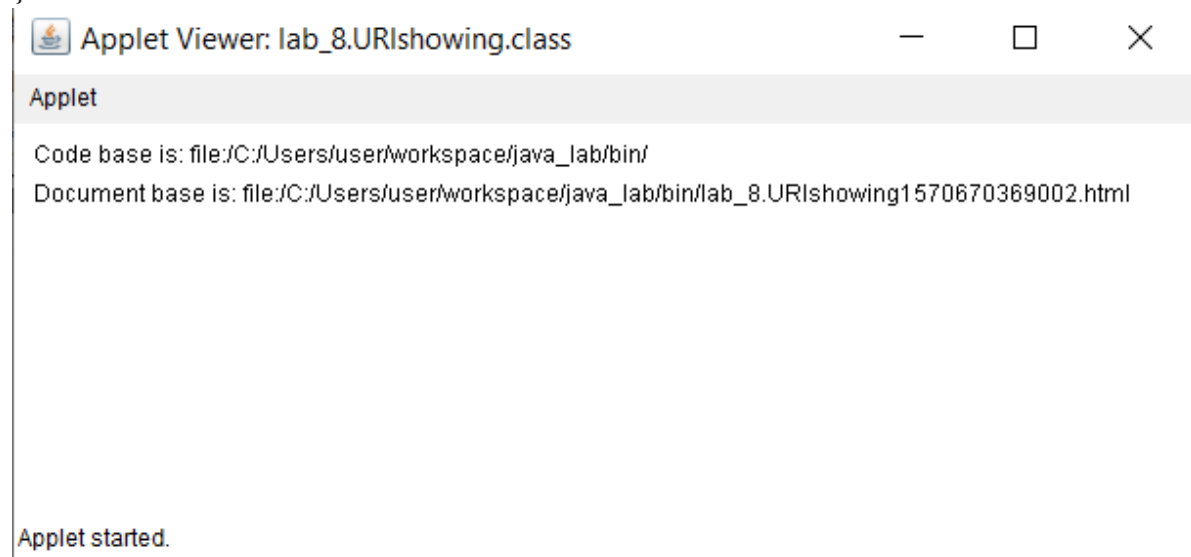
```
String msg;
```

```
URL url = getCodeBase();  
msg = "Code base is: " + url.toString();  
g.drawString(msg, 10, 20);
```

```
url = getDocumentBase();  
msg = "Document base is: " + url.toString();  
g.drawString(msg, 10, 40);
```

```
}
```

```
}
```



Practical No. 9

1. Write an applet to print the message of click , enter, exit , press and release messages when respective Mouse event happens in the applet and print dragged and moved when respective mouse motion event happens in the applet.

NOTE: Implement MouseListener and MouseMotionListener Interfaces.

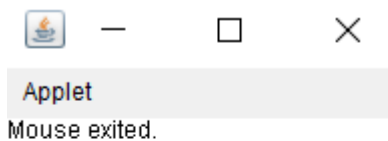
```
package lab_9;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="MouseEvents" width=300 height=100>
</applet>
*/
public class MouseEvents extends Applet implements MouseListener, MouseMotionListener {
    String msg = "";
    int mouseX = 0, mouseY = 0;

    public void init() {
        addMouseListener(this);
        addMouseMotionListener(this);
    }
    public void mouseClicked(MouseEvent me) {
        mouseX = 0;
        mouseY = 10;
        msg = "Mouse clicked.";
        repaint();
    }
    public void mouseEntered(MouseEvent me) {

        mouseX = 0;
        mouseY = 10;
        msg = "Mouse entered.";
        repaint();
    }
    public void mouseExited(MouseEvent me) {

        mouseX = 0;
        mouseY = 10;
        msg = "Mouse exited.";
        repaint();
    }
}
```

```
}  
public void mousePressed(MouseEvent me) {  
    mouseX = me.getX();  
    mouseY = me.getY();  
    msg = "mouse pressed";  
    repaint();  
}  
  
public void mouseReleased(MouseEvent me) {  
  
    mouseX = me.getX();  
    mouseY = me.getY();  
    msg = "mouse released";  
    repaint();  
}  
public void mouseDragged(MouseEvent me) {  
  
    mouseX = me.getX();  
    mouseY = me.getY();  
    msg = "*";  
    showStatus("Dragging mouse at " + mouseX + ", " + mouseY);  
    repaint();  
}  
public void mouseMoved(MouseEvent me) {  
  
    showStatus("Moving mouse at " + me.getX() + ", " + me.getY());  
}  
public void paint(Graphics g) {  
    g.drawString(msg, mouseX, mouseY);  
}  
}
```



2. Write an applet to print the message of click , enter, exit , press and release messages when respective Mouse event happens in the applet and print dragged and moved when respective mouse motion event happens in the applet.

NOTE: Extend MouseAdapter and MouseMotionAdapter class.

```
package lab_9;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="AdapterDemo" width=300 height=100>
</applet>
*/
public class AdapterDemo extends Applet {

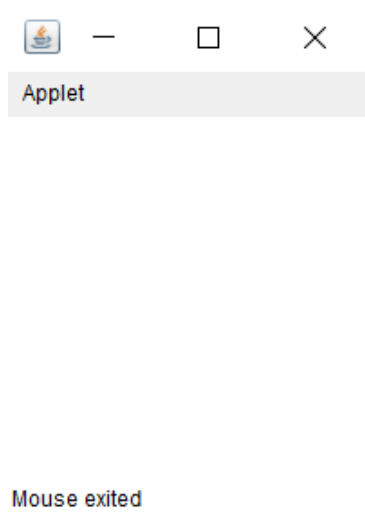
    public void init() {
        addMouseListener(new MyMouseAdapter(this));
        addMouseMotionListener(new MyMouseMotionAdapter(this));
    }
}
class MyMouseAdapter extends MouseAdapter {

    AdapterDemo adapterDemo;
    public MyMouseAdapter(AdapterDemo adapterDemo) {
        this.adapterDemo = adapterDemo;
    }
    public void mouseClicked(MouseEvent me) {
        adapterDemo.showStatus("Mouse clicked");
    }
    public void mouseEntered(MouseEvent me) {
        adapterDemo.showStatus("Mouse entered");
    }
    public void mousePressed(MouseEvent me) {
        adapterDemo.showStatus("Mouse pressed");
    }
    public void mouseReleased(MouseEvent me) {
        adapterDemo.showStatus("Mouse released");
    }
    public void mouseExited(MouseEvent me) {
        adapterDemo.showStatus("Mouse exited");
    }
}
```

```

class MyMouseMotionAdapter extends MouseMotionAdapter {
    AdapterDemo adapterDemo;
    public MyMouseMotionAdapter(AdapterDemo adapterDemo) {
        this.adapterDemo = adapterDemo;
    }
    public void mouseDragged(MouseEvent me) {
        adapterDemo.showStatus("Mouse dragged");
    }
    public void mouseMoved(MouseEvent me) {
        adapterDemo.showStatus("Mouse moved");
    }
}

```



3. Write an applet to print the message implementing KeyListener Interface. Also show the status of key press and release in status window.

```

package lab_9;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="KeyEvents" width=300 height=100>
</applet>
*/

public class KeyEvents extends Applet implements KeyListener {

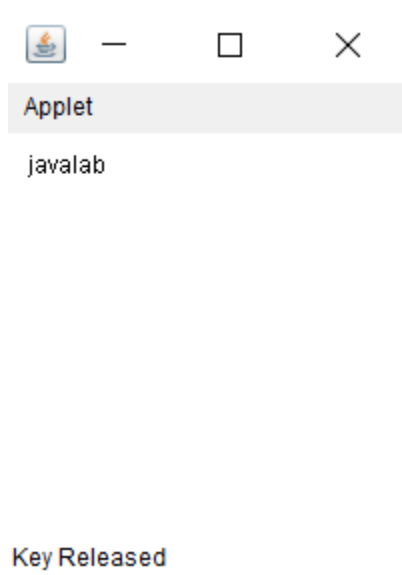
```

```

String msg = "";
int X = 10, Y = 20;
public void init() {
    addKeyListener(this);
}
public void keyPressed(KeyEvent ke) {
    showStatus("Key Pressed");
}
public void keyReleased(KeyEvent ke) {
    showStatus("Key Released");
}

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

```



4. Write an applet to print the message extending KeyAdapter class in Inner class and Anonymous Inner class. Also show the status of key press and release in status window.

```
package lab_9;
```

```

import java.applet.*;
import java.awt.Graphics;
import java.awt.event.*;
/*
<applet code="InnerClassKeyEvents" width=200 height=100>
</applet>
*/
public class InnerClassKeyEvents extends Applet {

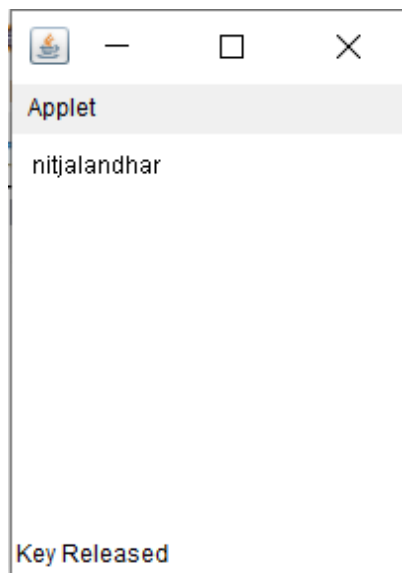
    String msg = "";
    int X = 10, Y = 20;
    public void init() {
        addKeyListener(new MyKeyAdapter());
    }
    class MyKeyAdapter extends KeyAdapter {
        public void keyPressed(KeyEvent ke) {
            showStatus("Key Pressed");
        }
        public void keyReleased(KeyEvent ke) {
            showStatus("Key Released");
        }
        public void keyTyped(KeyEvent ke) {
            msg += ke.getKeyChar();
            showStatus("Key typed");
            repaint();
        }
    }
    public void paint(Graphics g) {
        g.drawString(msg, X, Y);
    }
}

```



Key Released

```
package lab_9;
import java.applet.*;
import java.awt.Graphics;
import java.awt.event.*;
/*
<applet code="AnonymousInnerKeyEvents" width=200 height=100>
</applet>
*/
public class AnonymousInnerKeyEvents extends Applet {
    String msg = "";
    int X = 10, Y = 20;
    public void init() {
        addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent ke) {
                showStatus("key Pressed");
            }
            public void keyReleased(KeyEvent ke) {
                showStatus("Key Released");
            }
            public void keyTyped(KeyEvent ke) {
                msg += ke.getKeyChar();
                repaint();
            }
        });
    }
    public void paint(Graphics g) {
        g.drawString(msg, X, Y);
    }
}
```



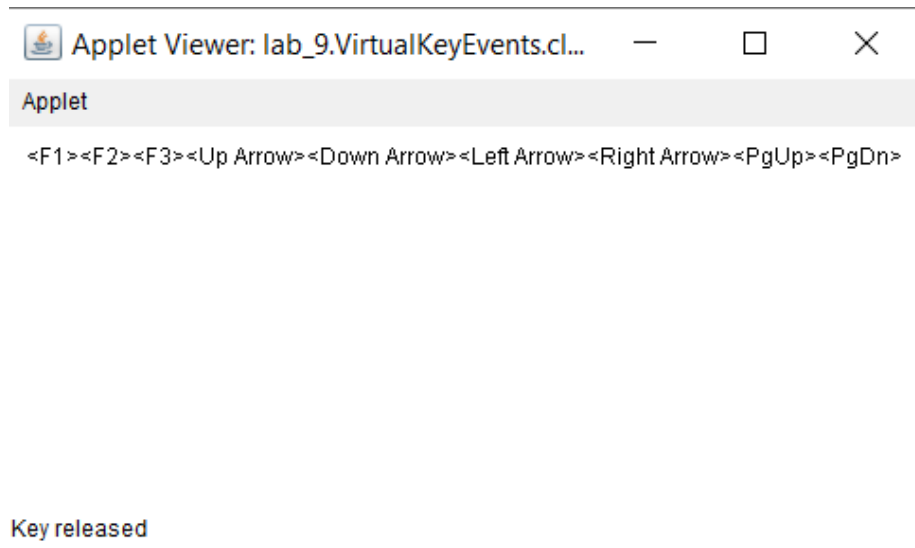
5. Write an applet demonstrating some virtual key codes i.e. Function keys, Page Up, Page Down and arrow keys. To demonstrate only print the message of the key pressed. Also show the status of key press and release .

```
package lab_9;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="VirtualKeyEvents" width=300 height=100>
</applet>
*/
public class VirtualKeyEvents extends Applet implements KeyListener {

    String msg = "";
    int X = 10, Y = 20;
    public void init() {
        addKeyListener(this);
    }
    public void keyPressed(KeyEvent ke) {
        showStatus("Key pressed");
        int key = ke.getKeyCode();
        switch(key) {
            case KeyEvent.VK_F1:
                msg += "<F1>";
                break;
            case KeyEvent.VK_F2:
                msg += "<F2>";
                break;
            case KeyEvent.VK_F3:
                msg += "<F3>";
                break;
            case KeyEvent.VK_PAGE_DOWN:
                msg += "<PgDn>";
                break;
            case KeyEvent.VK_PAGE_UP:
                msg += "<PgUp>";
                break;
            case KeyEvent.VK_LEFT:
                msg += "<Left Arrow>";
                break;
            case KeyEvent.VK_RIGHT:
                msg += "<Right Arrow>";
```

```
        break;
    case KeyEvent.VK_UP:
        msg += "<Up Arrow>";
        break;
    case KeyEvent.VK_DOWN:
        msg += "<Down Arrow>";
        break;
    }
    repaint();
}
public void keyReleased(KeyEvent ke) {
    showStatus("Key released");
}
public void keyTyped(KeyEvent ke) {
    msg += ke.getKeyChar();
    repaint();
}

public void paint(Graphics g) {
    g.drawString(msg, X, Y);
}
}
```



Practical No. 10

1. Write an applet to design a calculator. Calculations should be atleast done with two numbers.

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

public class Lab8 extends Applet implements ActionListener {
    String msg = " ";
    int v1, v2, result;
    TextField t1;
    Button b[] = new Button[10];
    Button add, sub, mul, div, EQ;
    char OP;

    public void init() {
        Color k = new Color(120, 89, 90);
        setBackground(k);
        t1 = new TextField(10);
        GridLayout gl = new GridLayout(4, 4);
        setLayout(gl);
        for (int i = 0; i < 10; i++) {
            b[i] = new Button("" + i);
        }
        add = new Button("add");
        sub = new Button("sub");
        mul = new Button("mul");
        div = new Button("div");
        EQ = new Button("EQ");
        t1.addActionListener(this);
        add(t1);
        for (int i = 0; i < 10; i++) {
            add(b[i]);
        }
        add(add);
        add(sub);
        add(mul);
        add(div);
        add(EQ);
        for (int i = 0; i < 10; i++) {
            b[i].addActionListener(this);
        }
        add.addActionListener(this);
        sub.addActionListener(this);
        mul.addActionListener(this);
```

```
div.addActionListener(this);
EQ.addActionListener(this);
}

public void actionPerformed(ActionEvent ae) {
    String str = ae.getActionCommand();
    char ch = str.charAt(0);
    if (Character.isDigit(ch))
        t1.setText(t1.getText() + str);
    else if (str.equals("add")) {
        v1 = Integer.parseInt(t1.getText());
        OP = '+';
        t1.setText("");
    } else if (str.equals("sub")) {
        v1 = Integer.parseInt(t1.getText());
        OP = '-';
        t1.setText("");
    } else if (str.equals("mul")) {
        v1 = Integer.parseInt(t1.getText());
        OP = '*';
        t1.setText("");
    } else if (str.equals("div")) {
        v1 = Integer.parseInt(t1.getText());
        OP = '/';
        t1.setText("");
    }
    if (str.equals("EQ")) {
        v2 = Integer.parseInt(t1.getText());
        if (OP == '+')
            result = v1 + v2;
        else if (OP == '-')
            result = v1 - v2;
        else if (OP == '*')
            result = v1 * v2;
        else if (OP == '/')
            result = v1 / v2;
        else if (OP == '%')
            result = v1 % v2;
        t1.setText("" + result);
    }
}

public void paint(Graphics g) {
    showStatus("Calculator");
}
```



2. Write an applet displaying three buttons and three checkboxes labelled Rectangle ,Oval, Line Red, Green and Blue respectively. Fill the Rectangle or Oval or draw the line with respective colour when it is checked.

```
import java.awt.*;
```

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

```
public class MyFrame extends Frame implements ActionListener, ItemListener {
```

```
    Checkbox c1, c2, c3;
```

```
    Button b1, b2, b3;
```

```
    CheckboxGroup c;
```

```
    int a, b;
```

```
    MyFrame () {
```

```
        setLayout(new FlowLayout());
```

```
        c = new CheckboxGroup();
```

```
        c1 = new Checkbox("Red", c, false);
```

```
        c2 = new Checkbox("Green", c, false);
```

```
        c3 = new Checkbox("Blue", c, false);
```

```
        b1 = new Button("Rectangle");
```

```
        b2 = new Button("Oval");
```

```
        b3 = new Button("Line");
```

```
        add(b1);
```

```
        add(b2);
```

```
        add(b3);
```

```
        add(c1);
```

```
        add(c2);
```

```
        add(c3);
```

```
        b1.addActionListener((ae) -> {
```

```
            b = 1;
```

```
            repaint();
```

```
        });
```

```
b2.addActionListener((ae) -> {
    b = 2;
    repaint();
});
b3.addActionListener((ae) -> {
    b = 3;
    repaint();
});
c1.addItemListener(this);
c2.addItemListener(this);
c3.addItemListener(this);
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}
public void itemStateChanged(ItemEvent arg0) {
    repaint();
}
@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub
}
public void paint(Graphics g) {
    String t;
    t = c.getSelectedCheckbox().getLabel();
    if (b == 1) { // rectangle
        if (t.equals("Red")) {
            setForeground(Color.RED);
            g.fillRect(100, 100, 20, 20);
        } else if (t.equals("Green")) {
            setForeground(Color.GREEN);
            g.fillRect(100, 100, 20, 20);
        } else if (t.equals("Blue")) {
            setForeground(Color.BLUE);
            g.fillRect(100, 100, 20, 20);
        }
    }

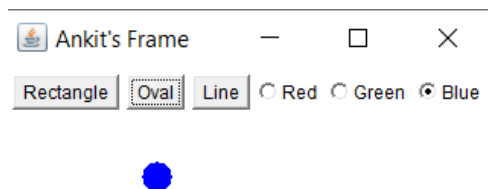
    } else if (b == 2) { // oval
        if (t.equals("Red")) {
            setForeground(Color.RED);
            g.fillOval(100, 100, 20, 20);
        } else if (t.equals("Green")) {
            setForeground(Color.GREEN);
        }
    }
}
```

```

        g.fillOval(100, 100, 20, 20);
    } else if (t.equals("Blue")) {
        setForeground(Color.BLUE);
        g.fillOval(100, 100, 20, 20);
    }
} else if (b == 3) { // line
    if (t.equals("Red")) {
        setForeground(Color.RED);
        g.drawLine(100, 100, 120, 120);
    } else if (t.equals("Green")) {
        setForeground(Color.GREEN);
        g.drawLine(100, 100, 120, 120);

    } else if (t.equals("Blue")) {
        setForeground(Color.BLUE);
        g.drawLine(100, 100, 120, 120);
    }
}
}
}
public static void main(String[] args) {
    MyFrame d = new MyFrame ();
    d.setSize(new Dimension(340, 260));
    d.setTitle("Ankit's frame");
    d.setVisible(true);
}
}

```



3. Write an applet to draw a square and two buttons labelled Enlarge and Shrink respectively. Enlarge and shrink the square when respective buttons are pressed.

```

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

public class EnlargeFrame extends Frame implements ActionListener {

```

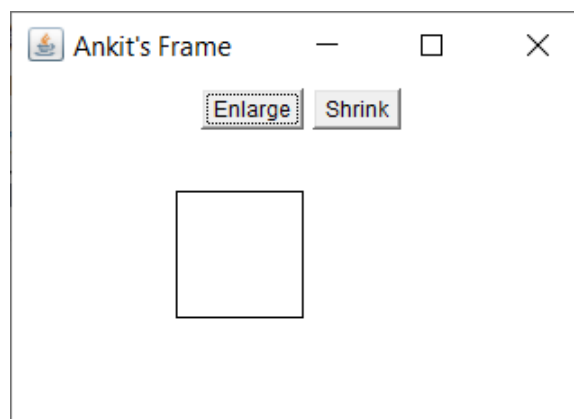


```
Button b1, b2;
int s = 50;
EnlargeFrame () {
    setLayout(new FlowLayout());
    b1 = new Button("Enlarge");
    b2 = new Button("Shrink");
    setForeground(Color.BLACK);
    add(b1);
    add(b2);
    b1.addActionListener((ae) -> {
        s = s + 10;
        repaint();
    });
    b2.addActionListener((ae) -> {
        s = s - 10;
        if (s < 20)
            s = 20;
        repaint();
    });
}

public void actionPerformed(ActionEvent arg0) {
}

public void paint(Graphics g) {
    g.drawRect(100, 100, s, s);
}

static public void main(String args[]) {
    EnlargeFrame d = new EnlargeFrame ();
    d.setSize(new Dimension(340, 260));
    d.setTitle("Ankit's Frame");
    d.setVisible(true);
}
}
```



4. Write an applet that change the font size and style.

```
package Lab10;
import java.awt.*;
import java.awt.event.*;

public class ChoiceList extends Applet implements ItemListener{

    Label l1,l2;
    String fn,fs;
    int size=40;
    Choice fontName,fontSize;
    String msg="EVENT HANDLING IN JAVA";
    Checkbox Centered,Left,Right,Bold,Italics;
    CheckboxGroup cbg;
    int px=50;
    public void init() {

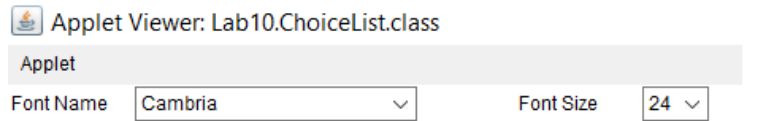
        l1=new Label("Font Name");
        l2=new Label("Font Size");
        GraphicsEnvironment ge=GraphicsEnvironment.getLocalGraphicsEnvironment();
        String[] fonts=ge.getAvailableFontFamilyNames();
        fontName=new Choice();
        fontSize=new Choice();
        for(String i:fonts){
            fontName.add(i);
        }
        for(int i=12;i<=50;++i){
            fontSize.add(""+i);
        }
        add(l1);add(fontName);
        add(l2);add(fontSize);

        fontName.addItemListener(this);
        fontSize.addItemListener(this);
        cbg=new CheckboxGroup();
        Centered=new Checkbox("Centered",cbg,false);
        Left=new Checkbox("Left",cbg,false);
        Right=new Checkbox("Right",cbg,false);
        Bold=new Checkbox("Bold",null,false);
        Italics=new Checkbox("Italics",null,false);
        add(Centered);
        add(Left);
```

```
        add(Right);
        add(Bold);
        add(Italics);
        Bold.addItemListener(this);
        Italics.addItemListener(this);
        Centered.addItemListener(this);
        Left.addItemListener(this);
        Right.addItemListener(this);
    }
    public void paint(Graphics g){
        fs=fontName.getSelectedItem();
        fn=fontSize.getSelectedItem();
        size=Integer.parseInt(fn);
        int x=0,y=0;
        if(Bold.getState())
            x=Font.BOLD;
        if(Italics.getState())
            y=Font.ITALIC;
        Font f=new Font(fs,x+y,size);
        g.setFont(f);
        this.l1.setLocation(0,0);
        this.fontName.setLocation(80,0);
        this.l2.setLocation(320,0);
        this.fontSize.setLocation(400,0);
        g.drawString(msg,px,80);
        this.Centered.setLocation(0,120);
        this.Left.setLocation(90,120);
        this.Right.setLocation(150,120);
        this.Bold.setLocation(0,140);
        this.Italics.setLocation(60,140);
    }
    // TODO overwrite start(), stop() and destroy() methods
    @Override
    public void itemStateChanged(ItemEvent e) {

        String s=cbg.getSelectedCheckbox().getLabel();
        if(s.equals("Centered"))
            px=50;
        else if(s.equals("Left"))
            px=0;
        else
            px=100;
```

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



EVENT HANDLING IN JAVA

☒ Centered ☐ Left ☐ Right
☒ Bold ☒ **Italics**

Practical No. 11

1. Write a program taking 16 buttons labelled 1 to 16 in a frame and represent them with FlowLayout and GridLayout. Print “Button <#> is pressed” message when a button is pressed.

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

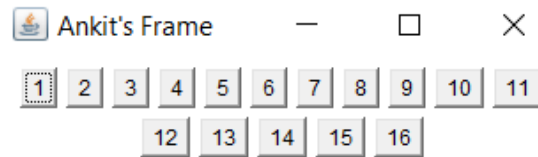
public class FlowlayoutFrame extends Frame implements ActionListener{

    String msg="";
    Button []b;
    FlowlayoutFrame ()
    {
        setLayout(new FlowLayout());
        int k;
        b=new Button[16];
        for( int i=0; i<16;++i)
        {
            k=i+1;
            add(b[i]=new Button(""+k));
        }
        for( int i=0;i<16;++i)
        {
            b[i].addActionListener(this);
        }
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent ae) {
        String s=ae.getActionCommand();
        msg="Button "+ s+ " is pressed ";
        repaint();
    }
    public void paint(Graphics g)
    {
        g.drawString(msg, 200, 200);
    }
    public static void main(String args[])
    {
        FlowlayoutFrame d= new FlowlayoutFrame ();
        d.setSize(new Dimension(340,260));
    }
}
```

```

        d.setTitle("Ankit's Frame");
        d.setVisible(true);
    }
}

```



GridLayout:

```

import java.awt.*;
import java.awt.event.*;
public class GridlayoutFrame extends Frame implements ActionListener{
    Label l;
    String msg="NONE";

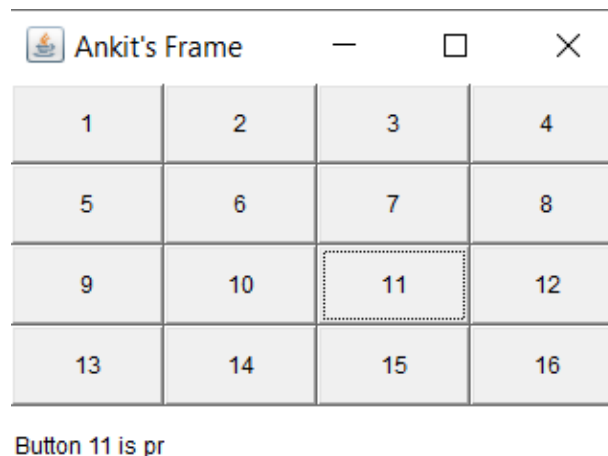
    Button []b;
    GridlayoutFrame ()
    {
        l=new Label("NoNe");
        setLayout(new GridLayout(5,4));
        int k;
        b=new Button[16];
        for( int i=0; i<16;++i)
        {
            k=i+1;
            add(b[i]=new Button(""+k));
        }
        add(l);
        for( int i=0;i<16;++i)
        {
            b[i].addActionListener(this);
        }
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });
    }
}

```

```

        }
    });
}
public void actionPerformed(ActionEvent ae) {
    String s=ae.getActionCommand();
    msg="Button "+ s+ "\n is pressed ";
    repaint();
}
public void paint(Graphics g)
{
    l.setText(msg);
}
public static void main(String[] args) {
    // TODO Auto-generated method stub
    GridLayoutFrame d= new GridLayoutFrame ();
    d.setSize(new Dimension(340,260));
    d.setTitle("Ankit's Frame");
    d.setVisible(true);
}
}

```



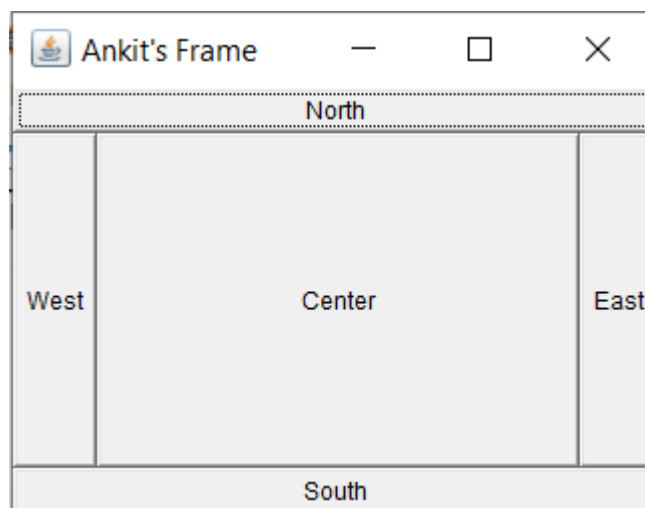
2. Write a program showing use of BorderLayout in a frame showing the directions i.e. North, East, South and West on buttons.

```

import java.awt.*;
import java.awt.event.*;
public class BorderLayoutFrame extends Frame implements ActionListener {
    Button a,b,c,d,e;
    BorderLayoutFrame ()

```

```
{
    setLayout(new BorderLayout());
    a=new Button("North");
    b=new Button("South");
    d=new Button("West");
    c=new Button("East");
    e=new Button("Center");
    add(a,BorderLayout.NORTH);
    add(b,BorderLayout.SOUTH);
    add(c,BorderLayout.EAST);
    add(d,BorderLayout.WEST);
    add(e,BorderLayout.CENTER);
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we)
        {
            System.exit(0);
        }
    });
}
public void actionPerformed(ActionEvent e) {
}
public static void main(String args[])
{
    BorderLayoutFrame d= new BorderLayoutFrame ();
    d.setSize(new Dimension(340,260));
    d.setTitle("Ankit's Frame");
    d.setVisible(true);
}
}
```



3. Write a program for menu demonstration same as in notepad extending Frame class. Menu bar should contain File, Edit, View and its submenus as in notepad. Show the message when clicked on the menu.

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

public class Notepad extends Frame implements ActionListener {
    TextArea t;
    Notepad ()
    {
        MenuBar mbar=new MenuBar();
        setMenuBar(mbar);
        MenuItem i1,i2,i3,i4;
        t=new TextArea("",100,100);
        /*
         * textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true)
        */
        add(t);
        Menu file=new Menu("File");
        file.add(i1=new MenuItem("New"));
        file.add(i2=new MenuItem("Open..."));
        file.add(i3=new MenuItem("Save"));
        file.add(i4=new MenuItem("Save As..."));
        mbar.add(file);

        Menu edit=new Menu("Edit");
        MenuItem e1,e2,e3,e4,e5;
        edit.add(e1=new MenuItem("Undo"));
        edit.add(e2=new MenuItem("Cut"));
        edit.add(e3=new MenuItem("Copy"));
        edit.add(e4=new MenuItem("Paste"));
        edit.add(e5=new MenuItem("Delete"));

        mbar.add(edit);
        MenuItem f1,f2;
        Menu format=new Menu("Format");
        format.add(f1=new MenuItem("Word Wrap"));
        format.add(f2=new MenuItem("Font..."));
        mbar.add(format);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we)
```

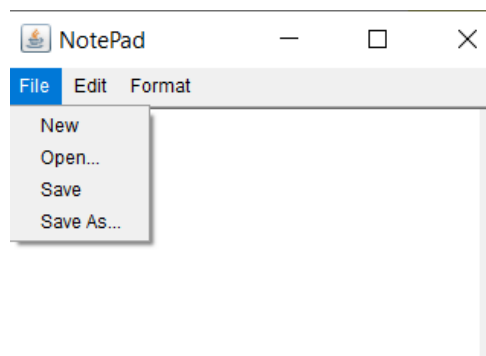
```

        {
            System.exit(0);
        }
    });

    Myhand hand=new Myhand();
    i1.addActionListener(hand);
    i2.addActionListener(hand);
    i3.addActionListener(hand);
    i4.addActionListener(hand);
    e1.addActionListener(hand);
    e2.addActionListener(hand);
    e3.addActionListener(hand);
    e4.addActionListener(hand);
    e5.addActionListener(hand);
    f1.addActionListener(hand);
    f2.addActionListener(hand);
}
class Myhand implements ActionListener
{
    public void actionPerformed(ActionEvent e) {
        String k=e.getActionCommand();
        t.setText("Pressed "+ k);
    }
}

public static void main(String[] args) {
    Notepad d= new Notepad();
    d.setSize(new Dimension(340,260));
    d.setTitle("NotePad");
    d.setVisible(true);
}
}

```



4.FileDialog control represents a dialog window from which the user can select a file. Write a program to demonstrate simple FileDialog using Frame class.

```
import java.awt.*;
import java.awt.event.*;
class SampleFrame extends Frame {
    SampleFrame(String title) {
        super(title);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);}
        });
    }
}
class Filedialog {
    public static void main(String args[]) {
        // create a frame that owns the dialog
        Frame f = new SampleFrame("File Dialog Demo");
        f.setVisible(true);
        f.setSize(100, 100);
        FileDialog fd = new FileDialog(f, "File Dialog");
        fd.setVisible(true);
    }
}
```

