1.Displaying default value of all primitive data types

AIM: To write a JAVA program to display default value of all primitive data type of JAVA

SOURCE-CODE:

```java
class defaultdemo
{
  static byte b;
  static short s;
    static int i;
  static long l;
  static float f;
static double d;
  static char c;
static boolean bl;
public static void main(String[] args)
{
System.out.println("The default values of primitive data types are:");
System.out.println("Byte :"+b);
System.out.println("Short :"+s);
System.out.println("Int :"+i);
System.out.println("Long :"+l);
System.out.println("Float :"+f);
System.out.println("Double :"+d);
System.out.println("Char :"+c);
System.out.println("Boolean :"+bl);
}
}
```

OUT-PUT:

The default values of primitive data types are:

Byte :0

Short :0

Int :0

Long :0

Float :0.0

Double :0.0

Char :

Boolean :false

2.Roots of a quadratic equation

AIM: To write a java program that display the roots of a quadratic equation ax2+bx=0.
Calculate the discriminate D and basing on value of D, describe the nature of root.

SOURCE-CODE:

```java
import java.util.*;
class quadraticdemo
{
public static void main(String[] args)
{
int a, b, c;
double r1, r2, D;
Scanner s = new Scanner(System.in);
System.out.println("Given quadratic equation:ax^2 + bx + c");
System.out.print("Enter a:");
a = s.nextInt();
System.out.print("Enter b:");
b = s.nextInt();
System.out.print("Enter c:");
c = s.nextInt();
D = b * b - 4 * a * c;
if(D > 0)
{
System.out.println("Roots are real and unequal");
r1 = ( - b + Math.sqrt(D))/(2*a);
r2 = (-b - Math.sqrt(D))/(2*a);
System.out.println("First root is:"+r1);
System.out.println("Second root is:"+r2);
}
else if(D == 0)
{
System.out.println("Roots are real and equal");
r1 = (-b+Math.sqrt(D))/(2*a);
System.out.println("Root:"+r1);
}
else
{
System.out.println("Roots are imaginary");
}
}
}
```

OUT-PUT:

Given quadratic equation:ax^2 + bx + c
Enter a:2
Enter b:3
Enter c:1
Roots are real and unequal
First root is:-0.5
Second root is:-1.0

3. Bike Race

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 racers.

SOURCE-CODE:

```java
import java.util.*;
class racedemo


public static void main(String[] args)
{
float s1,s2,s3,s4,s5,average;
Scanner s = new Scanner(System.in);
System.out.println("Enter speed of first racer:");
s1 = s.nextFloat();
System.out.println("Enter speed of second racer:");
s2 = s.nextFloat();
System.out.println("Enter speed of third racer:");
s3 = s.nextFloat();
System.out.println("Enter speed of fourth racer:");
s4 = s.nextFloat();
System.out.println("Enter speed of fifth racer:");
s5 = s.nextFloat();
average=(s1+s2+s3+s4+s5)/5;
if(s1>average)
System.out.println("First racer is qualify racer:");
else if(s2>average)
System.out.println("Second racer is qualify racer:");
else if(s3>average)
System.out.println("Third racer is qualify racer:");
else if(s4>average)
System.out.println("Fourth racer is qualify racer:");
```

```
else if(s5>average)
System.out.println("Fifth racer is qualify racer:");
}
}
```

OUT-PUT:

Enter speed of first racer:

4.5

Enter speed of second racer:

6.7

Enter speed of third racer:

3.8

Enter speed of fourth racer:

5.3

Enter speed of fifth racer:

4.9

Second racer is qualify racer:

4. Implementation of Binary search mechanism

AIM: To write a JAVA program to search for an element in a given list of elements using binary search mechanism

SOURCE-CODE:

```java
import java.util.Scanner;
class binarysearchdemo
{
public static void main(String args[])
{
int n, i, num,first, last, middle;int
a[ ]=new int[20];
Scanner s = new Scanner(System.in);
System.out.println("Enter total number of elements:");n =
s.nextInt();
System.out.println("Enter elements in sorted order:");
for (i = 0; i < n; i++)
a[i] = s.nextInt();
System.out.println("Enter the search value:");
num = s.nextInt();
first = 0; last
= n - 1;
middle = (first + last)/2;
while( first <= last )
{
if ( a[middle] < num )first
= middle + 1;
else if ( a[middle] == num )
{
System.out.println("number found");
break;
}
else
{
last = middle - 1;
}
middle = (first + last)/2;
}
if ( first > last )
System.out.println( " Number is not found");
}
}
```

OUT-PUT:

Enter total number of elements:5

Enter elements:

2 4 6 8 9

Enter the search value:8

number found

5. Bubble sort

AIM: To write a JAVA program to sort for an element in a given list of elements using bubble sort

SOURCE-CODE:

```java
import java.util.Scanner;
class bubbledemo
{
public static void main(String args[])
{
int n, i,j, temp;
int a[ ]=new int[20];
Scanner s = new Scanner(System.in);
System.out.println("Enter total number of elements:");n =
s.nextInt();
System.out.println("Enter elements:");
for (i = 0; i < n; i++)
a[i] = s.nextInt();
for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
System.out.println("The sorted elements are:");
for(i=0;i<n;i++)
System.out.print("\t"+a[i]);
}
}
```

OUT-PUT:

Enter total number of elements:

10
Enter elements:
3 2 5 7 6 8 9 1 4 0
The sorted elements are:
0 1 2 3 4 5 6 7 8 9

6. Merge sort:

AIM: To write a JAVA program to sort for an element in a given list of elements usingmerge sort

SOURCE-CODE:

```
import java.util.*;
class mergedemo
{
public static void main(String args[])
{
int n1,n2,i,j,k;
int a[ ]=new int[20];int
b[ ]=new int[20];int c[
]=new int[20];
Scanner s = new Scanner(System.in); System.out.println("Enter
number of elements in first array:");n1 = s.nextInt();
System.out.println("Enter sorted elements of first array:");for (i
= 0; i < n1; i++)
a[i] = s.nextInt();
System.out.println("Enter number of elements in second array:");n2 =
s.nextInt();
System.out.println("Enter sorted elements of second array:");for
(j = 0; j < n2; j++)
b[j] = s.nextInt();i =
0;
j = 0;
k = 0;
while((i < n1) && (j <n2))
{
if(a[i] > b[j])
c[k++] = b[j++];
else
c[k++] = a[i++];
}
while(i < n1)
c[k++] = a[i++];
while(j < n2)
c[k++] = b[j++];
```

```
System.out.println("After merging the elements are:\n");
for(i = 0; i < (n1 + n2); i++)
System.out.print("\t"+c[i]);
}
}
```

OUT-PUT:
Enter number of elements in first array:
6
Enter elements of first array:
8 9 12 13 15 18
Enter number of elements in second array:
5
Enter elements of second array:
6 7 10 11 20
After merging the elements are:
6   7 8 9 10 11 12 13 15 18 20

7. Implementing StringBuffer
AIM: To write a JAVA program using StringBuffer to delete, remove character
SOURCE-CODE:
```
class stringbufferdemo
{
public static void main(String[] args)
{
StringBuffer sb1 = new StringBuffer("Hello World");
sb1.delete(0,6);
System.out.println(sb1);
StringBuffer sb2 = new StringBuffer("Some Content");
System.out.println(sb2);
sb2.delete(0, sb2.length());
System.out.println(sb2);
StringBuffer sb3 = new StringBuffer("Hello World");
sb3.deleteCharAt(0);
System.out.println(sb3);
}
}
```

OUT-PUT:
World
Some Content
Hello World

8. Implementing Class & Objects

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

SOURCE-CODE:

1. no return type and without parameter-list:

```
class A
{
int l=10,b=20;
void display()
{
System.out.println(l);
System.out.println(b);
}
}
class methoddemo
{
public static void main(String args[])
{
A a1=new A();
a1.display();
}
}
```

OUT-PUT:

```
10
20
```

2. no return type and with parameter-list:

```
class A
{
void display(int l,int b)
{
System.out.println(l);
System.out.println(b);
}
}
class methoddemo
{
public static void main(String args[])
{
A a1=new A();
a1.display(10,20);
}
}
```

OUT-PUT:

10

20

3. return type and without parameter-list

```
class A
{
int l=10,b=20;

int area()
{
return l*b;
}
}
class methoddemo
{
public static void main(String args[])
{
A a1=new A();
int r=a1.area();
System.out.println("The area is: "+r);
}
}
```

OUT-PUT:

The area is:200

4. return type and with parameter-list:

```
class A
{
int area(int l,int b)
{
return l*b;
}
}
class methoddemo
{
public static void main(String args[])
{
A a1=new A();
int r=a1.area(10,20);
System.out.println("The area is:"+r);
}
}
```

OUT-PUT:

The area is:200

9. Implementing Constructor

AIM: To write a JAVAto implement constructor

SOURCE-CODEs:

(i) A constructor with no parameters:

```
class A
{
int l,b;
A()
{ l=10;
b=20;
}
int area()
{
return l*b;
}
}
class constructordemo
{
public static void main(String args[])
{
A a1=new A();
int r=a1.area();
System.out.println("The area is: "+r);
}
}
```

OUT-PUT:

The area is:200

(ii) A constructor with parameters

```
class A
{
int l,b;
A(int u,int v)
{
l=u;
b=v;
}
int area()
{
return l*b;
}
}
class constructordemo
```

11

```
{
public static void main(String args[])
{
A a1=new A(10,20);
int r=a1.area();
System.out.println("The area is: "+r);
}
}
```

OUT-PUT:

The area is:200

10.Constructor Overloading

AIM: To write a JAVA program to implement constructor overloading

SOURCE-CODE:

```
class A
{
int l,b;
A()
{ l=10;b=20;
}
A(int u,int v)
{
l=u;b=v;
}
int area()
{
return l*b;
}
}
class overconstructdemo
{
public static void main(String args[])
{
A a1=new A(); int
r1=a1.area();
System.out.println("The area is: "+r1);A a2=new
A(30,40);
int r2=a2.area(); System.out.println("The area is:
"+r2);
}
}
```

OUT-PUT:

The area is: 200 The area is:
1200

11. Method Overloading

AIM: To write a JAVA program implement method overloading

SOURCE-CODE:

```java
class A
{
int l=10,b=20;int area()
{
return l*b;
}
int area(int l,int b)
{
return l*b;
}
}
class overmethoddemo
{
public static void main(String args[])
{
A a1=new A(); int
r1=a1.area();
System.out.println("The area is: "+r1);int
r2=a1.area(5,20); System.out.println("The area is:
"+r2);
}
}
```

OUT-PUT:

The area is: 200The area is:
100

12.Implementing Single Inheritance

AIM: To write a JAVA program to implement Single Inheritance

SOURCE-CODE:

```java
class A
{ A()
{
System.out.println("Inside A's Constructor");
}
```

```
}
class B extends A
{ B()
{
System.out.println("Inside B's Constructor");
}
}
class singledemo
{
public static void main(String args[])
{
B b1=new B();
}
}
```

OUT-PUT:

Inside A's ConstructorInside B's
Constructor

13. Multi level Inheritance

AIM: To write a JAVA program to implement multi level Inheritance

SOURCE-CODE:

```
class A
{ A()
{
System.out.println("Inside A's Constructor");
}
}
class B extends A
{ B()
{
System.out.println("Inside B's Constructor");
}
}
class C extends B
{ C()
{
System.out.println("Inside C's Constructor");
}
}
class multidemo
{
public static void main(String args[])
{
```

```
C c1=new C();
}
}
```

OUT-PUT:

Inside A's Constructor Inside B's
ConstructorInside C's Constructor

14.   Abstract Class

AIM: To write a java program for abstract class to find areas of different shapes

SOURCE-CODE:

```
abstract class shape
{
abstract double area();
}
class rectangle extends shape
{
double l=12.5,b=2.5;double
area()
{
return l*b;
}
}
class triangle extends shape
{
double b=4.2,h=6.5;double
area()
{
return 0.5*b*h;
}
}
class square extends shape
{
double s=6.5;double
area()
{
return 4*s;
}
}
class shapedemo
{
public static void main(String[] args)
{
rectangle r1=new rectangle();triangle
```

```
t1=new triangle(); square s1=new
square();
System.out.println("The area of rectangle is: "+r1.area());System.out.println("The area of
triangle is: "+t1.area()); System.out.println("The area of square is: "+s1.area());
}
}
```

OUT-PUT:
The area of rectangle is: 31.25The area of
triangle is: 13.65 The area of square is:
26.0

15.super keyword implementation

AIM: Write a JAVA program give example for "super" keyword

SOURCE-CODEs:

(i)Using super to call super class constructor (Without parameters)

```
class A
{
int l,b;
A()
{ l=10;b=20;
}
}
class B extends A
{
int h;
B()
{
super();h=30;
}
int volume()
{
return l*b*h;
}
}
class superdemo
{
public static void main(String args[])
{
B b1=new B();
int r=b1.volume(); System.out.println("The vol.
is: "+r);
}
```

```
}
```
OUT-PUT:

The vol. is:6000

(ii) Using super to call super class constructor (With parameters)
```
class A
{
int l,b;
A(int u,int v)
{
l=u;b=v;
}
}
class B extends A
{
int h;
B(int u,int v,int w)
{
super(u,v);h=w;
}
int volume()
{
return l*b*h;
}
}
class superdemo
{
public static void main(String args[])
{
B b1=new B(30,20,30);
int r=b1.volume(); System.out.println("The vol.
is: "+r);
}
}
```
OUT-PUT:

The vol. is:18000

16.   Implementing interface
AIM: To write a JAVA program to implement Interface.

SOURCE-CODEs:

(i) First form of interface implementation
```
interface A
{
void display();
```

```
}
class B implements A
{
public void display()
{
System.out.println("B's method");
}
}
class C extends B
{
public void callme()
{
System.out.println("C's method");
}
}
class interfacedemo
{
public static void main(String args[])
{
C c1=new C();c1.display();
c1.callme();
}
}
```

OUT-PUT:

B's methodC's method

(ii) Second form of interface implementation

```
interface D
{
void display();
}
interface E extends D
{
void show();
}
class A
{
void callme()
{
System.out.println("This is in callme method");
}
}

class B extends A implements E
```

```
{
public void display()
{
System.out.println("This is in display method");
}
public void show()
{
System.out.println("This is in show method");
}
}
class C extends B
{
void call()
{
System.out.println("This is in call method");
}
}
class interfacedemo
{
public static void main(String args[])
{
C c1=new C();c1.display();
c1.show();
c1.callme();
c1.call();
}
}
```

OUT-PUT:

This is in display methodThis is in
show method This is in callme
methodThis is in call method

(iii) Third form of interface implementation

```
interface A
{
void display();
}
class B implements A
{
public void display()
{
System.out.println("This is in B's method");
}
```

```
}
class C implements A
{
public void display()
{
System.out.println("This is C's method");
}
}
class interfacedemo
{

public static void main(String args[])
{
B  b1=new  B(); C  c1=new
C();b1.display();
c1.display();
}
}
```

OUT-PUT:

This is in B's methodThis is C's
method

(iv) Fourth form of interface implementation

```
interface A
{
void display();
}
interface B
{
void callme();
}
interface C extends A,B
{
void call();
}
class D implements C
{
public void display()
{
System.out.println("interface A");
}
public void callme()
{
System.out.println("interface B");
```

```
}
public void call()
{
System.out.println("interface C");
}
}
class interfacedemo
{
public static void main(String args[])
{
D d1=new D();d1.display();
d1.callme();
d1.call();
}
}
```

OUT-PUT:
interface A interface B
interface C

17.Exception handling mechanism
AIM: To write a JAVA program that describes exception handling mechanism
SOURCE-CODE:
Usage of Exception Handling:

```
class trydemo
{
public static void main(String args[])
{
try
{
int a=10,b=0;int c=a/b;
System.out.println(c);
}
catch(ArithmeticException e)
{
System.out.println(e);
}
System.out.println("After the catch statement");
}
}
```

OUT-PUT:

java.lang.ArithmeticException: / by zeroAfter the catch
statement

18. **Illustrating multiple catch** clauses

AIM: To write a JAVA program Illustrating Multiple catch clauses

SOURCE-CODE:

```
class multitrydemo
{
public static void main(String args[])
{
try
{
int a=10,b=5;int c=a/b;
int d[]={0,1}; System.out.println(d[10]);
System.out.println(c);
}
catch(ArithmeticException e)
{
System.out.println(e);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println(e);
}
System.out.println("After the catch statement");
}
}
```

OUT-PUT:

java.lang.ArrayIndexOutOfBoundsException: 10 After the catch
statement

19.RuntimePolymorphism

AIM: To write a JAVA program that implements Runtime polymorphism

SOURCE-CODE:

```
class A
{
void display()
{
System.out.println("Inside A class");
```

```
}
}
class B extends A
{
void display()
{
System.out.println("Inside B class");
}
}
class C extends A
{
void display()
{
System.out.println("Inside C class");
}
}
class runtimedemo
{
public static void main(String args[])
{
A a1=new A();
B b1=new B();
C c1=new C();
A ref;
ref=c1;
ref.display();
ref=b1;
ref.display();
ref=a1;
ref.display();
}
}
```

OUT-PUT:

Inside C class

Inside B class

Inside A class

20. Case study on Runtime Polymorphism

AIM: To write a Case study on run time polymorphism, inheritance that implements in above problem

Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

☐ When an overridden method is called through a superclass reference, Java determines which version(superclass/subclasses) of that method is to be executed based upon the type of the object being referred to at the time the call occurs. Thus, this determination is made at run time.

☐ At run-time, it depends on the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed

☐ A superclass reference variable can refer to a subclass object. This is also known as upcasting. Java uses this fact to resolve calls to overridden methods at run time.

Upcasting

SuperClass obj=new SubClass

```
┌─────────────────┐
│   SuperClass    │
└─────────────────┘
         ▲
         │
     extends
         │
┌─────────────────┐
│    SubClass     │
└─────────────────┘
```
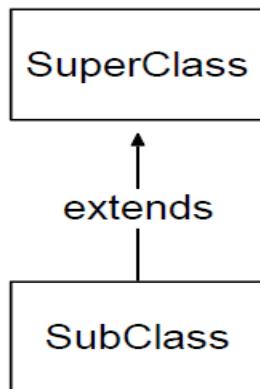
Therefore, if a superclass contains a method that is overridden by a subclass, then when different types of objects are referred to through a superclass reference variable, different versions of the method are executed. Here is an example that illustrates dynamic method dispatch:

The example is given by

Consider a scenario, Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest

```
                    ┌──────────────────────────┐
                    │           Bank           │
                    ├──────────────────────────┤
                    │                          │
                    ├──────────────────────────┤
                    │ getRateOfInterest():float│
                    └──────────────────────────┘
                                 ▲
                                 │
                             extends
         ┌───────────────────────┼───────────────────────┐
┌────────────────────┐ ┌────────────────────┐ ┌────────────────────┐
│        SBI         │ │       ICICI        │ │        AXIS        │
├────────────────────┤ ├────────────────────┤ ├────────────────────┤
│                    │ │                    │ │                    │
├────────────────────┤ ├────────────────────┤ ├────────────────────┤
│getRateOfInterest():│ │getRateOfInterest():│ │getRateOfInterest():│
│       float        │ │       float        │ │       float        │
└────────────────────┘ └────────────────────┘ └────────────────────┘
```
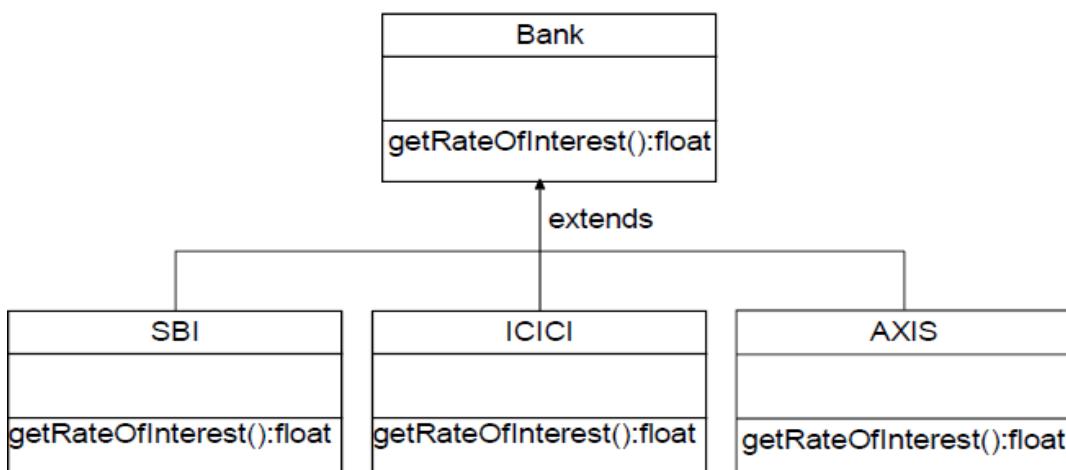
21.creation of illustrating throw

AIM: To write a JAVA program for creation of Illustrating throw

SOURCE-CODE:

```
class throwdemo
{
public static void main(String args[])
{
try
{
throw new NullPointerException("demo");
}
catch(NullPointerException e)
{
System.out.println(e);
}
}
}
```

OUT-PUT:

java.lang.NullPointerException: demo

22.creation of illustrating finally

AIM: To write a JAVA program for creation of Illustrating finally

SOURCE-CODE(i):

```
class finallydemo
{
public static void main(String args[])
{
try
{
int a=10,b=0;int
c=a/b;
System.out.println(c);
}
catch(ArithmeticException e)
{
System.out.println(e);
}
finally
{
```

```
System.out.println("This is inside finally block");
}
}
}
```

OUT-PUT:

java.lang.ArithmeticException: / by zeroThis is
inside finally block

SOURCE-CODE(ii):

```
class finallydemo
{
public static void main(String args[])
{
try
{
int a=10,b=5;int
c=a/b;
System.out.println(c);
}
catch(ArithmeticException e)
{
System.out.println(e);
}
finally
{
System.out.println("This is inside finally block");
}
}
}
```

OUT-PUT:

2

This is inside finally block

23.creation of Java Built-in-Exceptions

AIM: To write a JAVA program for creation of Java Built-in Exceptions

SOURCE-CODEs:

(i) Arithmetic exception

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

```
{
try
{
int a = 10, b = 0;int c =
a/b;
System.out.println (c);
}
catch(ArithmeticException e)
{
System.out.println (e);
}
}
}
```

OUT-PUT:

java.lang.ArithmeticException: / by zero

(ii)NullPointer Exception

```
class nullpointerdemo



{
public static void main(String args[])
{
try
{
String a = null; System.out.println(a.charAt(0));
}
catch(NullPointerException e)
{
System.out.println(e);
}
}
}
```

OUT-PUT:

java.lang.NullPointerException

(iii)StringIndexOutOfBound Exception

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

```
{
String a = "This is like chipping ";char c
= a.charAt(24); System.out.println(c);
}
catch(StringIndexOutOfBoundsException e)
{
System.out.println(e);
}
}
}
```

OUT-PUT:
java.lang.StringIndexOutOfBoundsException: String
index out of range: 24

(iv)FileNotFound Exception
```
import java.io.*;
class filenotfounddemo
{
public static void main(String args[])
{
try
{
File file = new File("E://file.txt");
FileReader fr = new FileReader(file);
}
```

```
catch (FileNotFoundException e)
{
System.out.println(e);
}
}
}
```
OUT-PUT:
java.io.FileNotFoundException: E:\file.txt (The system cannot find the file specified)
(v)NumberFormat Exception
```
class numberformatdemo
{
public static void main(String args[])
{
```

```
try
{
int num = Integer.parseInt ("akki") ;
System.out.println(num);
}
catch(NumberFormatException e)
{
System.out.println(e);
}
}
}
```

OUT-PUT:

java.lang.NumberFormatException: For input string: "akki"

(vi)ArrayIndexOutOfBounds Exception

```
class arraybounddemo
{
public static void main(String args[])
{
try
{
int a[] = new int[5];a[6] =
9;
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println (e);
}
}
}
```

OUT-PUT:

java.lang.ArrayIndexOutOfBoundsException: 6

24.creation of User Defined Exception

AIM: To write a JAVA program for creation of User Defined Exception

SOURCE-CODE:

```
class A extends Exception
{
A(String s1)
{
super(s1);
}
}
```

```
class owndemo
{
public static void main(String args[])
{
try
{
throw new A("demo ");
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

OUT-PUT:
A: demo

25.Extending Thread class

AIM: To 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)

SOURCE-CODEs:

(i) Creating multiple threads using Thread class

```
class A extends Thread
{
public void run()
{
try
{
for(int i=1;i<=10;i++)
{
sleep(1000); System.out.println("good
morning");
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

```
class B extends Thread
{
public void run()
{
try
{
for(int j=1;j<=10;j++)
{
sleep(2000); System.out.println("hello");
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
class C extends Thread
{
public void run()
{
try
{
for(int k=1;k<=10;k++)
{
sleep(3000);
System.out.println("welcome");
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
class threaddemo
{
public static void main(String args[])
{
A a1=new A();
B b1=new B();
Cc1=newC();
a1.start();
```

```
b1.start();
c1.start();
}
}
```

OUT-PUT:

good morninghello

good morninggood

morningwelcome

hello

good   morning

good   morning

hello

good morning

welcome good

morninghello

good morninggood

morningwelcome

hello

good morninghello

welcome hello

welcome

hello

hello

welcomehello

welcome

welcome

welcome

welcome

(ii) Creating multiple threads using Runnable interface

```
class A implements Runnable
{
public void run()
{
try
{
for(int i=1;i<=10;i++)
{
Thread.sleep(1000);
System.out.println("good morning");
}
}
catch(Exception e)
```

```
{
System.out.println(e);
}
}
}
class B implements Runnable
{
public void run()
{
try
{
for(int j=1;j<=10;j++)
{
Thread.sleep(2000);
System.out.println("hello");
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
class C implements Runnable
{
public void run()
{
try
{
for(int k=1;k<=10;k++)
{
Thread.sleep(3000);
System.out.println("welcome");
}
}
catch(Exception e)
{
System.out.println(e);
}

}
}
class runnabledemo
```

```
{
public static void main(String args[])
{
A   a1=new   A();  B
b1=new     B();    C
c1=new C();
Thread   t1=new    Thread(a1);
Thread   t2=new    Thread(b1);
Thread   t3=new    Thread(c1);
t1.start();
t2.start();
t3.start();
}
}
```

OUT-PUT:

good   morning
good   morning
hello
good morning
welcome good
morninghello
good morninggood
morningwelcome
hello
goodmorning
goodmorning
hello
good morning
welcome
good morning
hello
welcomehello
hello
welcomehello
welcomehello
hello
welcome
welcome
welcome
welcome

26.Implementing isAlive() and join()

AIM: To write a program illustrating isAlive and join ()

SOURCE-CODE:

```java
class A extends Thread
{
public void run()
{
try
{
for(int i=1;i<=10;i++)
{
sleep(1000); System.out.println("good
morning");
}
}
catch(Exception e)
{
System.out.println(e);
}
}
}
class B extends Thread
{
public void run()
{
try
{
for(int j=1;j<=10;j++)
{
sleep(2000); System.out.println("hello");
}
}
catch(Exception e)
{


System.out.println(e);
}
}
}
class C extends Thread
{
public void run()
```

```
{
try
{
for(int k=1;k<=10;k++)
{
sleep(3000);
System.out.println("welcome");
}

}
catch(Exception e)
{
System.out.println(e);
}
}
}
class isalivedemo
{
public static void main(String args[])
{
A   a1=new   A();  B
b1=new     B();    C
c1=new          C();
a1.start();
b1.start();
c1.start();
System.out.println(a1.isAlive());
System.out.println(b1.isAlive());
System.out.println(c1.isAlive());try
{
a1.join();
b1.join();
c1.join();
}
catch(InterruptedException e)
{
System.out.println(e);
}
System.out.println(a1.isAlive());
System.out.println(b1.isAlive());
System.out.println(c1.isAlive());
```

```
}
}
```

OUT-PUT:

true	good morning
true	hello
true	welcome
good morning	hello
good morning	hello
hello	welcome
good morning	hello
welcome	welcome
good morning	hello
hello	hello
good morning	welcome
good morning	welcome
welcome	welcome
hello	welcome
good morning	false
good morning	false
hello	false
good morning	
welcome	

27. Implementation of Daemon Threads

AIM: To write a program illustrating Daemon Threads

SOURCE-CODE:

```
class A extends Thread
{
public void run()
{
if(Thread.currentThread().isDaemon())
System.out.println("daemon thread work");else
System.out.println("user thread work");
}
}
class daemondemo
{
public static void main(String[] args)
{
A a1=new A();
```

```
A a2=new A();
A a3=new A();
a1.setDaemon(true);
a1.start();
a2.start();
a3.start();
}
}
```

OUT-PUT:

daemon thread work
user thread work
user thread work

28.Producer-Consumer problem

AIM: Write a JAVA program Producer Consumer Problem

SOURCE-CODE:

```
class A
{
int n;
boolean b=false;
synchronized int get()
{
if(!b)
try
{
wait();
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println("Got:"+n);
b=false;
notify();
return n;
}
synchronized void put(int n)
{
if(b)
try
{
wait();
```

```
}
catch(Exception e)
{
System.out.println(e);
}
this.n=n;
b=true;
System.out.println("Put:"+n);
notify();
}
}
class producer implements Runnable
{
A a1;
Thread t1;
producer(A a1)
{
this.a1=a1;
t1=new Thread(this);
t1.start();
}
public void run()
{
for(int i=1;i<=10;i++)
{
a1.put(i);
}
}
}
class consumer implements Runnable
{
A a1;
Thread t1;
consumer(A a1)
{
this.a1=a1;
t1=new Thread(this);
t1.start();
}
public void run()
{
for(int j=1;j<=10;j++)
```

```
{
a1.get();
}
}
}
class interdemo
{
public static void main(String args[])
{
A a1=new A();
producer p1=new producer(a1);
consumer c1=new consumer(a1);
}
}
```

OUT-PUT:
Put:1
Got:1
Put:2
Got:2
Put:3
Got:3
Put:4
Got:4
Put:5

Got:5
Put:6
Got:6
Put:7
Got:7
Put:8
Got:8
Put:9
Got:9
Put:10
Got:10

29.Case study on thread synchronization

AIM: To write a case study on thread Synchronization after solving the above producer consumer problem

A case study on thread synchronization after solving producer consumer problem:

☐ We can use wait, notify and notifyAll methods to communicate between threads in Java.

☐

For example, if we have two threads running in your SOURCE-CODE e.g.Producer and Consumer then producer thread can communicate to the consumer that it can start consuming now because there are items to consume in the queue.

☐ Similarly, a consumer thread can tell the producer that it can also start putting items now because there is some space in the queue, which is created as a result of consumption.

☐ A thread can use wait() method to pause and do nothing depending upon some condition.

☐ For example, in the producer-consumer problem, producer thread should wait if the queue is full and consumer thread should wait if the queue is empty.

☐ If some thread is waiting for some condition to become true, we can use notify and notifyAll methods to inform them that condition is now changed and they can wake up.

☐ Both notify() and notifyAll() method sends a notification but notify sends the notification to only one of the waiting thread, no guarantee which thread will receive notification and notifyAll() sends the notification to all threads.

Things to remember:

1. We can use wait() and notify() method to implement inter-thread communication in Java. Not just one or two threads but multiple threads can communicate to each other by using these methods.

2. Always call wait(), notify() and notifyAll() methods from synchronized method or synchronized block otherwise JVM will throw IllegalMonitorStateException.

3. Always call wait and notify method from a loop and never from if() block, because loop test waiting condition before and after sleeping and handles notification even if waiting for the condition is not changed.

4. Always call wait in shared object e.g. shared queue in this example.

5. Prefer notifyAll() over notify() method due to reasons given in this article

30.Illustration of class path

AIM: To write a JAVA program, illustrate class path

```
import java.net.URL;
import java.net.URLClassLoader;
public class App
{
public static void main(String[] args)
{
ClassLoader sysClassLoader = ClassLoader.getSystemClassLoader();URL[]
urls = ((URLClassLoader)sysClassLoader).getURLs();
for(int i=0; i< urls.length; i++)
{
System.out.println(urls[i].getFile());
```

```
}
}
}
```
OUT-PUT:

E:/java%20work/

31.A case study on including in class path in os environment

AIM: To write a case study on including in class path in your os environment of your package.

☐ The differences between path and classpath are given by.

1. The PATH is an environment variable used to locate "java" or "javac" command, to run java SOURCE-CODE and compile java source file. The CLASSPATH is an environment variable used to set path for java classes.

2. In order to set PATH in Java, we need to include **bin** directory in PATH environment while in order to set CLASSPATH we need to include all directories where we have put either our .class file or JAR file, which is required by our Java application.

3. PATH environment variable is used by operating system while CLASSPATH is used by Java ClassLoaders to load class files.

4. Path refers to the system while classpath refers to the Developing Environment.

→By default the java run time system uses the current working directory.

→Normally to execute a java SOURCE-CODE in any directory we have to set the path byas follows set path= c:\SOURCE-CODE Files\java\jdk1.5.0_10\bin;

Setting environmental variable in windows xp:

Step-1:

→Select My computer on the desktop and right click the mouse and then select properties

→It displays the following "System Properties" dialog.
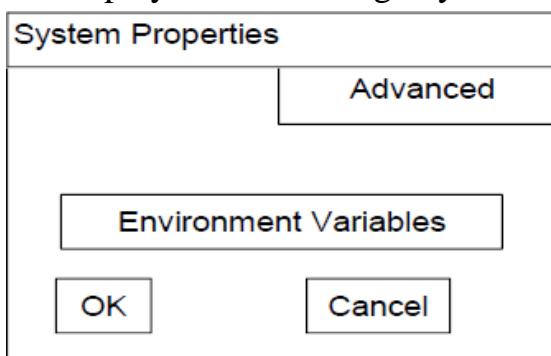
Step-2:

→In System Properties click Advanced and then click Environment Variables

→It displays the following "Environment Variables" dialog.

```
Environment Variables

    System Variables
              New

    OK              Cancel
```
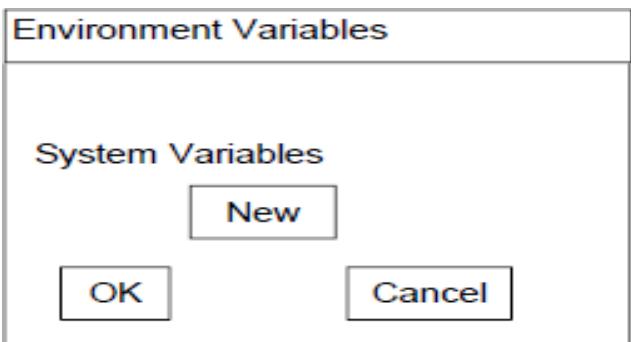
Step-3:

→In Environment Variables click New in System variables

→It displays the following "New System Variable" dialog box.

```
New System Variable

    variable name: [          ]
    variable value: [          ]

        OK          Cancel
```
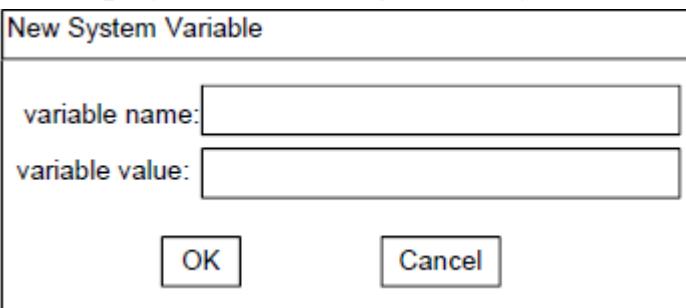
Step-4:

→Now type variable name as a path and then variable value as c:\SOURCE-CODE Files\java\jdk1.5.0_10\bin;

```
New System Variable

    variable name:  path
    variable value: c:\Program Files\java\jdk1.5.0_10\bin;

        OK          Cancel
```
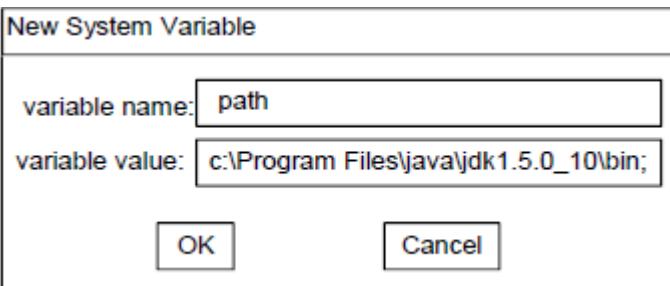
Step-5:

→Click OK

32.Creating and importing a package

AIM: To write a JAVA SOURCE-CODE that import and use the defined your package in the previousProblem

(i) Creating a package:

Steps:

1. First declare the name of the package using package keyword

Example: package mypack;

2. Type the following SOURCE-CODE under this package statement. In package : class

,data, methods all are **public**

```
package mypack;



public class box
{
public int l=10,b=20;
public void display()
{
System.out.println(l);
System.out.println(b);
}
}
```

3. Create sub directory with a name same that of package name under the current working directory by as follows. d:\>md mypack
4. Under this subdirectory store the above SOURCE-CODE with a file name "box.java".

(ii) importing a package:
Steps:
1. packages can be accessed by using the import statement

General form: import pack1[.pack2].(classname/*);Example:
import java.io.*;
Here pack1 is name of top level package and pack2 is name of sub package
2. Type the following SOURCE-CODE under the current working directory and save the SOURCE-CODE with a file name "example.java".

```
import mypack.box;
class packagedemo
{
public static void main(String args[])
{
box b1=new box();
b1.display();
}
}
```

3. Now compile the above SOURCE-CODE in the current working directory d:\

javac packagedemo.java
4. Execute the above SOURCE-CODE in current working directory

java packagedemo

OUT-PUT:

10

20

33.**Paint like Paint Brush in Applet**

AIM: To write a JAVA program to paint like paint brush in applet.

SOURCE-CODE:

```java
import java.applet.*; import
java.awt.*; import
java.awt.event.*;
//<applet code="paintdemo" width="800" height="500"></applet>
public class paintdemo extends Applet implements MouseMotionListener
{
int w, h; Image i;
Graphics g1;
public void init()
{
w = getSize().width; h = getSize().height;i =
createImage( w, h );
g1 = i.getGraphics();
g1.setColor( Color.white ); g1.fillRect( 0, 0, w, h ); g1.setColor( Color.red );i =
createImage( w, h );
g1 = i.getGraphics();
g1.setColor( Color.white ); g1.fillRect( 0, 0, w, h ); g1.setColor( Color.blue );
addMouseMotionListener( this );
}
public void mouseMoved( MouseEvent e ) { }public void
mouseDragged( MouseEvent me )
{
int x = me.getX(); int y = me.getY();
g1.fillOval(x-10,y-10,20,20); repaint();
me.consume();
}
public void update( Graphics g )
{
g.drawImage( i, 0, 0, this );
}
public void paint( Graphics g )
{
update(g);
}
```

}
OUT-PUT:
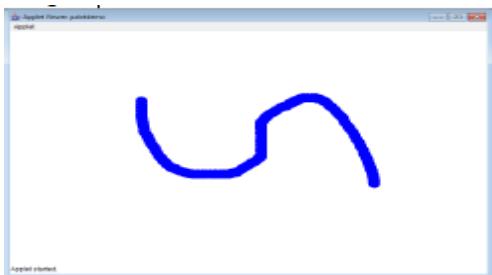
34.Display Analog Clock using Applet

AIM: To write a JAVA program to display analog clock using Applet.

SOURCE-CODE:

```
import java.util.*; import
java.text.*; import
java.applet.*;import
java.awt.*;
//<applet code="clockdemo" width="550" height="250"></appletpublic
class clockdemo extends Applet implements Runnable
{
int h=0, m=0, s=0;
String str=""; int wt, ht; Thread thr=null; boolean b;public
void init()
{
wt=getSize().width; ht=getSize().height;
}
public void start()
{
if (thr==null)
{
thr=new Thread(this);
b=false;
thr.start();
}
else
{
if(b)
{
b=false;
```

```
synchronized(this)
{
notify();
}
}
}
}
public void stop()
{
b=true;
}
public void run()
{
try
{
while(true)
{
Calendar clndr=Calendar.getInstance();
h=clndr.get(Calendar.HOUR_OF_DAY);
if(h>12)h-=12;
m=clndr.get(Calendar.MINUTE); s=clndr.get(Calendar.SECOND);
SimpleDateFormat frmatter=new SimpleDateFormat("hh:mm:ss",
Locale.getDefault());
Date d=clndr.getTime(); str=frmatter.format(d);if(b)
{
synchronized (this)
{
while(b)
{
wait();
}
}
}
repaint();
thr.sleep(1000);
}
}
catch(Exception e)
{
System.out.println(e);
}
}
```

```
void drawHand(double angle, int radius, Graphics grp)
{
angle-=0.5*Math.PI;
int a=(int)(radius*Math.cos(angle)); int b=(int)(radius*Math.sin(angle));
grp.drawLine(wt/2,ht/2,wt/2+a,ht/2+b);
}
void drawWedge(double angle,int radius, Graphics grp)
{
angle-=0.5*Math.PI;
int a=(int)(radius*Math.cos(angle)); int b=(int)(radius*Math.sin(angle));
angle+=2*Math.PI/3;
int a2=(int)(5*Math.cos(angle)); int b2=(int)(5*Math.sin(angle));
angle+=2*Math.PI/3;
int a3=(int)(5*Math.cos(angle)); int b3=(int)(5*Math.sin(angle));
grp.drawLine(wt/2+a2, ht/2+b2,wt/2+a,ht/2+b); grp.drawLine(wt/2+a3,
ht/2+b3,wt/2+a,ht/2+b); grp.drawLine(wt/2+a2, ht/2+b2,wt/2+a3,ht/2+b3);
}
public void paint(Graphics grp)
{
grp.setColor(Color.gray);
drawWedge(2*Math.PI*h/12,wt/5,grp); drawWedge(2*Math.PI*m/60,wt/3,grp);
drawHand(2*Math.PI*s/60,wt/2,grp);
}
}
```

OUT-PUT:
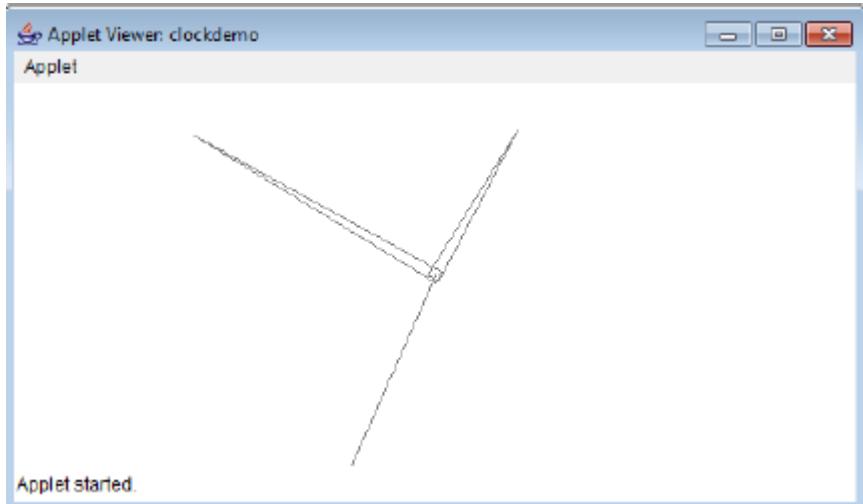
35.Display Analog Clock using Applet

AIM: To write a JAVA program to create different shapes and fill colors using Applet

SOURCE-CODE:

```
import java.awt.*; import
java.applet.*;
//<applet code="graphicsdemo" width="400" height="400"></applet>public
class graphicsdemo extends Applet
{
public void paint(Graphics g)
{
int x[]={10,220,220};
int y[]={400,400,520};
int n=3; g.drawLine(10,30,200,30);
g.setColor(Color.blue); g.drawRect(10,40,200,30); g.setColor(Color.red);
g.fillRect(10,80,200,30); g.setColor(Color.orange);
g.drawRoundRect(10,120,200,30,20,20); g.setColor(Color.green);
g.fillRoundRect(10,160,200,30,20,20); g.setColor(Color.blue);
g.drawOval(10,200,200,30); g.setColor(Color.black);
g.fillOval(10,240,40,40); g.setColor(Color.yellow);
g.drawArc(10,290,200,30,0,180); g.setColor(Color.yellow);
g.fillArc(10,330,200,30,0,180); g.setColor(Color.pink); g.fillPolygon(x,y,n);
}
}
```

OUT-PUT:
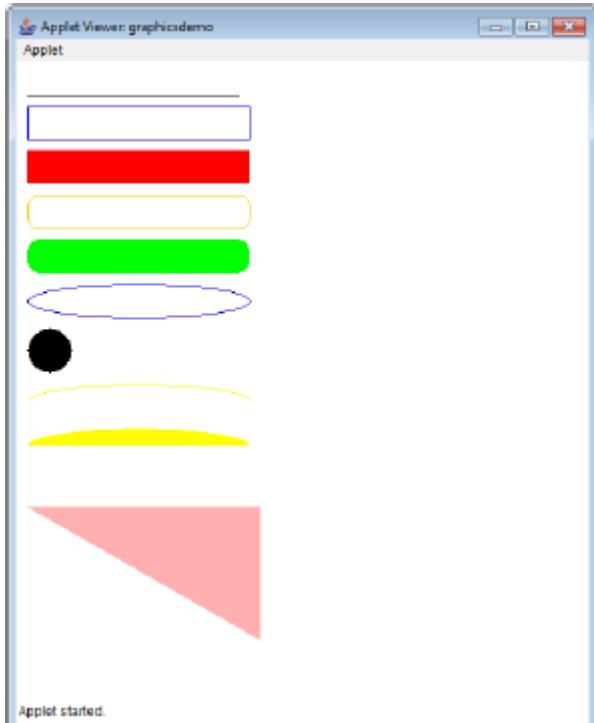
36.Cursor movement using mouse

AIM: To write a JAVA program that display the x and y position of the cursor movement usingMouse.

SOURCE-CODE:

```java
import java.awt.*; import
java.awt.event.*;import
java.applet.*;
//<applet code="mouseevent" width=450 height=300></applet>
public class mouseevent extends Applet
implements MouseListener, MouseMotionListener
{
String s1=" ";int
x,y;
public void init()
{
addMouseListener(this); addMouseMotionListener(this);
}
public void mouseClicked(MouseEvent me)
{ x=100;
y=100;
s1="Mouse clicked";
repaint();
}
public void mouseEntered(MouseEvent me)
{ x=100;
y=200;
s1="Mouse entered";
repaint();
}
public void mouseExited(MouseEvent me)
{ x=100;
y=300;
s1="Mouse exited";
repaint();
}
public void mousePressed(MouseEvent me)
{
x=me.getX();
y=me.getY();
s1="Mouse Pressed";
repaint();
}
public void mouseReleased(MouseEvent me)
```

```
{
x=me.getX();
y=me.getY(); s1="Mouse
Realeased";repaint();
}
public void mouseDragged(MouseEvent me)
{
x=me.getX();
y=me.getY();
s1="Mouse Dragged";
repaint();
}
public void mouseMoved(MouseEvent me)
{
x=me.getX();
y=me.getY();
s1="Mouse Moved";
repaint();
}
public void paint(Graphics g)
{
g.drawString(s1,x,y);
}
}
```

OUT-PUT:
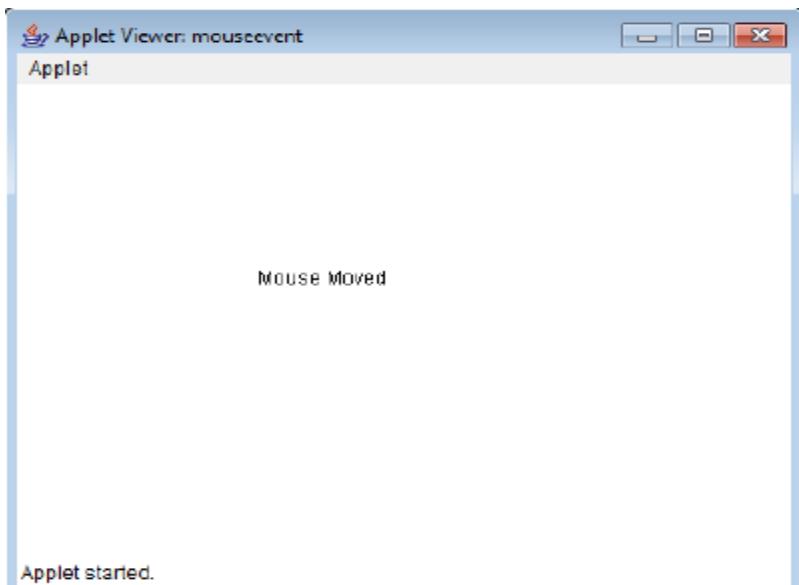
37.Key-up and Key-down event

AIM: To write a JAVA program that identifies key-up key-down event user entering textin a Applet.

SOURCE-CODE:

```java
import java.awt.*; import
java.awt.event.*;import
java.applet.*;
//<applet code="keyevent" width=450 height=300></applet>
public class keyevent extends Applet implements KeyListener
{
String s1=" ";int
x,y;
public void init()
{
addKeyListener(this);
requestFocus();
}
public void keyPressed(KeyEvent ke)
{ x=100;
y=200;
s1= "key pressed ";
repaint();
}
public void keyReleased(KeyEvent ke)
{ x=100;
y=400;
s1= "key Released ";
repaint();
}
public void keyTyped(KeyEvent ke)
{
s1=s1+ke.getKeyChar();
repaint();
}
public void paint(Graphics g)
{
g.drawString(s1,x,y);
}
}
```

OUT-PUT: