

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY



UNIVERSITY COLLEGE OF ENGINEERING VIZIANAGARAM

Department of
INFORMATION TECHNOLOGY

Java Programming Lab Record

Bachelor of technology
INFORMATION TECHNOLOGY



REGD NO: 20VV1A1263

Certificate

Certified that this is a Bonafide Record of practical work done by

Mr./ Kumari VENKATA SAI YASWANTH BATTU

of II B TECH I SEMESTER class in JAVA PROGRAMMING Laboratories

of JNTUK-UCEV College during the Academic Year 2021-2022

No of experiments done and certified:

Lecturer in-Charge:

Head of Department

Date:

Index

S.No	Date	Name of the Experiment	Page	Marks	Signature
1		Write a JAVA program to display default value of all primitive data type of JAVA	2		
2		Write a java program that display the roots of a quadratic equation $ax^2+bx=0$. Calculate the discriminate D and basing on value of D, describe the nature of roots.	4		
3		Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers	6		
4		Write a JAVA program to search for an element in a given list of elements using binary search mechanism	8		
5		Write a JAVA program to sort a given list of elements using bubble sort	11		
6		Write a JAVA program to sort a given list of elements using merge sort.	13		
7		Write a JAVA program using String Buffer to delete, remove character	15		
8		Write a JAVA program to implement class mechanism. –Create a class, methods and invoke them inside main method	16		
9		Write a JAVA program to implement constructor.	17		
10		Write a JAVA program to implement constructor overloading	18		
11		Write a JAVA program implement method overloading	20		
		Write a JAVA program to implement Single Inheritance	21		

12					
13		Write a JAVA program to implement multi level Inheritance	23		
14		Write a java program for abstract class to find areas of different shapes.	24		
15		Write a JAVA program give example for <code>super</code> keyword	26		
16		Write a JAVA program to implement Interface. What kind of Inheritance can be achieve	28		
17		Write a JAVA program that describes exception handling mechanism	30		
18		Write a JAVA program Illustrating Multiple catch clause	32		
19		Write a JAVA program that implements Runtime polymorphism	34		
20		Write a JAVA program for creation of Illustrating throw	35		
21		Write a JAVA program for creation of Illustrating finally	36		
22		Write a JAVA program for creation of Java Built-in Exceptions	37		
23		Write a JAVA program for creation of User Defined Exception	39		
24		Write a JAVA program that creates threads by extending Thread class .First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds ,(Repeat the same by implementing Runnable)	40		
25		Write a program illustrating isAlive and join ()	43		
26		Write a Program illustrating Daemon Threads	45		

27		Write a JAVA program Producer Consumer Problem	46		
28		Write a JAVA program illustrate class path	48		
29		Write a JAVA program that import and use the defined your package in the previous Problem	49		
30		Write a JAVA program to paint like paint brush in applet	50		

31		Write a JAVA program to display analogy clock using Apple	52		
32		Write a JAVA program to create different shapes and fill colours using Applet	55		
33		Write a JAVA program that display the x and y position of the cursor movement using Mouse	56		
34		Write a JAVA program that identifies key-up key-down event user entering text in a Applet.	57		
35					
36					
37					
38					
39					
40					
41					

42					
43					
44					
45					
46					

JAVA PROGRAMMING LAB RECORD

BY 20VV1A1263,
2-1 B. TECH IT. (R20)

EXERCISE –1:(Basics)

A)AIM: Write a JAVA program to display default value of all primitive data type of JAVA

DESCRIPTION:

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java

1.Primitive data types - The primitive data types include boolean, char, byte, short,int, long, float and double.

2.Non-primitive data types - The non-primitive data types include Classes, Interfaces, and Arrays.

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

PROGRAM:

```
class primitive
{
    static String s;
    static int i;
    static float f;
    static double d;
    static long l;
    static boolean b;
    public static void main(String[] args)
    {
        System.out.println("DEFAULT VALUES OF : ");
        System.out.println("strings is : "+s);
        System.out.println("integers is : "+i);
        System.out.println("float is : "+f);
        System.out.println("double is : "+d);
        System.out.println("long is : "+l);
        System.out.println("booleans is : "+b);
    }
}
```


OUTPUT:

```
[Running] cd "c:\Users\Y  
DEFAULT VALUES OF :  
strings is : null  
integers is : 0  
float is : 0.0  
double is : 0.0  
long is : 0  
booleans is : false  
  
[Done] exited with code=0
```

B)AIM: Write a java program that display the roots of a quadratic equation $ax^2+bx=0$. Calculate the discriminate D and basing on value of D, describe the nature of root

DESCRIPTION:

- The standard form of a quadratic equation is: $ax^2 + bx + c = 0$.
- Here, a, b, and c are real numbers and a can't be equal to 0.
- We can calculate the root of a quadratic by using the formula: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
- The \pm sign indicates that there will be two roots: $\text{root1} = \frac{-b + \sqrt{b^2-4ac}}{2a}$ $\text{root2} = \frac{-b - \sqrt{b^2-4ac}}{2a}$
- The term b^2-4ac is known as the determinant of a quadratic equation.
- It specifies the nature of roots.
- That is,
- •if determinant > 0 , roots are real and different
- •if determinant $= 0$, roots are real and equal
- •if determinant < 0 , roots are complex complex and different.

PROGRAM:

```
java.util.*;
class Nature
{
    int d,a,b,c;
    Nature(int x,int y,int z){
        a = x;
        b =import y;
        c = z;
    }
    void disc(){
        d = (int)Math.pow(b,2)-4*a*c;
    }
    void roots(){
        if(d<0){
            System.out.println("Roots are Imaginary.");
        }
        else{
            if (d>0){
                System.out.println("Roots are Real and Distinct.");
                System.out.println("Roots are: "+(-b+Math.sqrt(d))/(2*a)+" "+(-b-
Math.sqrt(d))/(2*a));
            }
            else{
                System.out.println("Roots are Real and Equal.");
                System.out.println("Roots are: "+(-b+Math.sqrt(d))/(2*a)+" "+(-b-
Math.sqrt(d))/(2*a));
            }
        }
    }
}
```

```

    }
}

public static void main(String[] args){
    int a,b,c;

    Scanner s = new Scanner(System.in);
    System.out.println("Enter the Coefficients: ");

    a = s.nextInt();
    b = s.nextInt();
    c = s.nextInt();

    Nature n = new Nature(a,b,c);
    n.disc();
    n.roots();

}
}

```

OUTPUT:

```

C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 1>java Nature
Enter the Coefficients:
1 2 3
Roots are Imaginary.

C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 1>javac Nature.java

C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 1>java Nature
Enter the Coefficients:
1 2 1
Roots are Real and Equal.
Roots are: -1.0 -1.0

```

C)AIM: *Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racer*

DESCRIPTION:

- Given 5 Bike racers.
- Taking input from users from the users, we have to calculate average speed which can be done by summing all the speeds / total number of bike racers participated which is : $\text{Sum of all Speeds} / 5$
- after calculating the average speed, we have to compare each racer whether a given racer is greater than the average or not and print their name as qualifying racers which can be achieved by a simple for loop and if-else block

PROGRAM:

```
import java.util.*;
class racers{
    int[] a = new int[5];
    double avg;
    Scanner s = new Scanner(System.in);
    void input(){
        for(int i = 1;i<=5;i++){
            System.out.print("Input speed of racer "+i+": ");
            a[i-1] = s.nextInt();
        }
    }
    void avgs(){
        int sum = 0;
        for(int i=0;i<5;i++){
            sum += a[i];
        }

        avg = sum/a.length;

        for(int i =1;i<=5;i++)
        {
            if(a[i-1]>avg){
                System.out.println("racer no "+i+" speed is: "+a[i-1]);
            }
        }
    }

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

```
{  
    racers r = new racers();  
    r.input();  
    System.out.println("The list of Qualifying Racers are: ");  
    r.avgs();  
}  
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 1>java racers  
Input speed of racer 1: 25  
Input speed of racer 2: 36  
Input speed of racer 3: 54  
Input speed of racer 4: 52  
Input speed of racer 5: 49  
The list of Qualifying Racers are:  
racer no 3 speed is: 54  
racer no 4 speed is: 52  
racer no 5 speed is: 49
```

EXERCISE –2:(Operations, Expressions, Control-flow, Strings)

A)AIM: Write a JAVA program to search for an element in a given list of elements using binary search mechanism.

DESCRIPTION:

- Given a sorted array `arr[]` of n elements, write a function to search a given element x in `arr[]`.
- A simple approach is to do a linear search.
- The time complexity of the above algorithm is $O(n)$.
- Another approach to perform the same task is using Binary Search.
- Binary Search: Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
- Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.
- The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log n)$.
- We basically ignore half of the elements just after one comparison.
- 1. Compare x with the middle element.
- 2. If x matches with the middle element, we return the mid index.
- 3. Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element.
- So we recur for the right half. 4. Else (x is smaller) recur for the left half.

PROGRAM:

```
class BinarySearch
```

```

{
void Binarysearch(int[] a,int s,int e,int k)
{
    int mid;
    while(s<=e)
    {
        mid = (s+e)/2;

        if(a[mid]<k)
        {
            s=mid+1;
        }
        else if(a[mid]>k)
        {
            e = mid-1;
        }
        else
        {
            System.out.println("\nElement is Found at Index: "+mid);
            return;
        }
    }
    if(s>e)
    {
        System.out.println("\nElement is not Found");
    }
}

public static void main(String[] args)
{
    int[] a = {12,15,17,23,33,37,49,57,63};
    int k = 37;
    int e = a.length-1;

    BinarySearch B = new BinarySearch();
    B.Binarysearch(a,0,e,k);
}
}

```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>java BinarySearch  
Element is Found at Index: 5
```


B)AIM: Write a JAVA program to sort a given list of elements using bubble sort.

DESCRIPTION:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

PROGRAM:

```
class BubbleSort
{
    void bubbleSort(int[] a)
    {
        int n = a.length;
        int t = 0;

        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                if(a[i]>a[j])
                {
                    t = a[i];a[i]=a[j];a[j]=t;
                }
            }
        }
    }

    public static void main(String[] args)
    {
        int a[] = {11,21,45,67,34,43,10,3,2,14};

        BubbleSort b = new BubbleSort();

        System.out.print("The Un Sorted Array is: ");
        for(int i=0;i<a.length;i++){
            System.out.print(a[i]+" ");
        }

        System.out.println("");
        b.bubbleSort(a);

        System.out.print("The Sorted Array is: ");
        for(int i=0;i<a.length;i++){
            System.out.print(a[i]+" ");
        }
    }
}
```

```
}  
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>javac BubbleSort.java  
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>java BubbleSort  
The Un Sorted Array is: 11 21 45 67 34 43 10 3 2 14  
The Sorted Array is: 2 3 10 11 14 21 34 43 45 67
```

C)AIM: Write a JAVA program to sort a given list of elements using merge sort

DESCRIPTION:

- Merge sort is the algorithm which follows divide and conquer approach.
- Consider an array A of n number of elements.
- The algorithm processes the elements in 3 steps.
- 1.If A Contains 0 or 1 elements then it is already sorted, otherwise, Divide A into two sub-array of equal number of elements.
- 2.Conquer means sort the two sub-arrays recursively using the merge sort.
- 3.Combine the sub-arrays to form a single final sorted array maintaining the ordering of the array.
- The main idea behind merge sort is that, the short list takes less time to be sorted

PROGRAM:

```
class MergeSort
{
    void merge(int arr[],int l,int mid,int r)
    {
        int i,j,k;
        int n1 = mid-l+1;
        int n2 = r-mid;
        int[] a = new int[n1];int[] b = new int[n2];

        for(i=0;i<n1;i++)
        {
            a[i]=arr[l+i];
        }
        for(i=0;i<n2;i++)
        {
            b[i]=arr[mid+1+i];
        }
        i=0;j=0;k=l;
        while(i<n1 && j<n2)
        {
            if(a[i] < b[j])
            {
                arr[k]=a[i];
                k++;i++;
            }
            else
            {
                arr[k]=b[j];
                k++;j++;
            }
        }
        while(i<n1)
    }
```

```

    {
        arr[k]=a[i];
        k++;i++;
    }
    while(j<n2)
    {
        arr[k]=b[j];
        k++;j++;
    }

}
void Mergesort(int arr[],int l,int r)
{
    if(l<r)
    {
        int mid = (l+r)/2;
        Mergesort(arr,l,mid);
        Mergesort(arr,mid+1,r);
        merge(arr,l,mid,r);
    }
}
public static void main(String[] args)
{
    int[] a = {12,21,34,5,67,80,8,43,32,11};
    int l=0;int r=a.length-1;
    MergeSort m = new MergeSort();
    System.out.print("The Un Sorted Array is: ");
    for(int p=0;p<a.length;p++){
        System.out.print(a[p]+" ");
    }
    System.out.println("");
    m.Mergesort(a, l, r);
    System.out.print("The Sorted Array is: ");
    for(int p=0;p<a.length;p++){
        System.out.print(a[p]+" ");
    }
}
}

```

OUTPUT:

```

C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>java MergeSort
The Un Sorted Array is: 12 21 34 5 67 80 8 43 32 11
The Sorted Array is: 5 8 11 12 21 32 34 43 67 80
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>

```

D)AIM: Write a JAVA program using String Buffer to delete, remove character.

DESCRIPTION:

- *StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences*
- *.StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.*
-

PROGRAM:

```
import java.io.*;
class StrBuffer
{
    public static void main(String[] args)
    {
        StringBuffer s = new StringBuffer("This is a un deleted string.");
        System.out.print("Original String: ");
        System.out.println(s);
        s.delete(9,12);
        System.out.print("AFter Deletion String: ");
        System.out.println(s);
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 2>java StrBuffer
Original String: This is a un deleted string.
After Deletion String: This is a deleted string.
```

EXERCISE –3:(Class, Objects)

A)AIM: Write a JAVA program to implement class mechanism. –Create a class, methods and invoke them inside main method.

DESCRIPTION:

- Class in Java
- A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
- A class in Java can contain:
 - Fields
 - Methods
 - Constructors
 - Blocks
 - Nested class and interface
- Method in JavaIn Java, a method is like a function which is used to expose the behavior of an object.Advantage of Method
 - Code Reusability
 - Code Optimization.
-

PROGRAM:

```
public class ClassMethods
{
    static String s="Hello!";
    void print_(){
        System.out.println(s);
    }
    void add(int a,int b){
        System.out.println("The Sum of Two numbers is: "+a+b);
    }
    public static void main(String[] args){
        ClassMethods c = new ClassMethods();
        c.print_();
        c.add(8,0);
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 3>java ClassMethods
Hello!
The Sum of Two numbers is: 80
```

B)AIM: Write a JAVA program to implement constructor

DESCRIPTION:

- In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created.
- At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object.
- Every time an object is created using the new() keyword, at least one constructor is called. It calls a default constructor if there is no constructor available in the class.
- In such case, Java compiler provides a default constructor by default.
- There are two types of constructors in Java: no-arg constructor, and parameterized constructor.
- Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

PROGRAM:

```
class Constructors
{
    // default constructor
    Constructors()
    {
        System.out.println("Default Constructor");
    }
    // parametricized constructor
    Constructors(float f,int i)
    {
        System.out.println("Parametricized Constructor");
    }
    public static void main(String[] args){
        Constructors c = new Constructors();
        Constructors c1 = new Constructors(8.97f,3);
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 3>java Constructors
Default Constructor
Parametricized Constructor
```

EXERCISE –4:(Methods)

A)AIM: Write a JAVA program to implement constructor overloading

DESCRIPTION:

- *In Java, a constructor is just like a method but without return type.*
- *It can also be overloaded like Java methods.*
- *Constructor overloading in Java is a technique of having more than one constructor with different parameter lists.*
- *They are arranged in a way that each constructor performs a different task.*
- *They are differentiated by the compiler by the number of parameters in the list and their types*

PROGRAM:

```
class ConsOver{
    // default constructor
    ConsOver(){
        System.out.println("Constructor 1");
    }
    // parametricized constructors
    // Overloading the constructor
    ConsOver(int i,int j)
    {
        System.out.println("Constructor 2");
    }
    ConsOver(float f, int j){
        System.out.println("Constructor 3");
    }
    ConsOver(float f, float j){
        System.out.println("Constructor 4");
    }
    ConsOver(float f, int j, String s){
        System.out.println("Constructor 5");
    }
    public static void main(String[] args){
        ConsOver c = new ConsOver();
        ConsOver c1 = new ConsOver(1,2);
        ConsOver c2 = new ConsOver(2.3f,9);
        ConsOver c3 = new ConsOver(7.2f,4.3f);
        ConsOver c4 = new ConsOver(8.2f,6,"kishore");
    }
}
```

OUTPUT:


```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 4>javac ConsOver.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 4>java ConsOver
Constructor 1
Constructor 2
Constructor 3
Constructor 4
Constructor 5
```

B)AIM: Write a JAVA program implement method overloading.

DESCRIPTION:

- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
- If we have to perform only one operation, having same name of the methods increases the readability of the program.
- Suppose we have to perform addition of the given numbers but there can be any number of arguments, if we write the method such as `a(int,int)` for two parameters, and `b(int,int,int)` for three parameters then it may be difficult for us as well as other programmers to understand the behavior of the method because its name differs.
- So, we perform method overloading to figure out the program quick.
-

PROGRAM:

```
class MethodOver
{
    void method()
    {
        System.out.println("method 1");
    }
    void method(float f)
    {
        System.out.println("method 2");
    }
    void method(String s)
    {
        System.out.println("method 3");
    }

    public static void main(String[] args){
        MethodOver m = new MethodOver();
        m.method();
        m.method(9.8f);
        m.method("ramesh");
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 4>javac MethodOver.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 4>java MethodOver
method 1
method 2
method 3
```

EXERCISE –5:(Inheritance)

A)AIM: Write a JAVA program to implement Single Inheritance.

DESCRIPTION:

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
 - It is an important part of OOPs (Object Oriented programming system).
 - The idea behind inheritance in Java is that we can create new classes that are built upon existing classes.
 - When we inherit from an existing class, we can reuse methods and fields of the parent class.
 - Moreover, we can add new methods and fields in your current class also.
 - Inheritance represents the IS-A relationship which is also known as a parent-child relationship. Why use inheritance in java
 - For Method Overriding (so runtime polymorphism can be achieved).
 - For Code Reusability. Types of inheritance in java
- On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.
- Single Inheritance Example
- When a class inherits another class, it is known as a single inheritance

PROGRAM:

```
class singleinheri{
    public static void main(String args[]){
        students st = new students();
        System.out.println("MARKS OF STUDENT:");
        System.out.println("Maths: "+st.m);
        System.out.println("Chemistry: "+st.c);
        System.out.println("Physics: "+st.p);
        System.out.println("English: "+st.e);
        System.out.println("Sanskrit: "+st.s);
    }
}

class marks{
    int m = 75;
    int c = 59;
    int p = 60;
    int e = 77;
    int s = 98;
}

class students extends marks{
    //inherits all variables of class marks
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 5>javac singleinheri.java
```

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 5>java singleinheri
```

```
MARKS OF STUDENT:
```

```
Maths: 75
```

```
Chemistry: 59
```

```
Physics: 60
```

```
English: 77
```

```
Sanskrit: 98
```

B)AIM: Write a JAVA program to implement multi level Inheritance.

DESCRIPTION:

- When a class extends a class, which extends another class then this is called multilevel inheritance.
- For example class C extends class B and class B extends class A then this type of inheritance is known as multilevel inheritance.
- Multilevel inheritance there is a concept of grand parent class.
- If we take the example of this diagram, then class C inherits class B and class B inherits class A which means B is a parent class of C and A is a parent class of B.
- So in this case class C is implicitly inheriting the properties and methods of class A along with class B that's what is called multilevel inheritance
-

PROGRAM:

```
public class mullvlinh {  
    public static void main(String[] args){  
        intrest cal = new intrest();  
        System.out.println("Intrest Amount to be paid is: "+cal.in);  
    }  
}  
class deposit{  
    static float bal = 70000f;  
}  
class inrate extends deposit{  
    static float i = 7.23f;  
}  
class intrest extends inrate{  
    static float in = (bal*i)/100;  
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 5>javac mullvlinh.java  
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 5>java mullvlinh  
Intrest Amount to be paid is: 5061.0
```

C)AIM: Write a java program for abstract class to find areas of different shapes

DESCRIPTION:

- A class which is declared as abstract is known as an abstract class.
- It can have abstract and non-abstract methods. It needs to be extended and its method implemented.
- It cannot be instantiated.
- Points to Remember
 - An abstract class must be declared with an abstract keyword.
 - It can have abstract and non-abstract methods.
 - It cannot be instantiated.
 - It can have constructors and static methods also.
 - It can have final methods which will force the subclass not to change the body of the method.
-

PROGRAM:

```
import java.util.*;
public class abstareas {
    public static void main(String[] args){
        Scanner s = new Scanner(System.in);
        extension sh = new extension();
        sh.circle(s.nextInt());
        sh.rectangle(s.nextInt(),s.nextInt());
        sh.square(s.nextInt());
    }
}

abstract class Shapes{
    abstract void circle(int x);
    abstract void rectangle(int x,int y);
    abstract void square(int x);
}

class extension extends Shapes{
    void circle(int x){
        System.out.println("Area of circle: "+(3.14*Math.pow(x,2)));
    }
    void rectangle(int x,int y){
        System.out.println("Area of rectangle: "+x*y);
    }
    void square(int x){
        System.out.println("Area of square: "+Math.pow(x,2));
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 5>java abstareas
5
Area of circle: 78.5
5 7
Area of rectangle: 35
7
Area of square: 49.0
```

EXERCISE –6: (Inheritance -Continued)

A)AIM: Write a JAVA program give example for “super” keyword

DESCRIPTION:

- The super keyword in Java is a reference variable which is used to refer immediate parent class object.
- Whenever we create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.
- Usage of Java super Keyword:
 - 1.super can be used to refer immediate parent class instance variable.
 - 2.super can be used to invoke immediate parent class method.
 - 3.super() can be used to invoke immediate parent class constructor

PROGRAM:

```
class sup {
    String s = "This is Super Class's Variable.";
    void ntg(){
        System.out.println("This is Super Class's method.");
    }

    sup(){
        System.out.println("This is Super Class's Constructor.");
    }
}
class Sub extends sup{
    String s = "This is Sub Class's Variable.";
    void ntg(){
        System.out.println(super.s);
        System.out.println("This is Sub Class's method.");
        super.ntg();
    }
    Sub(){
        System.out.println("This is Sub Class's Constructor.");
    }
}
class superkey{
    public static void main(String[] args){
        Sub i = new Sub();
        System.out.println(i.s);
        i.ntg();
    }
}
```


OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 6>java superkey  
This is Super Class's Constructor.  
This is Sub Class's Constructor.  
This is Sub Class's Variable.  
This is Super Class's Variable.  
This is Sub Class's method.  
This is Super Class's method.
```

B)AIM: Write a JAVA program to implement Interface. What kind of Inheritance can be achieve.

DESCRIPTION:

- An interface in Java is a blueprint of a class.
- It has static constants and abstract methods.
- The interface in Java is a mechanism to achieve abstraction.
- There can be only abstract methods in the Java interface, not method body.
- It is used to achieve abstraction and multiple inheritance in Java. In other words, we can say that interfaces can have abstract methods and variables.
- It cannot have a method body.
- Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class.
- Since Java 8, we can have default and static methods in an interface.
- Since Java 9, we can have private methods in an interface.

PROGRAM:

```
interface Vehicle
{
    void ChangeGear(int a);void SpeedUp(int b);void ApplyBrakes(int c);
}
```

```
class Bicycle implements Vehicle
{
    int s;int g;

    public void ChangeGear(int a)
    {
        g = a;
    }
    public void SpeedUp(int b)
    {
        s+=b;
    }
    public void ApplyBrakes(int c)
    {
        s-=c;
    }
    void states()
    {
        System.out.println("Speed: "+s+" gear: "+g);
    }
}
```

```
}  
  
class Interfaces  
{  
    public static void main(String[] args)  
    {  
        Bicycle b = new Bicycle();  
        b.ChangeGear(3);b.SpeedUp(2);b.ApplyBrakes(2);  
  
        System.out.println("Bicycle present state: ");  
        b.states();  
    }  
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXCERCISE 6>java Interfaces  
Bicycle present state:  
Speed: 0 gear: 3
```

EXERCISE –7:(Exception)

a) AIM: Write a JAVA program that describes exception handling mechanism

DESCRIPTION:

- The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- What is Exception in Java
- Dictionary Meaning: Exception is an abnormal condition.
- In Java, an exception is an event that disrupts the normal flow of the program.
- It is an object which is thrown at runtime.
- What is Exception HandlingException :
- Handling is a mechanism to handle runtime errors such as
ClassNotFoundException, IOException, SQLException, RemoteException.
-

PROGRAM:

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

    public static void main(String[] args)
    {
        // declaring two integer variables
        int a,b;
        // creating Scanner Object
        Scanner s = new Scanner(System.in);
        // reading values from user
        System.out.print("Enter two numbers: ");
        a = s.nextInt();
        b = s.nextInt();
        try
        {
            System.out.println();
            System.out.println("The Division of Numbers is: "+a/b);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Enter Proper Numbers.");
            System.out.print("Enter two numbers: ");

            // reading values from user

            a = s.nextInt();
            b = s.nextInt();
        }
    }
}
```

```
System.out.println("The Division of Numbers is: "+a/b);  
    }  
    }  
}
```

OUTPUT:

```
NG LAB_55021e85\bin' 'ExceptionHandling'  
Enter two numbers: 5 0  
  
Enter Proper Numbers.  
Enter two numbers: 5 10  
The Division of Numbers is: 0
```

b)AIM: *Write a JAVA program Illustrating Multiple catch clause*

DESCRIPTION:

- *In Java, a single try block can have multiple catch blocks. When statements in a single try block generate multiple exceptions, we require multiple catch blocks to handle different types of exceptions. This mechanism is called multi-catch block in java.*
- *Each catch block is capable of catching a different exception. That is each catch block must contain a different exception handler.*
- *Java multi-catch block acts as a case in a switch statement. In the above syntax, if the exception object (ExceptionType1) is thrown by try block, the first catch block will catch the exception object thrown and the remaining catch block will be skipped.*
- *In case, exception object thrown does not match with the first catch block, the second catch block will check, and so on.*
- *At a time only one exception occurs and at a time only one catch block is used.*
- *All catch blocks must be arranged in such a way that exception must come from the first subclass and then superclass.*

PROGRAM:

```
public class MultipleCatch
{
    public static void main(String args[])
    {
        int[] arr = new int[]{0,1,3,9,10};

        try
        {
            System.out.println("Dividing Two No's: "+arr[1]/arr[0]);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Arithmetic Exception Occured.");

            try
            {
                System.out.println("Accesing element in the array: "+arr[4]);
            }
            catch(ArrayIndexOutOfBoundsException a)
            {
                System.out.println("Array Index Out of Bound Exception Occured.");
            }
        }
    }
}
```

```
        System.out.println("No Abnormal Termination Occured.");  
    }  
}
```

OUTPUT:

A screenshot of a Java program's output. The text is displayed on a dark background with light green and white characters. The first line is the program name in quotes: \bin' 'MultipleCatch'. The second line is 'Arithmetic Exception Occured.'. The third line is 'Accesing element in the array: 10'. The fourth line is 'No Abnormal Termination Occured.'.

```
\bin' 'MultipleCatch'  
Arithmetic Exception Occured.  
Accesing element in the array: 10  
No Abnormal Termination Occured.
```

EXERCISE –8:(Runtime Polymorphism

a)AIM: Write a JAVA program that implements Runtime polymorphism

DESCRIPTION:

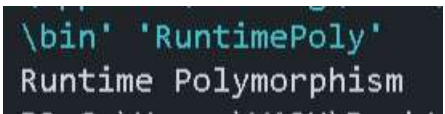
- *Runtime Polymorphism in Java Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.*
- *In this process, an overridden method is called through the reference variable of a superclass.*
- *The determination of the method to be called is based on the object being referred to by the reference variable.*
- *Upcasting If the reference variable of Parent class refers to the object of Child class, it is known as upcasting.*
-

PROGRAM:

```
class RuntimePoly extends Base
{
    void method()
    {
        System.out.println("Runtime Polymorphism");
    }
    public static void main(String[] args)
    {
        Base b = new RuntimePoly();
        b.method();
    }
}

class Base
{
    void method()
    {
        System.out.println("Base");
    }
}
```

OUTPUT:



```
\bin' 'RuntimePoly'
Runtime Polymorphism
```


EXERCISE –9:(User defined Exception)

a)AIM: Write a JAVA program for creation of Illustrating throw

DESCRIPTION:

- *The throw keyword in Java is used to explicitly throw an exception from a method or any block of code.*
- *We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.*
- *But this exception i.e, Instance must be of type Throwable or a subclass of Throwable.*
- *. For example Exception is a sub-class of Throwable and user defined exceptions typically extend Exception class.*
- *Unlike C++, data types such as int, char, floats or non-throwable classes cannot be used as exceptions.*
- *The flow of execution of the program stops immediately after the throw statement is executed and the nearest enclosing try block is checked to see if it has a catch statement that matches the type of exception.*
- *If it finds a match, control is transferred to that statement otherwise next enclosing try block is checked and so on.*
- *If no matching catch is found then the default exception handler will halt the program.*

PROGRAM:

```
class throwKeyWord
{   public static void main(String[] args)
    {
        try
        {
            throw new StringIndexOutOfBoundsException("Accessing Out of Index of array.");
        }
        catch(StringIndexOutOfBoundsException e)
        {
            System.out.println("Exception Occured : "+e);
        }
    }
}
```

OUTPUT:

```
java -cp throwkeyword
Exception Occured : java.lang.StringIndexOutOfBoundsException: Accessing Out of Index of array.
```

b)AIM: Write a JAVA program for creation of Illustrating finally block.

DESCRIPTION:

- The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection. The finally block executes whether exception rise or not and whether exception handled or not.
- A finally contains all the crucial statements regardless of the exception occurs or not.
- There are 3 possible cases where finally block can be used:
- Case 1: When an exception does not rise.
- Case 2: When the exception rises and handled by the catch block.
- Case 3: When exception rise and not handled by the catch blocks..
-

PROGRAM:

```
import java.io.FileNotFoundException;
class finallyKeyWord
{
    public static void main(String[] args)
    {
        try
        {
            System.out.println("Inside try block.");
            throw new FileNotFoundException("File Does't Exist.");
        }
        catch (FileNotFoundException e)
        {
            System.out.println("Inside catch block.");
            System.out.println("Exception: "+e);
        }
        finally
        {
            System.out.println("I always execute.");
        }
    }
}
```

OUTPUT:

```
\bin' 'finallyKeyWord'
Inside try block.
Inside catch block.
Exception: java.io.FileNotFoundException: File Does't Exist.
I always execute.
```

c)AIM: Write a JAVA program for creation of Java Built-in Exceptions

DESCRIPTION:

- Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.Examples of Built-in Exception:
- 1.Arithmetic exception : It is thrown when an exceptional condition has occurred in an arithmetic operation.
- 2.ArrayIndexOutOfBoundsException : It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
- 3.ClassNotFoundException : This Exception is raised when we try to access a class whose definition is not found.
- 4.FileNotFoundException : This Exception is raised when a file is not accessible or does not open.
- 5.IOException : It is thrown when an input-output operation failed or interrupted.
- 6.InterruptedOperationException : It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
- 7.NoSuchMethodException : It is thrown when accessing a method which is not found.
- 8.NullPointerException : This exception is raised when referring to the members of anull object. Null represents nothing.
- 9.NumberFormatException : This exception is raised when a method could not convert a string into a numeric format.
- 10.StringIndexOutOfBoundsException : It is thrown by String class methods to indicate that an index is either negative than the size of the string.
- 11.IllegalArgumentException : The Exception occurs explicitly either by the programmer or by API developer to indicate that a method has been invoked with Illegal Argument.
- 12.IllegalThreadStateException :
- The above exception rises explicitly either by programmer or by API developer to indicate that a method has been invoked at wrong time.
- 13.AssertionError : The above exception rises explicitly by the programmer or by API developer to indicate that assert statement fails.
-

PROGRAM:

```
import java.io.IOException;

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

```
        throw new IOException("IO Exception Occured.");
    }
    catch(IOException i)
    {
        System.out.println("Exception: "+i);
    }
}
}
```

OUTPUT:

```
56018e722d6ff29c3eb15e0d5b98c7f\redhat.java\jdt_ws\JAVA PROGRAMMING LAB_55021e85\bin' 'BuiltInException'
Exception: java.io.IOException: IO Exception Occured.
```

d)AIM: Write a JAVA program for creation of User Defined Exception

DESCRIPTION:

- If we are creating your own Exception that is known as custom exception or user-defined exception.
- Java custom exceptions are used to customize the exception according to user need.
- By the help of custom exception, we can have your own exception and message.
- We do this by extending our custom exception class with java Exception class and throwing explicitly when needed.
-

PROGRAM:

```
class UserDefinedException extends Exception
{
    public String toString()
    {
        return "UserDefinedException";
    }
    public static void main(String[] args)
    {
        try
        {
            System.out.println("Inside try block.");
            throw new UserDefinedException();
        }
        catch(UserDefinedException e)
        {
            System.out.println("Inside catch block.");
            System.out.println("Exception: "+e);
        }
    }
}
```

OUTPUT:

```
Inside try block.
Inside catch block.
Exception: UserDefinedException
```

EXERCISE –10:(Threads)

a) AIM: Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable

DESCRIPTION:

- Multithreading in Java is a process of executing multiple threads simultaneously.
- A thread is a lightweight sub-process, the smallest unit of processing.
- Multiprocessing and multithreading, both are used to achieve multitasking.
- However, we use multithreading than multiprocessing because threads use a shared memory area.
- They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.
- Advantages of Java Multithreading
- 1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- 2) You can perform many operations together, so it saves time.
- 3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.
- Thread in javaA thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution. Threads are independent. If there occurs exception in one thread, it doesn't affect other threads.
- It uses a shared memory area. A thread is executed inside the process. There is context-switching between the threads.
- There can be multiple processes inside the OS, and one process can have multiple threads. Java Thread class Java provides Thread class to achieve thread programming.
- Thread class provides constructors and methods to create and perform operations on a thread.
- Thread class extends Object class and implements Runnable interfaces

PROGRAM:

```
class RT1 implements Runnable
{
    public void run()
    {
        for(int i=0;i<8;i++)
        {
            try
```

```

    {
        System.out.println("Good Morning");
        Thread.sleep(1000);
    }
    catch(InterruptedException e)
    {
        System.out.println("Thread is Interrupted");
    }
}
}
}

```

class RT2 implements Runnable

```

{
    public void run()
    {
        for(int i=0;i<8;i++)
        {
            try
            {
                System.out.println("Hello");
                Thread.sleep(2000);
            }
            catch(InterruptedException e)
            {
                System.out.println("Thread is Interrupted");
            }
        }
    }
}

```

class RT3 implements Runnable

```

{
    public void run()
    {
        for(int i=0;i<8;i++)
        {
            try
            {
                System.out.println("Welcome");
                Thread.sleep(3000);
            }
            catch(InterruptedException e)
            {

```

```

        System.out.println("Thread is Interrupted");
    }
}
}

}

class RunnableThreads
{
    public static void main(String[] args)
    {
        Thread t1 = new Thread(new RT1());
        Thread t2 = new Thread(new RT2());
        Thread t3 = new Thread(new RT3());
        t1.start();t2.start();t3.start();
    }
}

```

OUTPUT:

```

56018e722d6ff29
Hello
Good Morning
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
Good Morning
Hello
Welcome
Good Morning
Good Morning
Hello
Welcome
Hello
Welcome
Hello
Hello
Welcome
Welcome
Welcome

```


b)AIM: Write a program illustrating `isAlive` and `join` .

DESCRIPTION:

- The `isAlive()` method of thread class tests if the thread is alive.
- A thread is considered alive when the `start()` method of thread class has been called and the thread is not yet dead.
- This method returns `true` if the thread is still running and not finished.
- The `join()` method waits for a thread to die.
- In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

PROGRAM:

```
class Thread1 extends Thread
{
    public void run()
    {
        for(int i = 0; i < 8; i++)
        {
            System.out.println(i);
        }
    }
}

class AJ
{
    public static void main(String[] args)
    {
        Thread1 t = new Thread1();
        t.start();
        System.out.println("t is alive: " + t.isAlive());
        try
        {
            t.join();
        }
        catch (InterruptedException e)
        {}
        System.out.println("t is alive: " + t.isAlive());
    }
}
```

OUTPUT:

```
\bin' 'AJ'  
t is alive: true  
0  
1  
2  
3  
4  
5  
6  
7  
t is alive: false
```

c)AIM: Write a Program illustrating Daemon Threads

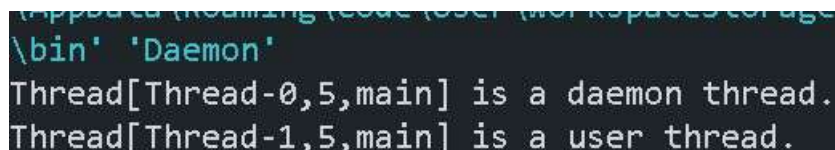
DESCRIPTION:

- *Daemon thread in java is a service provider thread that provides services to the user thread.*
- *Its life depend on the mercy of user threads i.e. when all the user threads dies, JVMterminates this thread automatically.*
- *There are many java daemon threads running automatically e.g. gc, finalizer etc.*
- *You can see all the detail by typing the jconsole in the command prompt.*
- *The jconsole tool provides information about the loaded classes, memory usage, running threads etc.*
- *Points to remember for Daemon Thread in Java*
 - *It provides services to user threads for background supporting tasks. It has no role in life than to serve user threads.*
 - *Its life depends on user threads.*
 - *It is a low priority thread.*

PROGRAM:

```
class Daemon extends Thread
{
    public void run()
    {
        if(Thread.currentThread().isDaemon())
        {
            System.out.println(this+" is a daemon thread.");
        }
        else
        {
            System.out.println(this+" is a user thread.");
        }
    }
    public static void main(String[] args)
    {
        Daemon d = new Daemon();
        Daemon d1 = new Daemon();
        d.setDaemon(true);
        d.start();
        d1.start();
    }
}
```

OUTPUT:



```
Thread[Thread-0,5,main] is a daemon thread.
Thread[Thread-1,5,main] is a user thread.
```

EXERCISE-11:(Threads continuity)

a)AIM: Write a JAVA program Producer Consumer Problem.

DESCRIPTION:

- In computing, the producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem.
- The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.
 - The producer's job is to generate data, put it into the buffer, and start again.
 - At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.
-

PROGRAM:

```
class Producer extends Thread
{
    StringBuffer sb = new StringBuffer();
    public void run()
    {
        synchronized(sb)
        {
            for(int i=0;i<=10;i++)
            {
                sb.append(i+" : ");
                System.out.println("Appending");

                try
                {
                    Thread.sleep(100);
                }
                catch(InterruptedException e)
                {}
            }
            sb.notify();
        }
    }
}

class Consumer extends Thread
{
    Producer prod;
    Consumer(Producer prod)
    {
        this.prod = prod;
    }
}
```

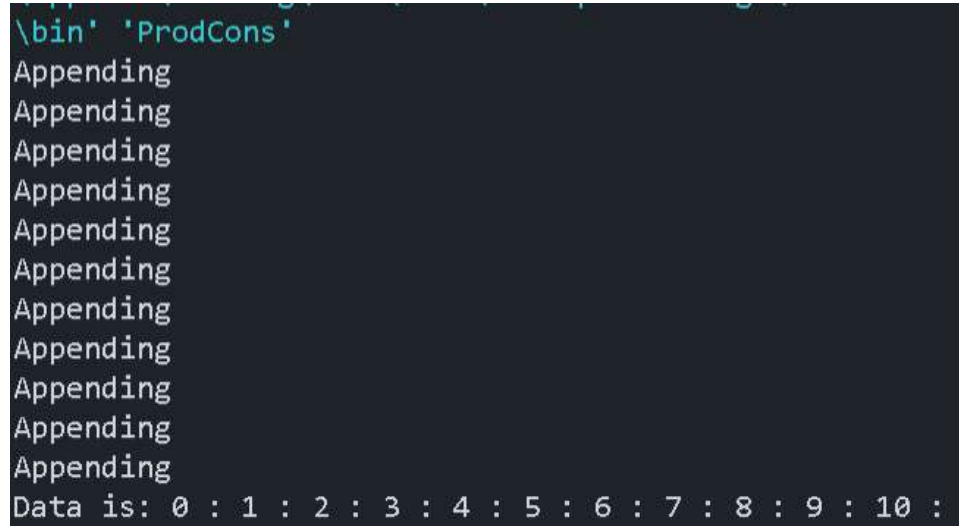
```

public void run()
{
    synchronized(prod.sb)
    {
        try
        {
            prod.sb.wait();
        }
        catch(InterruptedException e)
        {}
        System.out.println("Data is: "+prod.sb);
    }
}

class ProdCons
{
    public static void main(String[] args)
    {
        Producer p = new Producer();
        Consumer c = new Consumer(p);
        Thread t1 = new Thread(p);
        Thread t2 = new Thread(c);
        t2.start();
        t1.start();
    }
}

```

OUTPUT:



```

\bin' 'ProdCons'
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Appending
Data is: 0 : 1 : 2 : 3 : 4 : 5 : 6 : 7 : 8 : 9 : 10 :

```

EXERCISE –12:(Packages)

a)**AIM:** Write a JAVA program illustrate class path.

DESCRIPTION:

- **CLASSPATH:** CLASSPATH is an environment variable which is used by Application ClassLoader to locate and load the .class files.
- The CLASSPATH defines the path, to find third-party and user-defined classes that are not extensions or part of Java platform.
- Include all the directories which contain .class files and JAR files when setting the CLASSPATH.
- You need to set the CLASSPATH if:
 - You need to load a class that is not present in the current directory or any sub-directories.
 - You need to load a class that is not in a location specified by the extensions mechanism.

PROGRAM:

```
public class EX_12_A {  
    public static void main(String[] args) {  
        String path = System.getProperty("java.class.path");  
        System.out.println("Class Path: " + path);  
    }  
}
```

OUTPUT:



```
15e0d5b98c7f\redhat.java\jdt_ws\JAVA PROGRAMMING LAB_55021e85\bin' 'EX_12_A'  
Class Path: C:\Users\YASH\AppData\Roaming\Code\User\workspaceStorage\756018e722d6ff29c3eb15e0d5b98c7f\redhat.java\jdt_ws\JAVA  
PROGRAMMING LAB_55021e85\bin  
PS C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB>
```

c)AIM: Write a JAVA program that import and use the defined your package in the previous Problem.

PROGRAM:

```
package STR;
import java.util.*;
interface strings
{
    void reverseString();
}
class Str implements strings
{
    public void reverseString()
    {
        Scanner sc = new Scanner(System.in);
        StringBuilder sb = new StringBuilder();
        System.out.print("Enter String: ");
        sb.append(sc.nextLine());
        sb.reverse();
        System.out.println();
        System.out.print("Reversed String: "+sb);
    }
}
public class StrRev extends Str
{
    public static void main(String[] args)
    {
    }
}
import STR.StrRev;
public class string {
    public static void main(String[] args)
    {
        StrRev s = new StrRev();

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

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 12>javac StrRev.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 12>javac string.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 12>java string
Enter String: Namaste Beta
```

EXERCISE –13:(Applet)

a)**AIM:** Write a JAVA program to paint like paint brush in applet.

DESCRIPTION:

- We can perform painting operation in applet by the `mouseDragged()` method of `MouseMotionListener`.
- In the below example, `getX()` and `getY()` method of `MouseEvent` is used to get the current x-axis and y-axis. The `getGraphics()` method of `Component` class returns the object of `Graphics`.

PROGRAM:

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

/*<applet code="Paint" width=500 height=500></applet>*/
public class Paint extends Applet implements MouseMotionListener
{
    public void init()
    {
        addMouseMotionListener(this);
        setBackground(Color.red);
    }

    public void mouseDragged(MouseEvent e)
    {
        Graphics g = getGraphics();
        g.setColor(Color.blue);
        g.fillOval(e.getX(), e.getY(), 10, 10);
    }

    public void mouseMoved(MouseEvent me)
    {
    }
}
```

OUTPUT:

```
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 13>oldjavac Paint.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 13>appletviewer Paint.java
```




b)AIM: Write a JAVA program to display analogclock using Applet.

DESCRIPTION:

Analog clock can be created by using the Math class.

PROGRAM:

```
import java.applet.Applet;  
import java.awt.*;  
import java.util.*;  
/*<applet code="analogClock" width=500 height=500></applet>*/
```

```
public class analogClock extends Applet {
```

```
    public void init() {  
        this.setSize(new Dimension(800, 400));  
        setBackground(new Color(50, 50, 50));  
        new Thread() {  
            public void run() {  
                while (true) {  
                    repaint();  
                    delayAnimation();  
                }  
            }  
        }.start();  
    }
```

```
    private void delayAnimation() {  
        try {  
            Thread.sleep(1000);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
    }
```

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

```
        Calendar time = Calendar.getInstance();
```

```
        int hour = time.get(Calendar.HOUR_OF_DAY);  
        int minute = time.get(Calendar.MINUTE);  
        int second = time.get(Calendar.SECOND);
```

```
        if (hour > 12) {  
            hour -= 12;  
        }
```

```

g.setColor(Color.white);
g.fillOval(300, 100, 200, 200);
g.setColor(Color.black);
g.drawString("12", 390, 120);
g.drawString("9", 310, 200);
g.drawString("6", 400, 290);
g.drawString("3", 480, 200);

double angle;
int x, y;

angle = Math.toRadians((15 - second) * 6);

x = (int) (Math.cos(angle) * 100);
y = (int) (Math.sin(angle) * 100);

g.setColor(Color.red);
g.drawLine(400, 200, 400 + x, 200 - y);

angle = Math.toRadians((15 - minute) * 6);

x = (int) (Math.cos(angle) * 80);
y = (int) (Math.sin(angle) * 80);

g.setColor(Color.blue);
g.drawLine(400, 200, 400 + x, 200 - y);

angle = Math.toRadians((15 - (hour * 5)) * 6);

x = (int) (Math.cos(angle) * 50);
y = (int) (Math.sin(angle) * 50);

g.setColor(Color.black);
g.drawLine(400, 200, 400 + x, 200 - y);
}
}

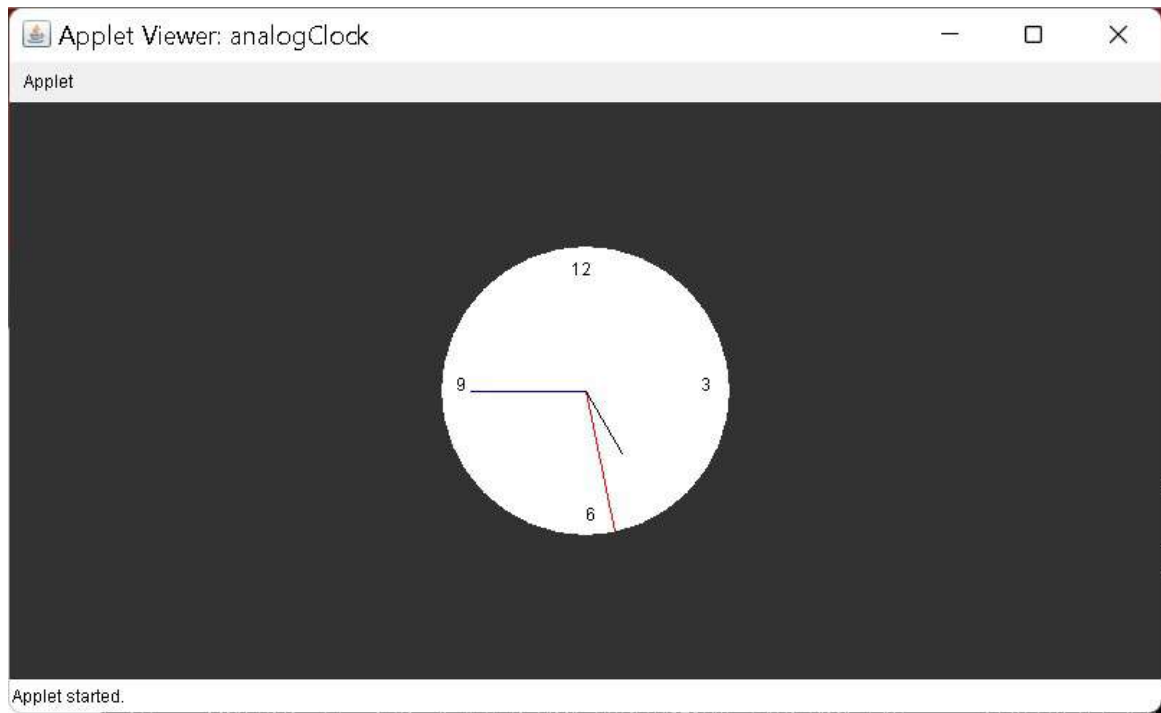
```

OUTPUT:

```

C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 13>oldjavac analogClock.java
C:\Users\YASH\Desktop\20vv1a1263\JAVA PROGRAMMING LAB\JAVA-LAB--main\EXERCISE 13>appletviewer analogClock.java

```



c)AIM: Write a JAVA program to create different shapes and fill colours using Applet.

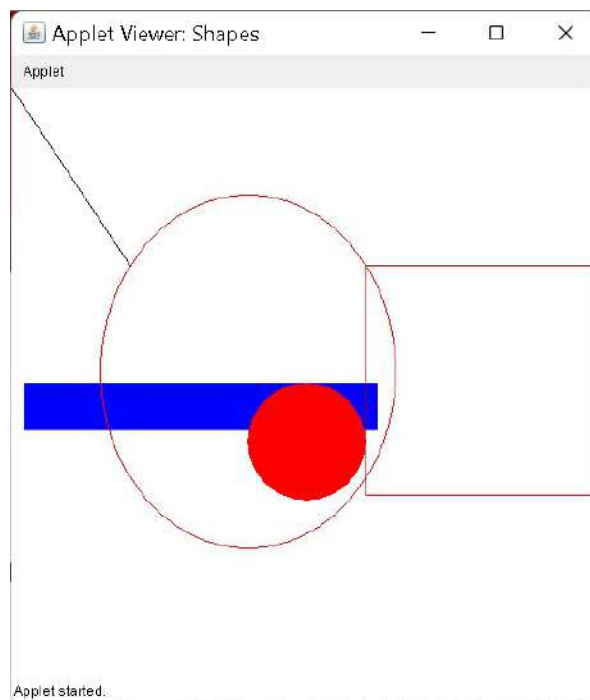
DESCRIPTION:

- Following example demonstrates how to create an applet which will have fill color in Various shapes using setColor(), fillRect() methods of Graphics class to fill color in a shape.
-

PROGRAM:

```
import java.applet.Applet;  
import java.awt.*;  
/*<applet code='Shapes' width=500 height=500> </applet> */  
public class Shapes extends Applet  
{  
    public void paint(Graphics g)  
    {  
        g.drawLine(0,0,100,150);  
        g.setColor(Color.blue);  
        g.fillRect(10,250,300,40);  
        g.setColor(Color.red);  
        g.drawOval(75,90,250,300);  
        g.fillOval(200,250,100,100);  
        g.draw3DRect(300, 150, 195, 195,false);  
    }  
}
```

OUTPUT:



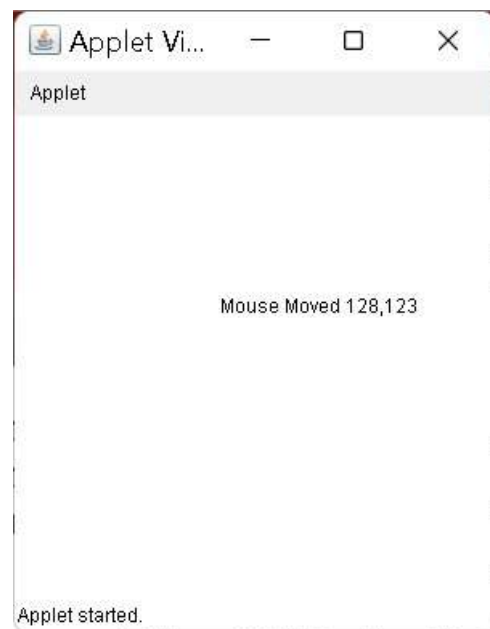
EXERCISE –14:(Event Handling)

a)**AIM:** Write a JAVA program that display the x and y positionof the cursor movement using Mouse.

PROGRAM:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
/*<applet code='cursorPosition' width=300 height=300></applet>*/
public class cursorPosition extends Applet implements MouseMotionListener
{
    int x,y;
    String msg="";
    public void init()
    {
        addMouseMotionListener(this);
    }
    public void paint(Graphics g)
    {
        g.drawString(msg,x,y);
    }
    public void mouseMoved(MouseEvent m)
    {
        x=m.getX();y=m.getY();
        msg = "Mouse Moved "+x+", "+y;
        repaint();
    }
    public void mouseDragged(MouseEvent m)
    {
        x = m.getX();
        y = m.getY();
        msg = "Mouse Dragged " + x + ", " + y;
        repaint();
    }
}
```

OUTPUT:



b)AIM: Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

DESCRIPTION:

- Java Swing is a GUI (graphical user Interface) widget toolkit for Java. Java Swing is a part of Oracle's Java foundation classes . Java Swing is an API for providing graphical user interface elements to Java Programs. Swing was created to provide more powerful and flexible components than Java AWT (Abstract Window Toolkit).

PROGRAM:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
/*<applet code='keyEvent' width=300 height=300></applet>*/
public class keyEvent extends Applet implements KeyListener
{
    int x=20,y=30;
    String msg = "";
    public void init()
    {
        addKeyListener(this);
        //requestFocus();
    }
    public void keyPressed(KeyEvent k)
    {
        showStatus("Key Pressed");
        msg = "Key Pressed";
        repaint();
    }
    public void keyReleased(KeyEvent k)
    {
        showStatus("Key Released");
        msg = "Key Released";
        repaint();
    }
    public void keyTyped(KeyEvent k)
    {
        showStatus("Key Typed");
        msg = "Key Typed";
        repaint();
    }
    public void paint(Graphics g)
    {

```

```
    g.drawString(msg,x,y);  
  }  
}
```

OUTPUT:

