Gokaraju Rangaraju Institute of Engineering and Technology

2021

JAVA PROGRAMMING

LAB

KANDUKURI JASWANTH

Gokaraju Rangaraju Institute of Engineering & Technology. Nizampet-Bachupally Road. Kukatpally, Hyderabad. Telangana 500090

Table of Contents

Introduction to JAVA 5

Task-1: Write Java programs that implements the following 6

a) Constructor b) Parameterized overloading c)Method overloading d)Constructor overloading

Task-2: 13

a) Write a Java program that checks whether a given string is a palindrome or not.

Ex: MADAM is a palindrome

b) Write a Java program for sorting a given list of names in ascending order

c) Write a Java program that reads a line of integers, then displays each integer and the sum of all the integer (Use StringTokenizer class of java.util)

Task-3: Write a Java programs that uses the following keywords 20

a) this b)super c) static d)final

Task-4: 30

a) Write a Java program to implement method overriding.

b) Write a Java program to implement dynamic dispatch.

c) Write a Java program to implement multiple inheritance.

d) Write a Java program that uses access specifiers.

Task-5: 39

a) Write a Java program that reads a file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

b) Write a Java program that reads a file and displays the file on the screen, with a line number before each line.

c) Write a Java program that displays the number of characters, lines and words in a text file.

Task-6: 45

a) Write a Java program for handling Checked Exceptions.

b) Write a Java program for handling Unchecked Exceptions

Task-7: 50

a) Write a Java program that creates three threads. First thread displays “Good Morning” every one second, the second thread displays “Hello” every two seconds and the third thread displays “Welcome” every three seconds.

b) Write a Java program that correctly implements producer consumer problem using the concept of inter thread communication.

Task-8: Write a program illustrating following collections framework 56

a) ArrayList b) Vector c) HashTable d) Stack

Task-9: 65

a) Develop an applet that displays a simple message.

b) Develop an applet that receives an integer in one text field and compute itsfactorial value and return it in another text field, when the button named “Compute” is clicked.

c) Write a Java program that works as a simple calculator. Use a grid layout toarrange button for the digits and for the +, -,\*, % operations. Add a text field to display the result

Task-10: 78

a) Write a Java program for handling mouse events.

b) Write a Java program for handling key events

Task-11: 87

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields Num1 and Num 2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1or Num2 were not an integer, the program would throw Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception and display the exception in a message dialog box.

Task-12: 91

a) Write a java program that simulates traffic light. The program lets the user select one of three lights: red, yellow or green. When a radio button is selected, the light is turned on, and only one light can be on at a time No l Light is on when the program starts.

b) Write a Java program that allows the user to draw lines, rectangles and ovals.

INTRODUCTION TO JAVA:

1. Java is related to C++, which is direct descendant of c.

Syntax from c

JAVA

OOP from C++

1. Hot Java 🡪 The first Java-enabled web browser
2. JDK Evolutions-versions
3. J2SE,J2ME,J2EE
4. Java-James Gosling-sun Microsystems,1991
5. Portable, platform(OS)- independent

Any machine

1. The second force was the world wide web(www)
2. Initial name was ‘Oak’ , it was renamed as Java in 1995.
3. First version-18 months

10)JDK(Java-Development kit):

1. Versions-JDK 1.02(1995)
2. JDK 1.1(1996)
3. Java 2 SDK(Skill Development kit) v1.2(JDK1.2,1998)

Task-1:

1. Write a java programs that implements constructor.

Aim: To write a java program that implements a constructor.

Description:

In java is a block of code similar to a method that’s called when an instance of an object is created. A constructor is called automatically when a new instance of an object.

Rules for creating java constructor:

1. Constructor name must be the same as it class name.
2. A constructor must have no explicit return type.
3. A java constructor cannot be abstract, static, final and synchronized.

Algorithm:

Step-1: Start the program.

Step-2: Declare the class as an example with variables.

Step-3: Declare the constructor declaration in class.

Step-4: Define constructor definition outside class with variable initialization.

Step-5: Stop.

Program:

public class Box

{

double width;

double height;

double depth;

Box(double w, double h, double d)

{

width = w;

height = h;

depth = d;

}

double volume()

{

return width\*height\*depth;

}

}

class constructor

{

public static void main(String args[])

{

Box mybox1= new Box(10,50,20);

double vol;

vol = mybox1.volume();

System.out.println(“volume is “+vol);

}

}

Output:

Volume is:1000

Result: Thus, in the above program successfully executed without errors using constructor.

1. Write java programs that implements the parameterized constructor.

Aim: To write Java programs that implements the parameterized constructor.

Description:

A constructor is called parameterized constructor when it accepts a specific number of parameters.

It used to provide different values to distinct objects. However, we can provide the same values.

Types of java constructors:

We have two types in constructor:

1. No Argument constructor (Default constructor)
2. Parameterized constructor

No we are writing program for no constructors with no parameters and parameters.

No parameters = parameterless constructor

Default constructor

user-provided initialization values are provided.

Syntax for default constructor:

<class\_name>(){ }

Algorithm:

Step-1: Start the program.

Step-2: Create a constructor with parameters whose name is the same as the class name.

Step-3: Create an object in the main method and pass values to the constructor.

Step-4: Stop.

Program:

public class student4

{

int id;

String name;

student4( int I, String n);

{

id=i;

name=n;

}

Void display()

{

System.out.println(id+” ” +name);

}

public static void main(String args[])

{

student4 s1=new student4 (111,”sai”);

student4 s2= new student4 (202,”meher”);

s1.diplay();

s2.display();

}

}

Output:

111 sai

202 meher

Result: Thus, in the above program successfully executed without errors using parameterized constructor.

1. Write a java program that implements the method overloading.

Aim: To write a java program that implements the method overloading.

Description:

1. Method overloading is a feature of java in which a class has more than one method of the same name and their parameters are different.
2. Method overloading can be done by changing number of arguments or by changing the data type of argument.
3. If two or more method have same and same parameter list but differs in return type can not be overloading.

Algorithm:

Step-1: Start the program.

Step-2: Create a class A.

Step-3: create two class methods in class A, one is with parameters and another one is with no parameters which calculate the area.

Step-4: Create the main method in another method.

Step-5: Create a new object for class A.

Step-6: Call two methods in class A by pass parameters if required.

Step-7: Stop the program.

Program:

public 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);

}

}

Output:

The area is: 200

The area is: 100

Result: Thus , in the above program successfully executed without errors using method overloading.

1. Write a JAVA program to implement constructor overloading

Aim: To 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 overload like java methods.

It can also be overload like java methods.

1. Java constructor overload helps to achieve static polymorphism.
2. It to allow an instance of a class to be initialized in various ways

It allows to define of the multiple constructors of a class with different signatures

Algorithm:

Step-1: Start the program.

Step-2: Create multiple constructors in a class with a different parameter that assigns the value.

Step-3: Create a main class and main method.

Step-4: Create an object in the main class which automatically calls constructors based on the parameter provided.

Step-5: Stop the program.

Program:

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);

}

}

Output:

The area is: 200

The area is: 1200

Result: Thus , in the above program successfully executed without errors using constructor overloading.

Task-2

1. Write a Java program to implement given string is palindrome or not

AIM: To write a Java program to implement given string is palindrome or not.

Description:

A java string is a sequence of characters that exist as an object of the class java.

What is string in java?

Objects that are backed internally by a characters array. Since arrays are immutable, strings are immutable as well. Whenever a change to a string is made, an entirely new string is created.

So I am writing the difference between string and character in java.

Unlike C/C++ character arrays and strings are two different things in java. Both character arrays and strings are a collection of characters but are different in terms of properties.

Palindrome number in java:

A palindrome number is a number that is same after reverse.

Ex: 151, 48984 are palindrome numbers

We can write also be a string like LOL, MADAM etc.

In c language:

A said to be palindrome if the reverse of the string is same as a string.

Ex: “abba” is palindrome but “abbc” is not palindrome

In this program, we are introducing import java , Scanner and System.in .

Import:

In java, the import keyword used to import used to import built-in and user-defined packages. When a package has imported, we can refer to all the classes of that package using their name directly.

The import statement must be after the package statement, and before any other tatement.

Import java.util\*: Is a built-in java which encapsulates a similar group of classes, sub-packages and interfaces.

Import java.util.Scanner: Util package for obtaining the input of primitive types like int, double etc and strings.

Scanner:

The java scanner class is used to collect user input.

The scanner is part of the java. It is into tokens based on a delimiter.

By default, the scanner class uses white space as the delimiter.

The tokens can be read in either strings:

next(): Is a method of java scanner class which finds and returns the next complete token from the scanner which in using.

nextLine(): Scans from the current position util it finds a line separator delimiter.

System.in: Is a input stream which is typically connected to keyboard input of console programs.

Algorithm:

Step-1: Start the program.

Step-2: Get the string to check for palindrome.

Step-3: Hold the string in the temporary variable.

Step-4: Reverse the string.

Step-5: Compare the temporary with a reversed string.

Step-6: If both strings are the same, print" palindrome string".

Step-7: Else print "not palindrome string".

Step-8: Stop.

Program:

import java.util.Scanner;

class palindrome

{

public static void main(String args[])

{

String str, rev = "";

int i;

System.out.println("enter a string");

Scanner sc= new Scanner(System.in);

str= sc.nextLine();

int length= str.length();

for ( i = length - 1; i >= 0; i-- )

rev = rev + str.charAt(i);

if (str.equals(rev))

System.out.println(str+" is a palindrome");

else

System.out.println(str+" is not a palindrome");

}

}

Output:

Enter a string MADAM

Is a palindrome

Result: Thus, the above program was successfully executed without errors using string palindrome or not.

1. Write a java program for sorting a given list of names is ascending order.

Aim: To write a java program for sorting a given list of names in ascending order

Description:

Enter the size of the array and then enter all the names in that array. Now with the help of the comparison. To the operator, we can easily sort names in alphabetical order.

Algorithm:

Step-1: Start the program.

Step-2: Using the compare To() method compares one string with the rest of the strings.

Step-3: To swap the elements based on the comparison between the two strings.

Step-4: Print the sorted names in alphabetical order.

Step-5: Stop the program.

Program:

import java.util.Scanner;

class SortString

{

public static void main(String args[])

{

String temp;

Scanner SC = new Scanner(System.in);

System.out.print("Enter the value of N: ");

int N= SC.nextInt(); SC.nextLine(); //ignore next line

characterString names[] = new String[N];

System.out.println("Enter names: ");

for(int i=0; i0)

{

temp=names[j-1];

names[j-1]=names[j];

names[j]=temp;

}

}

}

System.out.println("\nSorted names are in Ascending Order: ");

for(int i=0;i <N;i++)

{

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

}

}

Output:

C:ProgramFiles/Java/jdk1.7..0\_09/bin>javac SortString.java C:ProgramFiles/Java/jdk1.7..0\_09/bin>java SortString

Enter the value of N: 4

Enter names: Enter name [ 1 ] : z

Enter name [ 2 ] : w

Enter name [ 3 ] : a

Enter name [4 ] : r

Sorted names are in Ascending Order: a r wz

Result: Thus, the above program was successfully executed without errors using sorting a given list of names is ascending order.

1. Write a Java Program that reads a line of integers, and then displays each integer, and the sum of all the integers (Use StringTokenizer class of java.util)

Aim: To write a Java Program that reads a line of integers, and then displays each integer, and the sum of all the integers (Use StringTokenizer class of java.util).

Description:

These are the smallest elements of a program which are identified by the compiler.

We have 5 types of the tokens are there:

1. Keywords 2) Identifiers 3) Literals 4)operators 5)special symbols

Tokenizer:

The string tokenizer class allows an application to break a string into tokens.

The tokenization method is much simpler than the one used by the stream tokenizer class.

Java uil.\*:

Is built-in package in java which encapsulates a similar group of classes, sub-packages and interfaces.

Close():

It is used to close the stream and release the resources that were busy in the stream, if any.

Algorithm:

Step-1: Start the program.

Step-2: Creates StringTokenizer with the specified string.

Step-3: Creates StringTokenizer with specified string and delimiter.

Step-4: Display all elements.

Step-5: Creates StringTokenizer with specified string, delimiter, and return value. If the return value is true, the delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens.

Step-6: Check if there are more tokens available.

Step-7: Return the next token from the StringTokenizer object.

Step-8: Return the next token based on the delimiter.

Step-9: Same as hasmoretokens() method.

Step-10: Same as nextToken() but its return type is object.

Step-11: Return the total number of tokens.

Step-12: Add all tokens and display the sum of tokens.

Step-13: Stop.

Program:

import java.util.StringTokenizer;

import java.util.Scanner;

public class tokens

{

public static void main(String[] args)

{

String temp; int k,total=0;

Scanner input=new Scanner(System.in);//Take the input from the user.

System.out.println("Enter the Numbers : "); //Display the message to user.

String sentence=input.nextLine(); //read the numbers entered by the user.

//System.out.println("Number="+sentence);

StringTokenizer s1=new StringTokenizer(sentence); //Create object of theString

//Tokenizer and pass the numbers entered by the user to it.

System.out.println("Total Number of tokens:"+s1.countTokens());

while(s1.hasMoreTokens())

{

temp=s1.nextToken();

//nextToken() takes the next token and returns string

k=Integer.parseInt(temp);//Make use of parseInt() method from the Integer class

//to convert the String token into integer token.

total=total+k;//total+=k;

//Perform the addtion for obtaining final sum of all theSystem.out.print(k+"\t")

}

System.out.println("Sum of tokens :"+total); //Display the sum of all the tokensentered

//the user.

}

}

Output:

C:ProgramFiles/Java/jdk1.7..0\_09/bin>javac tokens.java

C:ProgramFiles/Java/jdk1.7..0\_09/bin>java tokens

Enter the Numbers : 1 2 3 4 5

Total Number of tokens : 5 12345

Sum of tokens : 15

Result: Thus, the above program was successfully executed without errors using reads a line of integers, and then displays each integer, and the sum of all the integers.

Task-3

Write a Java programs that uses the following keywords

1. this b)super c) static d)final
2. this keyword:
3. Aim: To write Java programs that use this keyword

Description:

This is a reference variable that refers to the current class object.

What is meant by the current object?

Is the region of memory that contains the data on which a member function currently operates.

It can also differentiate between instance variables and local variables.

Uses:

1. It can be used to refer to current class instance variables.

Syntax: this.variable=value;

we can write an instance variable into the formal parameter.

1. It is used it invoke current class constructor.

Syntax: this(par-list);

1. It is used to invoke the current class method.

Syntax: this.method-name(par-list);

Algorithm:

Step-1: Start the program.

Step-2: This can be used to refer current class instance variable.

Step-3: This can be used to refer current class method(implicity).

Step-4: This () can be used to invoke the current class constructor.

Step-5: This can be passed as an argument in the method call.

Step-6: This can be passed as an argument in the constructor call.

Step-7: This can be used to return the current class instance from the method.

Step-8: Stop.

Program:

public class Box

{

double width;

double height; Instance variable

double depth;

Box {double width, double height, double depth}

Formal parameter

{

this.width=width;

istance variable this.height=height;

this.depth=depth;

}

double volume();

{

Return width\*height\*depth;

}

}

class constructor

{

P ublic static void main(String args[])

{

Box mybox1= new box(10,20,30);

Double vol;

vol = mybox1.volume();

System.out.println(“volume is”+vol);

}

}

Output:

volume=6000

Result: Thus, the above program was successfully executed without errors using this keyword.

1. Aim: To write Java programs that use super keyword

Description:

In java is a reference variable that is used to refer to parent class object.

super can be used to call the parent class' variables and methods. super() can be used to call the parent class' constructors only .

It is used for the following three purposes:

1. For calling the method of the super class.
2. For accessing the member variables of the super class.
3. For invoking the constructors of the super class.

Algorithm:

Step-1: Start the program.

Step-2: For calling the method of the superclass.

Step-3: For accessing the member variables of the superclass.

Step-4: For invoking the constructors of the superclass.

Step-5: Stop.

Program:

Class  Animal

{

String color="white";

}

**class** Dog **extends** Animal

{

String color="black";

**void** printColor()

{

System.out.println(color);//prints color of Dog class

System.out.println(**super**.color);//prints color of Animal class

}

}

**class** TestSuper1

{

public static void main (String args[]){

Dog d= new  Dog();

d.printColor();

}

}

Output:

black

white

or

. class Person

{

int id;

String name;

Person(int id,String name)

{

this.id=id;

this.name=name;

}

}

class Emp extends Person

{

float salary;

Emp(int id,String name,float salary)

{

super(id,name);//reusing parent constructor

this.salary=salary;

}

. void display()

{

System.out.println(id+" "+name+" "+salary);

}

}

class TestSuper5

{

public static void main(String[] args){

Emp e1=new Emp(1,"ankit",45000f);

e1.display();

}

}

Output:

1 ankit 45000

Result: Thus, the above program was successfully executed without errors using super keyword

1. Aim: To write Java programs that use static keyword

Description:

When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

We can declare both methods and variables to be static.

The static keyword includes local variables, instance variables, and class/static variables.

1. Local variable:

If we declared a local variable into a static keyword the variable is now created at the start of the program and destroyed at the end of the program.

Some access specifiers like private, public, and protected cannot be used with local variables

1. Instance variable:

We can declare inside a class, but outside of a method.

They are also called a data member, fields, or attributes.

* Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used.
* Initialization of Instance Variable is not Mandatory. Its default value is 0
* Instance Variable can be accessed only by creating objects.

**Syntax:** Static and instance variables

class GFG

{

// Static variable

static int a;

// Instance variable

int b;

}

class/static variable:

* the variable which belongs to the class and is initialized only once at the start of the execution

Static variables are also known as Class variables.

Java does not allow global variables.

Static variables are used to  keep track of information that relates logically to an entire class as opposed to information that varies from instance to instance.

Example: 1 A class having class/static variables

class test

{

int x=0;

static int y=0;

//setter methods

void setx(int n)

{

x=n

}

void sety(int n)

{

y=n

}

//getter methods

int getx()

{

return x;

}

int gety()

{

return y;

}

Static method:

Is used to create methods that will exist independently of any instances created for the class. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables

Example: 1

class test1

{

int x=3;

static int return

{

return x;

}

public static void main(String args[])

{

System.out.println(return());//static method invoked directly

}

}

Example-2 A class having static members:

class T

{

int x=3;

static int returnIt()

{

return x;

}

}

class T1

{

public static void main(String args[])

{

System.out.println(T.returnIt());

}

}

class area

{

static int area;//static variable

static int computearea (int width.int height)

{

area=width\*height;

return area;

}

}

Example-4 A class calling method ,defined in another class

class area

{

public static void main(String args[])

{

System.out.println(area.compute (4,3));

}

}

Algorithm:

Step-1: Start the program.

Step-2: Create a static variable with some value.

Step-3: Access static variables directly in the static method of the same class.

Step-4: Public static variables can be used in other classes with the class name and object creation is not needed.

Step-5: The static block gets initialized and displays the values of the initialized variables.

Step-6: They can directly call other static methods only.

Step-7: They can access static data directly.

Step-8: Stop the program.

Program:

//Java Program to demonstrate the use of static variable

**class** Student

{

**int** rollno;//instance variable

   String name;

**static** String college ="ITS";//static variable

   //constructor

   Student(**int** r, String n)

{

    rollno = r;

    name = n;

  }

    //method to display the values

**void** display ()

{

System.out.println(rollno+" "+name+" "+college);

}

}

//Test class to show the values of objects

**public** **class** TestStaticVariable1

{

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

  Student s1 = **new** Student(111,"Karan");

  Student s2 = **new** Student(222,"Aryan");

  //we can change the college of all objects by the single line of code

  //Student.college="BBDIT";

  s1.display();

  s2.display();

  }

}

Output:

111 karan ITS

222 Aryan ITS

Result: Thus, the above program was successfully executed without errors using static keyword

Aim: To write Java programs that use final keyword

Description:

The final keyword is a **non-access modifier used for classes, attributes and methods**, which makes them non-changeable (impossible to inherit or override)

the keyword final is used for the following purposes:

1. To declare constants(used with variable and arguments declaration)
2. To disallow method overriding (used with method declaration)
3. To disallow inheritance (used with class declaration)

It is used to prevent inheritance and create constants.

Program:

class Final\_ demo

{

final intMax= 100; //constant declaration

final void show(final int x)

{

System.out.println(“Supper Class show method:”+x);

Class Final\_demo\_1 extends Final\_ demo

{

System.out.println(“Sub Class show method”);

}

public static void main(String args[])

Final\_demo\_1 f2= new Final\_demo\_1();

f2.show(12);

}

}

Output:

Super Class show method:12

Result: Thus, the above program was successfully executed without errors using final keyword

Task-4

1. Write a java program to implement method overriding.

Aim: To write a java program to implement method overriding

Description:

A method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass.

It is a feature that supports polymorphism.

In method overloading, we have done the same name, but different parameter lists but here method overriding we have done the same name and same parameter lists passing in a class.

Algorithm

Step-1: Start the program

Step-2: In Java, a method can only be written in the child class and not in the same class.

Step-3: The argument list should be the same as that of the overridden method of that class.

Step-4: Instance methods can also be overridden if they are inherited by the child class.

Step-5: A constructor cannot be overridden.

Step-6: Final - declared methods cannot be overridden.

Step-7: Any static method cannot be used to override.

Step-8: The return type must have to be the same, or a subtype of the return type declared in the original overridden method in the parent class.

Step-9: If a method cannot be inherited, then it cannot be overridden.

Step-10: A child class within the same package as the instance's parent class can override any parent class method that is not declared private or final.

Step-11: A child class in a different package can only override the non-final methods declared as public or protected.

Step-12: Stop.

Program:

public class A

{

int i=0;

void dooverride (int k)

{

i=k;

}

}

//Subclass definition starts here

class B extends A

{

//method overriding

void dooverriding (int k)

{

i = 2\*k;

System.out.println("The value of i is:" +i);

}

public static void main(String[] args)

{

B b= new B(); //create an instance of class B

b. do override(12); //class B method do override()

}

}

Output:

The value of i is : 24

Or

class college

{

public void move()

{

System.out.println("College is open");

}

}

class univ extends college

{

public void move()

{

System.out.println("University is open too");

}

}

public class stud

{

public static void main(String args[])

{

college a = new college();

college b = new univ();

a.move();

b.move();

}

}

Output:

College is open

University is open too

Result: , the above program was successfully executed without errors using overriding method.

1. Write a java program to implement dynamic method dispatch.

Aim: To write a java program to implement dynamic method dispatch.

Description:

Dynamic method dispatch is a mechanism by which a call to an overridden method is resolved at runtime. This is how java implements runtime polymorphism. When an overridden method is called by a reference, java determines which version of that method to execute based on the type of object it refers to. In simple words, the type of object to which it refers determines which version of the overridden method will be called.

Parent

child

Extends

Parent p= new parent();

Child c= new child();

Parent p= new child();

Upcasting

Child c= new parent();

Incompatible type

Upcasting:

When **Parent** class reference variable refers to **Child** class object, it is known as **Upcasting.**

Algorithm:

Step-1: Start the program

Step-2: Inside the void() method in class Test, initially, objects of game type Indoor and Outdoor are declared.

Step-3:When outdoor; will be executed, it will call the method game type() of the class game because the reference variable “cricket” is pointing towards the object of the class game.

Step-4: gm.type(); will call ck.type() of class game because reference variable “ck” refers to object of class cri cket.

Step-5: Stop.

Program:

class Game

{

public void type()

{

System.out.println("Indoor and outdoor");

}

}

Class Cricket extends Game

{

public void type()

{

System.out.println("outdoor game");

}

public static void main(String[] args)

{

Game gm = new Game();

Cricket ck = new Cricket();

gm.type();

ck.type();

gm = ck; //gm refers to Cricket object

gm.type(); //calls Cricket's version of type

}

}

Output:

Indoor and outdoor

Outdoor game

Result: , the above program was successfully executed without errors using dynamic method dispatch.

1. Write a java program to implement multiple inheritance.

Aim: To write a java program to implement multiple inheritances.

Description:

Inheritance: The mechanism of deriving a new class from an old one is called inheritance.

The old class is known as “base” or “super” or “parent” class.

Old base

Super

Parent

New derived

Sub

Child

Multiple inheritance:

A class can inherit from more than one unrelated class.

c

B

A

Java does not support multiple inheritances amongst classes. It can still be achieved with the help of interfaces.

Algorithm:

Step-1: Start the program

Step-2:Declare the interface Interface1

Step-3: Declare and define the function show() to get the Interfaces details

Step-4:Declare the other class Interfaces

Step-5:Declare and define the function void show() to read the interfaces

Step-6:Create the class statement derived from class and obj.

Step-7:Declare and define the function display() to find out the obj.show()

Step-8: Declare the derived class object, call the functions obj. show()

Step-9:Stop

Program:

interface Interface1

{

public void show();

}

interface Interface2

{

public void show();

}

public class SubClass implements Interface1, Interface2

{

public void show()

{

System.out.println("A class can implements more than one interfaces");

}

public static void main(String args[]){

SubClass obj = new SubClass();

obj.show();

}

}

Output:

A class can implements more than one interfaces

Result: , the above program was successfully executed without errors using multiple inheritance.

1. Write a java program that uses access specifiers.

Aim: To write a java program that uses access specifiers

Description:

Methods and data members of a class/interface can have one of the following four access specifiers. The access specifiers are listed according to their restrictiveness order.

1. public (accessible from any class)
2. private (accessible within the class where defined)
3. protected
4. default or package-private (when no access specifier is specified)
   1. Public: Can be accessed by any other code.
   2. Private: Can only be accessed by other members of its class.
   3. Protected: Applies only when inheritance is involved.
   4. Default: When no access specifier is used, then by default member of a class is public within its package (own), but cannot be accessed outside of its package.

Not a keyword.

Algorithm:

Step-1: Start the program

Step-2: Default declarations are visible only within the package (package private)

Step-3: Private declarations are visible within the class only

Step-4: Protected declarations are visible within the package or all subclasses

Step-5: Public declarations are visible everywhere

Step-6: Stop

Program:

// use of public and default accesss specifier

public class Rectangle

{

public int length; // public access

public int breadth; //public access

static int rectCount = 0;// default access

//Constructor to initialize length and breadth

Rectangle()

{ rectCount++;}

//method to calculate area of rectangle

public int area()

{

int rectArea;

rectArea = length \* breadth;

return rectArea;

}

}

class Access Specifer

{

public static void main(String[] args)

{

//create first rectangle object

Rectangle firstRect = new Rectangle();

//accessing public members outside that Rectangle class

firstRect.length = 10;

firstRect.breadth = -20;

System.out.println("Area of Rectangle is : "+ firstRect.area());

//Rectangle.rectCount = 5;

//accessing member with default access

System.out.println("Number of Object Created : "+ Rectangle.rectCount);

}

}

Output:

Enter of rectangle is:-200

Number of objectcreated:1

Result: the above program was successfully executed without errors using access specifier.

Task-5

1. Write a Java program that reads a file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

Aim: To write a Java program that reads a file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file, and the length of the file in bytes.

Description:

File handling: How we can read and write data on a file.

Java io package contains all the classes through which we can perform all input and output operations in the file.

we have file handling methods in java:

1. canread(): This function determines whether the program can read the file denoted by the abstract pathname.

The function returns true if the abstract file path exists and the application is allowed to read the file.

Syntax:

file.canRead()

Parameters: This method does not accept any parameter.

1. Canwrite(): This function determines whether the program can write the file denoted by the abstract path name.

Syntax:

File.canWrite()

1. Create Newfile(): Is a method of file class which belongs to a java.io package.
2. It does not accept any argument.
3. The method automatically creates a new empty file.
4. The method returns a Boolean value:
5. true, if the file was created successfully.
6. false, if the file already exists.

Syntax:

public Boolean createNewFile() throws IOExpception

1. Delete():Is an inbuilt method in Java that is used to remove or delete the characters in a substring of this sequence. The substring starts at a specified index start\_point and extends to the character at the index end\_point.

Syntax:

public StringBuffer delete(int start\_point, int end\_point)

1. Exists(): The method returns Boolean true, if and only if the file defined by the abstract pathname exists; else false.

Syntax:

file.exists()

1. Length(): The java string length() method finds the length of a string.

The length of the java string is the same as the Unicode code units of the string.

Syntax:

public int length()

1. getName(): This method of java class is used to get the name of entity, and that entity can be class, interface, array, enum, method etc. of the class object.

Syntax:

file.getName()

1. list(): Java provides the facility to maintain the ordered collection. It contains the index-based methods to insert, update, delete and search the elements. It can have duplicate elements also. We can also store the null elements in the list.

Syntax:

public interfaceList<E>extendsCollection <E>

1. Read() : In java is used to read a single character from the stream.

Syntax:

public abstract int read()

1. Write(): Is used to read a single character from the stream in java is used to write character-oriented data to a file as this class is a character-oriented class because of what is used in file handling in java.
2. Rename():Changing the name of the file is known as Renaming the file.

And we have file handling classes:

1. File
2. Filereader
3. Filewriter
4. Fileinputstream
5. Fileoutputstream
6. Bufferinputstream
7. Bufferoutputstream

Algorithm:

Step-1: Start the program

Step-2: import java.io.\*; package

Step-3: create class Filedemo{

Step-4: Create function: public static void p(String str){

Step-5: Againg create another function: public static void analyze(String s){

Step-6: if( f.exists()){

Step-7: if(f.isDirectory()) then

Step-8: Create main function: public static void main(String rr[])throws IOException{

Step-9: Stop

Program:

import java.io\*;

import java.util.\*;

public class AboutFile

{

public static void main(String [] args)

{

Scanner input= new Scanner(System.in);

System.out.println(“Enter the name of the file:”);

String file\_name=input(file\_name);

File f=new File(file\_name);

if(f.exists())

System.out.println(“The file” +file\_name+”exist”);

else

System .out.println(“The file”+file\_name+”does not exist”);

if(f.exists())

{

if(f.canRead())

System.out.println(“The file”+file\_name+”is readable”);

else

System.out.println(“ The file”+file\_name+”is not readable”);

if(f.canwrite())

System.out.println(“ The file”+file\_name+”is readable”);

else

System.out.println(“ The file”+file\_name+”is not readable”);

System.out.println(“The file type is:” +file\_name.substring(file\_name.indexof(‘.’)+1));

System.out.println(“The Length of the file:”+f.length());

}

}

}

Output:

The file Abc.java exist

The file Abc.java is readable

The file Abc.java is writeable

The file type is: java

The Length of the file:1846

Result: The above program was successfully executed without errors using file exists

1. Write a Java program that reads a file and displays the file on the screen, with a line number before each line

Aim: To write a Java program that reads a file and displays the file on the screen, with a line number before each line

Description:

A computer program is a collection of instructions that performs a specific task when executed by a computer.

The computer requires programs to function.

A computer program is usually written by a computer programmer in the programming language.

A collection of computer programs, libraries, and related data are referred to as software.

Computer programs may be categorized along functional lines, such as application software and system software.  
Algorithm:

Step-1: Start the program

Step-2: import java package

Step-3: Create class Linenum

Step-4: Create main function

Step-5: public static void main( String[] args)throws IOException

Step-6: Create try block

Step-7: Catch( FileNotFoundException e )

Step-8: Print No such file found

Step-9: Inside do

Step-10: read line i = line.read();

Step-11: Use if condition

Step- 12: Use while loop:

Step-13: Stop

Program:

import java.util.\*;

import java.io.\*;

public class Rfile

{

public static void main(String args[])throws IOExpception

{

int j=1;

char ch;

scanner scr=new scanner(System.in);

System.out.println(“\nEnter File name:”);

String str=scr.next( );

FileInputStream f= new FileInputStream(str);

System.out.println(“\nContents of the file are”);

int n=f.available();

System.out.println(n);

System.out.println(j+” ”);

for(int i=0;i<n;i++)

{

ch=(char)f.read();

System.out.println(ch);

if(ch==’\n’)

{

System.out.println(++j+”:”);

}

}

}

}

Output:

Enter File name: raj.txt

Contents of the file are :

1: kotireddy

2: Santhosh

3: Vamsi

Result: The above program was successfully executed without errors using reads a file and displays the file on the screen

1. Write a Java program that displays the number of characters, lines and words in a text file

Aim: To write a Java program that displays the number of characters, lines, and words in a text file

Description:

Stream: This is a sequence of data.

On the basis of the java.io package stream.

Byte character

Byte: Is a signed integer based on the two’s complement 8-bit mechanism.

The values that can be stored in a single byte are -128 to 127

Character: The Java programming language provides a wrapper class that "wraps" the **char** in a Character object for this purpose.

Algorithm:

Step-1: Start the program

Step-2: Open a file in reading mode using file pointer

Step-3: Read a line from a file.

Step-4: Split the line into words and store it in an array

Step-5: Iterate through the array, increment count by 1 for each word.

Step-6: Repeat all these steps till all the lines from the files have been read.

Step-7: Stop

Program:

import java.util.\*;

import java.io.\*;

class Cfile

{

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

            {

                        int nl=1,nw=0;

                        char ch;

                        Scanner scr=new Scanner(System.in);

                        System.out.println("\nEnter File name: ");

                        String str=scr.nextLine();

                        FileInputStream f=new FileInputStream(str);

                        int n=f.available();

                        for(int i=0;i<n;i++)

                        {

                                    ch=(char)f.read();

                                    if(ch=='\n')

                                    nl++;

                                    else if(ch==' ')

                                                nw++;

                        }

                        System.out.println("\nNumber of lines : "+nl);

                        System.out.println("\nNumber of words : "+(nl+nw));

                        System.out.println("\nNumber of characters : "+n);

            }

}

Output:

Enter File name:raj.txt

Number of lines:3

Number of words:3

Number of characters:26

Result: The above program was successfully executed without errors using displays the number of characters, lines and words in a text file.

Task-6

1. Write a java program for handling checked exceptions.

Aim: To write a java program for handling checked exceptions.

Description:

Why this compilation error?

As I mentioned at the beginning that checked exceptions get checked during compile time. Since we didn’t handle/declare the exceptions, our program gave the compilation error.

How to resolve the error?

There are two ways to avoid this error. We will see both the ways one by one.

Method 1: Declare the exception using the throws keyword.

As we know that all three occurrences of checked exceptions are inside the main() method so one way to avoid the compilation error is: Declare the exception in the method using the throws keyword. You may be thinking that our code is throwing FileNotFoundException and IOException both then why we are declaring the IOException alone. The reason is that IOException is a parent class of FileNotFoundException so it by default covers that. If you want you can declare them like this public static void main(String args[]) throws IOException, FileNotFoundException.

Algorithm:

Step-1: Start the program.

Step-2: Used with method (or constructor) signature

Step-3: Used to declare exceptions

Step-4: Can declare multiple exceptions

Step-5: Followed by an exception class name

Step-6: Can be used to propagate checked exceptions by itself

Step-7: Stop

Program:

import java.io.\*;

public class Example

{

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

{

FileInputStream fis = null;

fis = new FileInputStream("B:/myfile.txt");

int k;

while(( k = fis.read() ) != -1)

{

System.out.print((char)k);

} fis.close();

}

}

Output:

Exception in thread "main" java.io.FileNotFoundException: B:/myfile.txt (No such file or directory)

at java.base/java.io.FileInputStream.open0(Native Method)

at java.base/java.io.FileInputStream.open(FileInputStream.java:216)

at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)

at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111)

at Example.main(Example.java:7)

Method 2: Handle them using try-catch blocks.

The approach we have used above is not good at all. It is not the best exception handling practice. You should give a meaningful message for each exception type so that it would be easy for someone to understand the error.

The code should be like this:

Algorithm:

Step-1: Start the program.

Step-2: Add a try/catch block to the lambda expression

Step-3:Create an extracted method, as in the unchecked example

Step-4: Write a wrapper method that catches checked exceptions and rethrows them unchecked.

Step-5: stop

Program:

import java.io.\*;

class Example

{

public static void main(String args[])

{

FileInputStream fis = null;

try

{

fis = new FileInputStream("B:/myfile.txt");

}

catch(FileNotFoundException fnfe)

{

System.out.println("The specified file is not " + "present at the given path");

}

int k;

try

{

while(( k = fis.read() ) != -1)

{

System.out.print((char)k);

}

fis.close();

}

catch(IOException ioe)

{

System.out.println("I/O error occurred: "+ioe);

}

}

}

Output:

The specified file is not present at the given path

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "java.io.FileInputStream.read()" because "<local1>" is null

at Example.main(Example.java:17)

Result: Thus, above program was successfully executed without errors using CheckedExceptions.

1. Write a java program for handling UncheckedExceptions.

Aim: To write a program for handling UncheckedExceptions.

Description:

Unchecked exceptions are not checked at compile time. It means if your program is throwing an unchecked exception and even if you didn’t handle/declare that exception, the program won’t give a compilation error. Most of the time this exception occurs due to the bad data provided by the user during the user-program interaction. It is up to the programmer to judge the conditions in advance, that can cause such exceptions and handle them appropriately. All Unchecked exceptions are direct subclasses of RuntimeException class.

Unchecked Exception Example-1

Algorithm:

Step-1: Start the program.

Step-2: My array has only 5th elements but we are trying to

display the value of the 8th element.

Step-3: It should throw ArrayIndexOutOfBoundsException.

Step-4: print the output.

Step-5: Stop.

Program-1:

public class Example

{

public static void main(String args[])

{

int arr[] ={1,2,3,4,5};

/\* My array has only 5 elements but we are trying to

\* display the value of 8th element. It should throw

\* ArrayIndexOutOfBoundsException

\*/

System.out.println(arr[7]);

}

}

Output:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 7 out of bounds for length 5

at Example.main(Example.java:9)

Explanation:

If you compile this code, it would compile successfully however when you will run it, it would throw ArithmeticException. That clearly shows that unchecked exceptions are not checked at compile-time, they occur at runtime.

Unchecked Exception Example-2

Algorithm:

Step-1: Start the program.

Step-2: My array has only 5 elements but we are trying display the value of the 8th element.

Step-3: It should throw ArrayIndexOutOfBoundsException.

Step-4: print the output.

Step-5: Stop.

Program-2:

public class Example

{

public static void main(String args[])

{

int arr[] ={1,2,3,4,5};

/\* My array has only 5 elements but we are trying to

\* display the value of 8th element. It should throw

\* ArrayIndexOutOfBoundsException

\*/

System.out.println(arr[7]);

}

}

Output:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 7 out of bounds for length 5

at Example.main(Example.java:9)

Explanation:

This code would also compile successfully since ArrayIndexOutOfBoundsException is also an unchecked exception.

Note: It doesn’t mean that compiler is not checking these exceptions so we shouldn’t handle them. In fact, we should handle them more carefully. For e.g. In the above example, there should be an exception message to the user that they are trying to display a value that doesn’t exist in the array so that the user would be able to correct the issue.

Unchecked Exception Example-3

public class Example

{

public static void main(String args[])

{

try

{

int array[] ={1,2,3,4,5};

System.out.println(array[7]);

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("The specified index does not exist in array. Please correct the error.");

}

}

}

Output:

The specified index does not exist in array. Please correct the error.

Result: Thus, above program was successfully executed without errors using UncheckedExceptions.

Task-7

1. Write a java program that creates three threads. First thread displays “Good Morning” every one second the second thread display “Hello” every two seconds and the third thread displays “Welcome” every three seconds.

Aim: To write a java program that creates three threads. The first thread displays “Good Morning” every one second the second thread display “Hello” every two seconds and the third thread displays “Welcome” every three seconds.

Description:

1. Create three classes by GoodMorning, Hello, Welcome with extends of Threads

2. All these three classes will implement Runnable interfaces.

3. One class thread will print Goodmorning for every one second. sleep this thread for 1000 seconds.

4. Second class thread will print Hello every two seconds. sleep this thread for 2000 seconds.

5. Third class thread will print World for every three seconds. sleep this thread for 3000 seconds.

Algorithm:

Step-1: import java.io.\*;

Step-2: Create class A and extends Thread

Step-3: create string name.

Step-4: Create class function and pass argument:

* 1. A(String tname)

Step-5: name = tname

Step-6: Print name: System.out.print(name)

Step-7: public void run()

Step-8: Again Print (name)

Step-9: Create another class Hellow Thread

Step-10: Create the main function

1. public static void main(String args[])

Step-11: Create an object: A obj1= new A(“Good Morning”);

Step-12: Within try, block create, Thread. sleep(1000); and close try block.

Step13: Same you can follow. Create an object: A obj2= new A(“Hello”); and within a try, block create thread. sleep(1000).

Step-14: End.

Program:

public class GoodMorning extends Thread

{

synchronized public void run()

{

try

{

int i=0;

while (i<5)

{

sleep(1000);

System.out.println("Good morning ");

i++;

}

}

catch (Exception e)

{

}

}

}

class Hello extends Thread

{

synchronized public void run()

{

try

{

int i=0;

while (i<5)

{

sleep(2000);

System.out.println("hello");

i++;

}

}

catch (Exception e)

{

}

}

}

class Welcome extends Thread

{

synchronized public void run()

{

try

{

int i=0;

while (i<5)

{

sleep(3000);

System.out.println("welcome");

i++;

}

}

catch (Exception e)

{

}

}

}

class MultithreadDemo

{

public static void main(String[] args)

{

GoodMorning t1 = new GoodMorning();

Hello t2 = new Hello();

Welcome t3 = new Welcome();

t1.start();

t2.start();

t3.start();

}

}

Output:

Good morning

Good morning

hello

welcome

Good morning

hello

Good morning

Good morning

welcome

hello

hello

welcome

hello

welcome

welcome

Result: Thus, the above program was successfully executed without errors creates three threads.

1. Write a Java program that correctly implements producer-consumer problems using the concept of inter-thread communication.

Aim: To write a Java program that correctly implements producer-consumer problems using the concept of inter-thread communication.

Description:

The producer-consumer issue (is also known as the bounded buffer issue) is a classic Java Example of a multi-process synchronization issue. The issue describes two processes, the producer and the consumer, who share a typical, fixed-size buffer used as a queue.

The producer’s responsibility is to create a piece of data, put it into the buffer and start once more. At the same time, the consumer is consuming the data (i.e., eliminating it from the buffer) each piece in turn. The issue is to ensure that the producer will do whatever it takes not to add data into the buffer if it’s full and that the consumer will make an effort not to remove data from an empty buffer.

Algorithm:

Step-1: We create a class that extends java. lang.Thread class.

Step-2: This class overrides the **run()** method available in the Thread class.

Step-3: A thread begins its life inside the **run()** method.

Step-4: We create an object of the Thread class and call **the start()** method to start the execution of a thread

Step-5:  **Start()** invokes the **run()** method on the thread object.

Step-6: stop.

Program:

public class Q

{

int n;

boolean valueSet=false;

synchronized int get()

{

if(!valueSet)

try

{

wait();

}

catch(InterruptedException e)

{

System.out.println("Interrupted Exception caught");

}

System.out.println("Got:"+n);

valueSet=false;

notify();

return n;

}

synchronized void put(int n)

{

if(valueSet)

try

{

wait();

}

catch(InterruptedException e)

{

System.out.println("Interrupted Exception caught");

}

this.n=n;

valueSet=true;

System.out.println("Put:"+n);

notify();

}

}

class Producer implements Runnable

{

Q q;

Producer(Q q)

{

this.q=q;

new Thread(this,"Producer").start();

}

public void run()

{

int i=0;

while(true)

{

q.put(i++);

}

}

}

class Consumer implements Runnable

{

Q q;

Consumer(Q q)

{

this.q=q;

new Thread(this,"Consumer").start();

}

public void run()

{

while(true)

{

q.get();

}

}

}

class ProdCons

{

public static void main(String[] arrgs)

{

Q q=new Q();

new Producer(q);

new Consumer(q);

System.out.println("Press Control-c to stop");

}

}

Output:

D:\Lab>javac ProducerConsumer.java

D:\Lab>java ProducerConsumer

put0

Press Control+c to exit

get0

put1

get1

put2

get2

put3

Result: Thus, the above program was successfully executed without errors concept of inter-thread communication.

Task-8

Write a program illustrating following collections framework

1. ArrayList b) Vector c) HashTable d) Stack

Framework:

The **Collection in Java** is a framework that provides an architecture to store and manipulate a group of objects.

Java Collections can achieve all the operations that you perform on data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

What is Collection in Java

A Collection represents a single unit of objects, i.e., a group.

#### What is a framework in Java?

* It provides readymade architecture.
* It represents a set of classes and interfaces.
* It is optional

#### What is the Collection framework?

The Collection framework represents a unified architecture for storing and manipulating a group of objects. It has:

1. Interfaces and their implementations, i.e., classes
2. Algorithm

### **Hierarchy of Collection Framework**

#### Let us see the hierarchy of the Collection framework. The **java. util** package contains all the [classes](https://www.javatpoint.com/object-and-class-in-java) and [interfaces](https://www.javatpoint.com/interface-in-java) for the Collection framework



### Methods of Collection interface

There are many methods declared in the Collection interface. They are as follows:

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | public boolean add(E e) | It is used to insert an element in this collection. |
| 2 | public boolean addAll(Collection<? extends E> c) | It is used to insert the specified collection elements in the invoking collection. |
| 3 | public boolean remove(Object element) | It is used to delete an element from the collection. |
| 4 | public boolean removeAll(Collection<?> c) | It is used to delete all the elements of the specified collection from the invoking collection. |
| 5 | default boolean removeIf(Predicate<? super E> filter) | It is used to delete all the elements of the collection that satisfy the specified predicate. |
| 6 | public boolean retainAll(Collection<?> c) | It is used to delete all the elements of invoking collection except the specified collection. |
| 7 | public int size() | It returns the total number of elements in the collection. |
| 8 | public void clear() | It removes the total number of elements from the collection. |
| 9 | public boolean contains(Object element) | It is used to search an element. |
| 10 | public boolean containsAll(Collection<?> c) | It is used to search the specified collection in the collection. |
| 11 | public Iterator iterator() | It returns an iterator. |
| 12 | public Object[] toArray() | It converts collection into array. |
| 13 | public <T> T[] toArray(T[] a) | It converts collection into array. Here, the runtime type of the returned array is that of the specified array. |
| 14 | public boolean isEmpty() | It checks if collection is empty. |
| 15 | default Stream<E> parallelStream() | It returns a possibly parallel Stream with the collection as its source. |
| 16 | default Stream<E> stream() | It returns a sequential Stream with the collection as its source. |
| 17 | default Spliterator<E> spliterator() | It generates a Spliterator over the specified elements in the collection. |
| 18 | public boolean equals(Object element) | It matches two collections. |
| 19 | public int hashCode() | It returns the hash code number of the collection. |

### **Iterator interface**

  Iterator interface provides the facility of iterating the elements in a forward direction only.

#### Methods of Iterator interface

There are only three methods in the Iterator interface. They are:

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | public boolean hasNext() | It returns true if the iterator has more elements otherwise it returns false. |
| 2 | public Object next() | It returns the element and moves the cursor pointer to the next element. |
| 3 | public void remove() | It removes the last elements returned by the iterator. It is less used. |

## **Iterable Interface**

The iterable interface is the root interface for all the collection classes. The Collection interface extends the iterable interface and therefore all the subclasses of the Collection interface also implement the iterable interface.

It contains only one abstract method. i.e.,

1. Iterator<T> iterator()

It returns the iterator over the elements of type T**.**

**Collection Interface**

The Collection interface is the interface that is implemented by all the classes in the collection framework. It declares the methods that every collection will have. In other words, we can say that the Collection interface builds the foundation on which the collection framework depends.

**List Interface**

List interface is the child interface of the Collection interface. It inhibits a list-type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

To instantiate the List interface, we must use :

1. List <data-type> list1= **new** ArrayList();

1. List <data-type> list2 = **new** LinkedList();

3. List <data-type> list3 = **new** Vector();

1. List <data-type> list4 = **new** Stack();

There are various methods in the List interface that can be used to insert, delete, and access the elements from the list.

1. Write a program illustrating collections framework ArrayList

Aim: To write a program illustrating collections framework ArrayList

Description:

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. Consider the following example.

Algorithm:

Step-1: Start the program.

Step-2: Is unsynchronized therefore its performance is better in a single-threaded environment.

Step-3: Is not thread-safe so care must be given when used in a multithreaded environment.

Step-4: Can contain duplicate values.

Step-5: Allows “null” value.

Step-6: Objects in java ArrayList are added in order.

Step-7: default capacity is defined as 10. We can change the default capacity through the constructor or by calling ensure capacity(int incapacity) method.

Step-8: Iterator and ListIterator implementation is fail-fast. If the list structure is modified after creating the iterator in any other way except the iterator add or remove methods, it will throw ConcurrentModificationException

Step-9: Provides random access.

Step-10: supports Generics and is the recommended way.

Step-11: Stop the program.

Program:

import java.util.\*;

public class TestJavaCollection1

{

public static void main(String args[])

{

ArrayList<String> list=new ArrayList<String>();//Creating arraylist

list.add("Ravi");//Adding object in arraylist

list.add("Vijay");

list.add("Ravi");

list.add("Ajay");

//Traversing list through Iterator

Iterator itr=list.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

}

}

}

Output:

Ravi

Vijay

Ravi

Ajay

Result: Thus, the above program was successfully executed without errors using ArrayList.

1. Write a program illustrating collections framework Vector

Aim: To write a program illustrating collections framework Vector

Description:

Vector uses a dynamic array to store the data elements. It is similar to ArrayList. However, It is synchronized and contains many methods that are not the part of Collection framework.

Consider the following example.

Algorithm:

Step-1: Start the program.

Step-2: This constructor is used to create an empty vector so that its internal data array has size 10 and its standard capacity increment is zero.

Step-3: This constructor is used to create a vector containing the elements of the specified collection, in the order, they are returned by the collection's iterator.

Step-4: This constructor is used to create an empty vector with the specified initial capacity and with its capacity increment equal to zero.

Step-5: This constructor is used to create an empty vector with the specified initial capacity and capacity increment.

Step-6: Stop

Program:

import java.util.\*;

public class TestJavaCollection3

{

public static void main(String args[])

{

Vector<String> v=new Vector<String>();

v.add("Ayush");

v.add("Amit");

v.add("Ashish");

v.add("Garima");

Iterator<String> itr=v.iterator();

while(itr.hasNext( ))

{

System.out.println(itr.next( ));

}

}

}

Output:

Ayush

Amit

Ashish

Garima

Result: Thus, the above program was successfully executed without errors using Vector.

1. Write a program illustrating collections framework HashTable

Aim: To write a program illustrating collections framework HashTable

Description:

Java Hashtable class implements a hashtable, which maps keys to values. It inherits Dictionary class and implements the Map interface.

### **Hashtable class declaration**

Let's see the declaration for java. util.Hashtable class.

1.    **public** **class** Hashtable<K,V> **extends** Dictionary<K,V> **implements** Map<K,V>, Cloneable, Serializable

### **Hashtable class Parameters**

Let's see the Parameters for java. util.Hashtable class.

* **K**: It is the type of keys maintained by this map.
* **V**: It is the type of mapped values.

Algorithm:

Step-1: Start the program.

Step-2: A Hashtable is an array of a list. Each list is known as a bucket. The position of the bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.

Step-3: Java Hashtable class contains unique elements.

Step-4: Java Hashtable class doesn't allow null keys or values.

Step-5: Java Hashtable class is synchronized.

Step-6: The initial default capacity of the Hashtable class is 11 whereas the load factor is 0.75.

Step-7: Stop.

Program:

import java.util.\*;

public class Hashtable1

{

public static void main(String args[])

{

Hashtable<Integer,String> hm=new Hashtable<Integer,String>();

hm.put(100,"Amit");

hm.put(102,"Ravi");

hm.put(101,"Vijay");

hm.put(103,"Rahul");

for(Map.Entry m:hm.entrySet())

{

System.out.println(m.getKey()+" "+m.getValue());

}

}

}

Output:

103 Rahul

102 Ravi

101 Vijay

100 Amit

Result: Thus, the above program was successfully executed without errors using HashTable

1. Write a program illustrating collections framework Stack

Aim: To write a program illustrating collections framework Stack

Description:

The stack is the subclass of Vector. It implements the last-in-first-out data structure, i.e., Stack. The stack contains all of the methods of the Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(Object o), which defines its properties.

Consider the following example.

Algorithm:

Step-1: Start the program.

Step-2: The method checks the stack is empty or not.

Step-3: The method pushes (insert) an element onto the top of the stack.

Step-4: The method removes an element from the top of the stack and returns the same element as the value of that function.

Step-5: The method looks at the top element of the stack without removing it.

Step-6: The method searches the specified object and returns the position of the object.

Step-7: Stop.

Program:

import java.util.\*;

public class TestJavaCollection4

{

public static void main(String args[])

{

Stack<String> stack = new Stack<String>();

stack.push("Ayush");

stack.push("Garvit");

stack.push("Amit");

stack.push("Ashish");

stack.push("Garima");

stack.pop();

Iterator<String> itr=stack.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

}

}

}

Output:

Ayush

Garvit

Amit

Ashish

Result: Thus, the above program was successfully executed without errors using Stack

Task-9

1. Develop an applet that displays a simple message.

Aim: To develop an applet that displays a simple message

Description:

Applet: Is a java program that is launched from HTML and runs in a web browser.

Or

Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a Web document.

How to create an applet in java?

1. In the navigator, select the project in which you want to create the new applet.
2. Choose File**then choose**New to open the new galley.
3. In the categories tree, expand Client Trier and select Swing/AWT.
4. In the items list, double-click Applet to open the new Applet dialog.

Advantage of Applet:

1. As applet is a small java program, it is platform-independent code that is capable to run on any browser.
2. Applets can perform various small tasks on client-side machines. They can play sounds, show images, get user inputs, get mouse clicks and even get user keystrokes, etc ...
3. Applets create and edit graphics on the client-side which are different and independent of the client-side platform.
4. As compared to stand-alone application applets that are small in size, the advantage of transferring it over the network makes it more usable.
5. Applets run on client browsers so they provide the functionality to import resources such as images, audio clips based on Url.
6. Applets are quite secure because of their access to resources.
7. Applets are secure and safe to use because they cannot perform any modifications to the local system.
8. Various small tasks such as performing login, inventory checking, task scheduling can be done by applets running over Intranets

t Swing/AWT.

### Drawback of Applet:

### The plugin is required at the client browser to execute the applet.

### Hierarchy of Applet:

### hierarchy of appletLifecycle of Java Applet:

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

Alogrithm:

Step-1: Start the program.

Step-2: import java.awt\*;

Step-3: import java.applet.\*;

Step-4: Create applet frame in under comment

* 1. /\*<applet code=weeka height=400 width=400> </applet>\*/

Step-5: Create Class Week extends Applet

Step-6: Set foreground(Color.yellow)

Step-7: public static void paint(Graphics g)

Step-8: g.drawString(“Hellow World”, size of text x, sizey)

Step-9: Stop

Program:**import java.applet.\*; import java.awt.\*;public class task9a extends Applet**

### {

### public void paint(Graphics g)

### {

### g.drawString("hello world",25,50);

### }

### }

### Ouput:

### D:\education\java\programs>javac task9a.java

### D:\education\java\programs>appletviewer task9a.java

### WhatsApp Image 2022-01-22 at 11.24.45 AM.jpeg

### Result: Thus, the above program was successfully executed without errors using applets that displays a simple message.

### b) Develop an applet that receives an integer in one text field and compute its factorial value and return it in another text field, when the button named “Compute” is clicked.

Aim: : Develop an applet that receives an integer in one text field and compute its factorial value and return it in another text field, when the button named “Compute” is clicked..

Description:

# Java AWT Text Field:

The [object](https://www.javatpoint.com/object-and-class-in-java) of a **Text Field** class is a text component that allows a user to enter a single line of text and edit it. It inherits **Text Component** class, which further inherits **the Component** class.

When we enter a key in the text field (as key pressed, key released or key typed), the event is sent to **Text Field**. Then the **Key Event** is passed to the registered **Key Listener**. It can also be done using **Action Event**; if the Action Event is enabled on the text field, then the Action Event may be fired by pressing the return key. The event is handled by the **Action Listener** interface.

Applets are designed to bring the web alive. they function to add animation sound and eventually complete multimedia into HTML documents.java is also part of the future of interfacing with virtual-reality environments implemented via VRML (**Virtual Reality Modeling Language**, pronounced vermal or by its initials, originally—before 1995—known as the Virtual Reality Markup Language). At present, java is limited only by the capabilities of the internet itself. applets are Java programs that are specialized for use over the Web. The Applet life cycle.

Algorithm:

Step-1: Start the program.

Step-2: The init()method is where your applet does much of its setup, such as defining its layout, parsing parameters, or setting the background colors.

Step-3: The start()method is used mainly when implementing threads in java.

Step-4: The stop() method is used to do what its name suggests: stop what is going on.

Step-5: when it is called, the applet is told to free up system resources.

Step-6: Stop.

Program:

package domain;

import java.applet.\*;

import java.awt.\*;

import javax.swing.\*;

/\*<applet code="task9b.class" height=200 width=200></applet>\*/

public class task9b extends Applet implements ActionListener

{

TextField t1,t2;

Label l1,l2;

Button b;

int a,fact;

public void init()

{

l1=new Label("enter a number");

t1= new TextField(5);

l2=new Label("Factorial of the given number is :");

t2=new TextField(10);

b=new Button("compute");

add(l1);

add(t1);

add(b);

add(l2);

add(t2);

b.addActionListener(this);

}

public void actionPerformed(ActionEvent e)

{

a=Integer.parseInt(t1.getText());

fact=1;

if(a<0)

{

t2.setText("wrong Input");

}

else

{

for(int i=a;i>1;i--)

fact\*=i;

t2.setText(" "+fact);

}

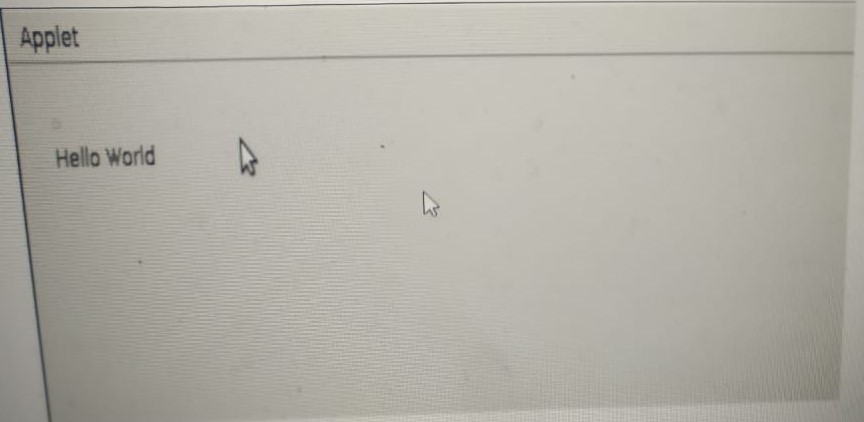
}

}

Output:

D:\education\java\programs>javac task9b.java

D:\education\java\programs>appletviewer task9b.java



Result: Thus, the above program was successfully executed without errors using an integer in one text field and compute its factorial value and return it in another text field

c) Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -, \*, % operations. Add a text field to display the result.

Aim: To write a java program that works as a simple calculator.

Description:

Grid Layout is one of the Layout managers. A layout manager automatically arranges your controls within a window by using some type of algorithm. Grid Layout lays out a component in a two-dimensional grid. When you instantiate a Grid Layout, you define the number of rows and columns.

Algorithm:

Step-1: Start the program.

Step-2: Add a component to the container.

Step-3: add actionListener for the specified component.

Step-4: Set the background color of the specified container.

Step-5: Set the size of the container to specified dimensions.

Step-6: Set the text of the label to s.

Step-7: returns the text of the label.

Step-8: Stop.

Program:

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

/\*

<applet code="Cal" width=300 height=300>

</applet>

\*/

public class Cal extends Applet implements ActionListener

{

String msg=" ";

int v1,v2,result;

TextField t1;

Button b[]=new Button[10];

Button add,sub,mul,div,clear,mod,EQ;

char OP;

public void init()

{

Color k=new Color(120,89,90);

setBackground(k);

t1=new TextField(10);

GridLayout gl=new GridLayout(4,5);

setLayout(gl);

for(int i=0;i<10;i++)

{

b[i]=new Button(""+i);

}

add=new Button("add");

sub=new Button("sub");

mul=new Button("mul");

div=new Button("div");

mod=new Button("mod");

clear=new Button("clear");

EQ=new Button("EQ");

t1.addActionListener(this);

add(t1);

for(int i=0;i<10;i++)

{

add(b[i]);

add(add);

add(sub);

add(mul);

add(div);

add(mod);

add(clear);

add(EQ);

for (int i=0;i<10;i++) {

b[i].addActionListener(this);

}

add.addActionListener(this);

sub.addActionListener(this);

mul.addActionListener(this);

div.addActionListener(this);

mod.addActionListener(this);

clear.addActionListener(this);

EQ.addActionListener(this);

}

public void actionPerformed(ActionEvent ae)

{

String str=ae.getActionCommand();

char ch=str.charAt(0);

if ( Character.isDigit(ch))

t1.setText(t1.getText()+str);

else if(str.equals("add"))

{

v1=Integer.parseInt(t1.getText());

OP='+';

t1.setText("");

}

else if(str.equals("sub"))

{

v1=Integer.parseInt(t1.getText());

OP='-';

t1.setText("");

}

else if(str.equals("mul"))

{

v1=Integer.parseInt(t1.getText());

OP='\*';

t1.setText("");

}

else if(str.equals("div"))

{

v1=Integer.parseInt(t1.getText());

OP='/';

t1.setText("");

}

else if(str.equals("mod"))

{

v1=Integer.parseInt(t1.getText());

OP='%';

t1.setText("");

}

if(str.equals("EQ"))

{

v2=Integer.parseInt(t1.getText());

if(OP=='+')

result=v1+v2;

else if(OP=='-')

result=v1-v2;

else if(OP=='\*')

result=v1\*v2;

else if(OP=='/')

result=v1/v2;

else if(OP=='%')

result=v1%v2;

t1.setText(""+result);

}

if(str.equals("clear"))

{

t1.setText("");

}

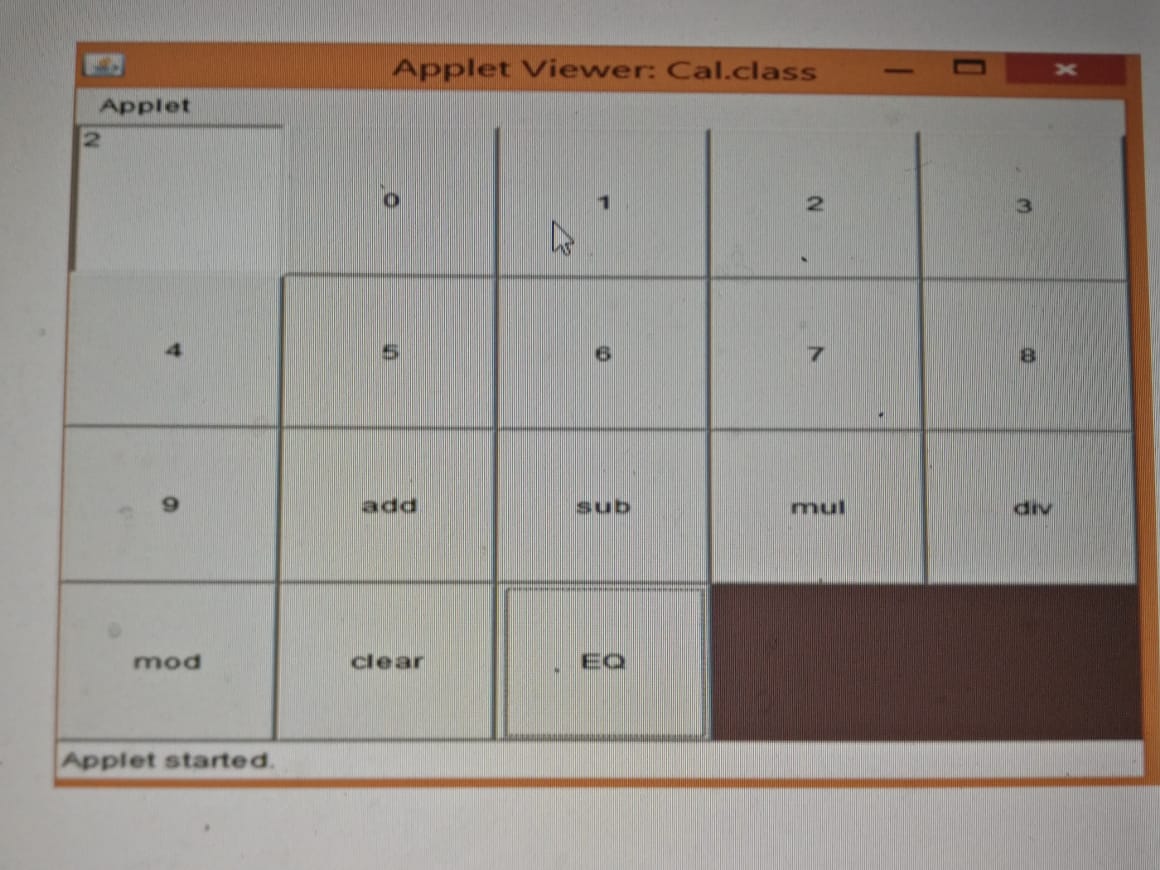
}

}

Output:

D:\education\java\programs>javac Cal.java

D:\education\java\programs>appletviewer Cal.java



Result: Thus, the above program was successfully executed without errors using that works as a simple calculator.

Task-10

1. Write a Java program for handling mouse events.

Aim: To write a Java program for handling mouse event.

Description:

Event: Is an object that is created when something changes within a graphical user interface.

In Java we can be handling Mouse events by implementing two interfaces they

1. Mouse listener interface
2. Mouse motion listener interface

So by implementing these two interfaces we can handle most events.

Mouse listener interface:

In Java, Mouse Listener is a class that gets notified when there is a change in the mouse state. Changes of the mouse can be pressing, clicking, and releasing. It can also be entering or exiting the window area. Mouse listener is working with the help of keyword implements and this listener interface can be gained from java. awt . event package.

It contains mainly five methods:

1. Mouse click: This method is used when an event occurs on clicking of moue on any of the specified components over the frame.

Syntax of this method:

public void mouseClicked(MouseEvent){}

1. Mouse pressed: This method is used when an event occurs on the long press of the mouse key on any of the specified components over the frame.

Syntax of this method:

public void mousePressed(MouseEvent e){}

1. Mouse released: This method is used when an event occurs on the release of the mouse key occurs after a long press on any of the specified components over the frame.

Syntax of this method:

public void mouseReleased(MouseEvent){}

1. Mouse entered: his method is used when an event occurs on entry of the mouse on any of the components over the Frame.

Syntax of this method:

public void mouseEntered(MouseEvent){}

1. Mouse exited: This method is used when an event occurs on exit of the mouse on any of the specified components over the Frame.

Syntax of this method:

public void mouseExited(MouseEvent){}

Mouse motion listener interface:

The Java MouseMotionListener is notified whenever you move or drag the mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java. awt. event package. It has two methods.

1. **public** **abstract** **void** mouseDragged(MouseEvent e);
2. **public** **abstract** **void** mouseMoved(MouseEvent e);

Algorithm:

Step-1: Start the program.

Step-2: Set is a collection of elements that can not contain duplicates. values.

Step-3: Set is implemented in HashSets, LinkedHashSets, TreeSet, etc.

Step-4: List is an ordered collection of elements that can have duplicates.

Step-5: Lists are classified into ArrayList, LinkedList, Vectors.

Step-6: FIFO approach, while instantiating the Queue interface you can either choose LinkedList or PriorityQueue.

Step-7: Deque(Double Ended queue) is used to add or remove elements from both the ends of the Queue(both head and tail).

Step-8: Map contains key-values pairs which don't have any duplicates. The map is implemented in HashMap, TreeMap, etc.

Step-9: Stop the program.

Program:

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

import java.applet.Applet;

/\*<applet code="Task10.class" width=400 height=350></Applet>\*/

public class Task10 extends Applet implements MouseListener

{

String msg="";

public void init()

{

addMouseListener(this);

}

public void mouseClicked(MouseEvent obj)

{

msg="mouse clicked";

repaint();

}

public void mouseEntered(MouseEvent obj)

{

msg="mouse entered";

repaint();

}

public void mouseExited(MouseEvent obj)

{

msg="mouse exit";

repaint();

}

public void mousePressed(MouseEvent obj)

{

msg="mouse pressed";

repaint();

}

public void mouseReleased(MouseEvent obj)

{

msg="mouse released";

repaint();

}

public void mouseMoved(MouseEvent obj)

{

msg="mouse moved";

repaint();

}

public void mouseDragged(MouseEvent obj)

{

msg="mouse Dragged";

repaint();

}

public void paint(Graphics g)

{

Font currentFont=new Font("TimesRoman",Font.BOLD,29);

g.setFont(currentFont);

//FontMetrics, Dimension class are used to display the text in themiddle of the

screen.

FontMetrics fm=g.getFontMetrics();

Dimension d= getSize();

int xc,yc;

xc=d.width/2- fm.stringWidth(msg)/2;

yc=d.height/2+fm.getDescent();

//g.drawString(msg,50,50);

g.drawString(msg,xc,yc);

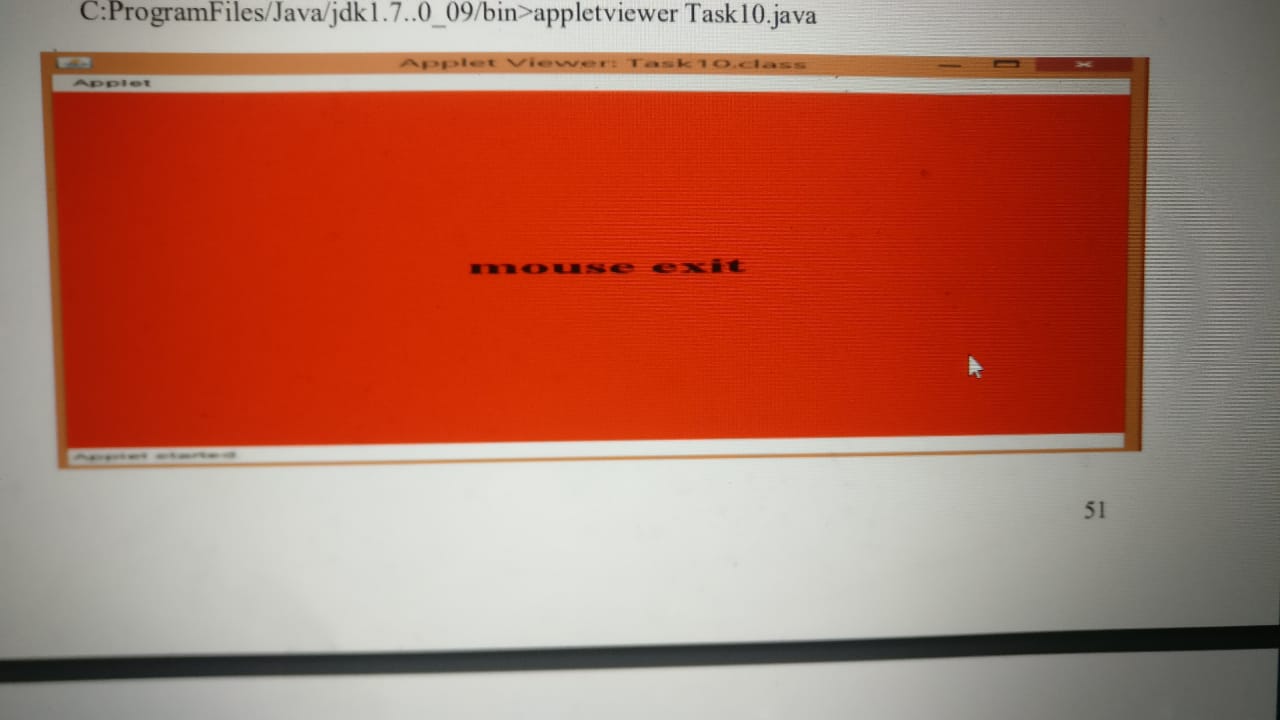
setBackground(Color.red);

}

}

Output:

C:ProgramFiles/Java/jdk1.7..0\_09/bin>javac Task10.java C:ProgramFiles/Java/jdk1.7..0\_09/bin>appletviewer Task10.java



Result: Thus, the above program was successfully executed without errors using that handling mouse events.

1. Write a Java program for handling key events.

Aim: To write a Java program for handling key events

Description:

Event handling three main components

Event: An event is a change in state in state of object.

Events Source: An event source is an object an event.

Listeners: A listener is an object that listens to the event. A listener gets notified when an event occurs.

**How are events handled?**

A source produces an event and sends it with the source to one or more listeners. Once the listener receives the event, the event will be processed and returned. Many Java packages, such as java.util, java.awt and java.awt.event, support events.

| **Event Classes** | **Description** | **Listener Interface** |
| --- | --- | --- |
| **ActionEvent** | generated when button is pressed, menu-item is selected, list-item is double clicked | ActionListener |
| **MouseEvent** | generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exits a component | MouseListener |
| **KeyEvent** | generated when input is received from keyboard | KeyListener |
| **ItemEvent** | generated when check-box or list item is clicked | ItemListener |
| **TextEvent** | generated when value of textarea or textfield is changed | TextListener |
| **MouseWheelEvent** | generated when mouse wheel is moved | MouseWheelListener |
| **WindowEvent** | generated when window is activated, deactivated, deiconified, iconified, opened or closed | WindowListener |
| **ComponentEvent** | generated when component is hidden, moved, resized or set visible | ComponentEventListener |
| **ContainerEvent** | generated when component is added or removed from container | ContainerListener |
| **AdjustmentEvent** | generated when scroll bar is manipulated | AdjustmentListener |
| **FocusEvent** | generated when component gains or loses keyboard focus | FocusListener |

Algorithm:

Step-1: Start the program.

Step-2: Write a Java program for handling key events.

Step-3: It is invoked when a key has been released.

Step-4: It is invoked when a key has been typed.

Step-5: Stop the program.

Program:

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

import java.applet.Applet;

/\*<applet code="Task10b.class" width=400 height=350></Applet>\*/

public class Task10b extends Applet implements KeyListener

{

String msg=" ";

public void init()

{

addKeyListener(this);

//Register Listener with Applet.requestFocus();

//requestFocus() method is present in the Component class. This method makes a request to obtain

the Input Focus for the current program.

//Otherwise program will not receive any keyboard events.

}

public void keyPressed(KeyEvent obj)

{

showStatus("Key Pressed");

}

public void keyReleased(KeyEvent obj)

{

showStatus("Key Released");

}

public void keyTyped(KeyEvent obj)

{

showStatus("Key Typed");

msg=msg+obj.getKeyChar();

repaint();

//getKeyChar() returns the key typed as a character.

//showStatus(String msg) method displays the message on the status bar of theapplet window.

}

public void paint(Graphics g)

{

Font currentFont=new Font("TimesRoman",Font.BOLD,29);

g.setFont(currentFont);

//FontMetrics, Dimension class are used to display the text in the middleof the screen

FontMetrics fm=g.getFontMetrics();

Dimension d= getSize();

int xc,yc;

xc=d.width/2-fm.stringWidth(msg)/2;

yc=d.height/2+fm.getDescent();

//g.drawString(msg,50,50);

g.drawString(msg,xc,yc);

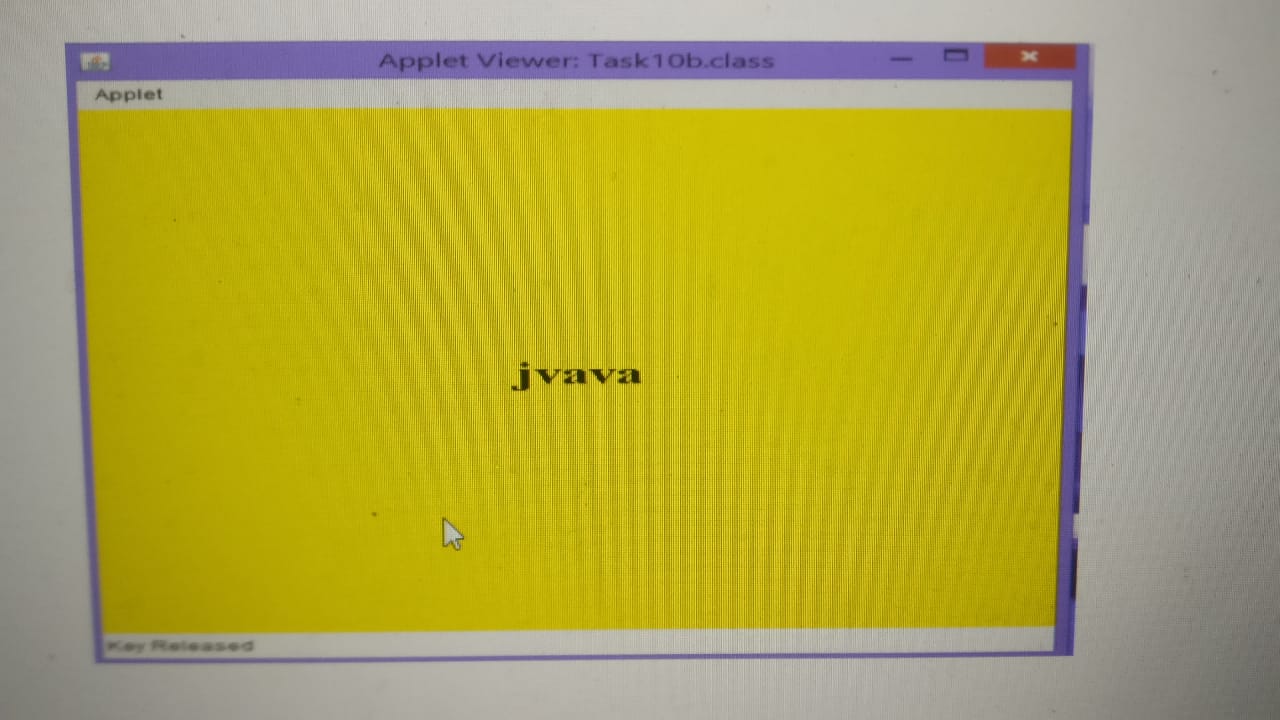
setBackground(Color.red);

}

}

Output:

C:ProgramFiles/Java/jdk1.7..0\_09/bin>javac Task10b.java C:ProgramFiles/Java/jdk1.7..0\_09/bin>appletviewer Task10b.html



Result: Thus, the above program was successfully executed without errors using that handling key events.

Task-11

1. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfields, Num1 and Num2. The division of Num1 and Num2 displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer,the program would throw a NumberFormatException. If Num2 were Zero, the program

would throw an ArithmeticException. Display the exception in a message dialog box.

Aim: To create a user interface for the division of two numbers

Description:

Create a user interface using swing components. First, create a class by extending the JFrame class of javax. swing package and implement the ActionListener interface to perform an action when the button is clicked on the frame. Create three text fields, a button, and a label and attach those components onto the getContentPane() of JFrame.

Attach action Listener to the button and provide action Performed() method to perform the operation division. When an exception is raised then display that information in a JOptionPane.

Algorithm:

Step-1: Start the program.

Step-2: An interface in Java is a template of a class. It has static

constants and abstract methods.

Step-3: Java has a mechanism to achieve interface abstraction.

Step-4: The Java interface can only contain abstract methods, not

method entities.

Step-5: It is used to achieve abstraction and multiple inheritances in

Java.

Step-6: Stop the program.

Program:

import java.awt.\*;

import javax.swing.\*;

import java.applet.Applet;

import java.awt.event.\*;

public class Task11 extends Applet implements ActionListener

{

TextField t1,t2,t3;

Button b;

Label L1,L2,L3,L4;

String s;

Task11 e;

public void init()

{

e=this;

//setLayout(new GridLayout(3,2));

t1=new TextField(5);

t2=new TextField(5);

t3=new TextField(5);

L1=new Label("enter num1");

L2=new Label("enter num2");

L3=new Label("Result is");

L4=new Label("Division of 2 numbers");

b=new Button("Divide");

add(L4);

add(L1);

add(t1);

add(L2);

add(t2);

add(L3);

add(t3);

add(b);

b.addActionListener(this);

}

public void paint(Graphics g)

{

setBackground(Color.red);

setForeground(Color.black);

L1.setBackground(Color.blue);

L2.setBackground(Color.blue);

L3.setBackground(Color.blue);

L4.setBackground(Color.green);

//L4.setAlignment(Label.CENTER);

b.setBackground(Color.cyan);

}

public void actionPerformed(ActionEvent ae)

{

try

{

int num1=Integer.parseInt(t1.getText());

int num2=Integer.parseInt(t2.getText());

s=""+(num1/num2);

t3.setText(s);

}

catch(ArithmeticException a)

{

JOptionPane.showMessageDialog(null,"Divide by zero");

}

catch(NumberFormatException b)

{

JOptionPane.showMessageDialog(null,"Number FormateException");

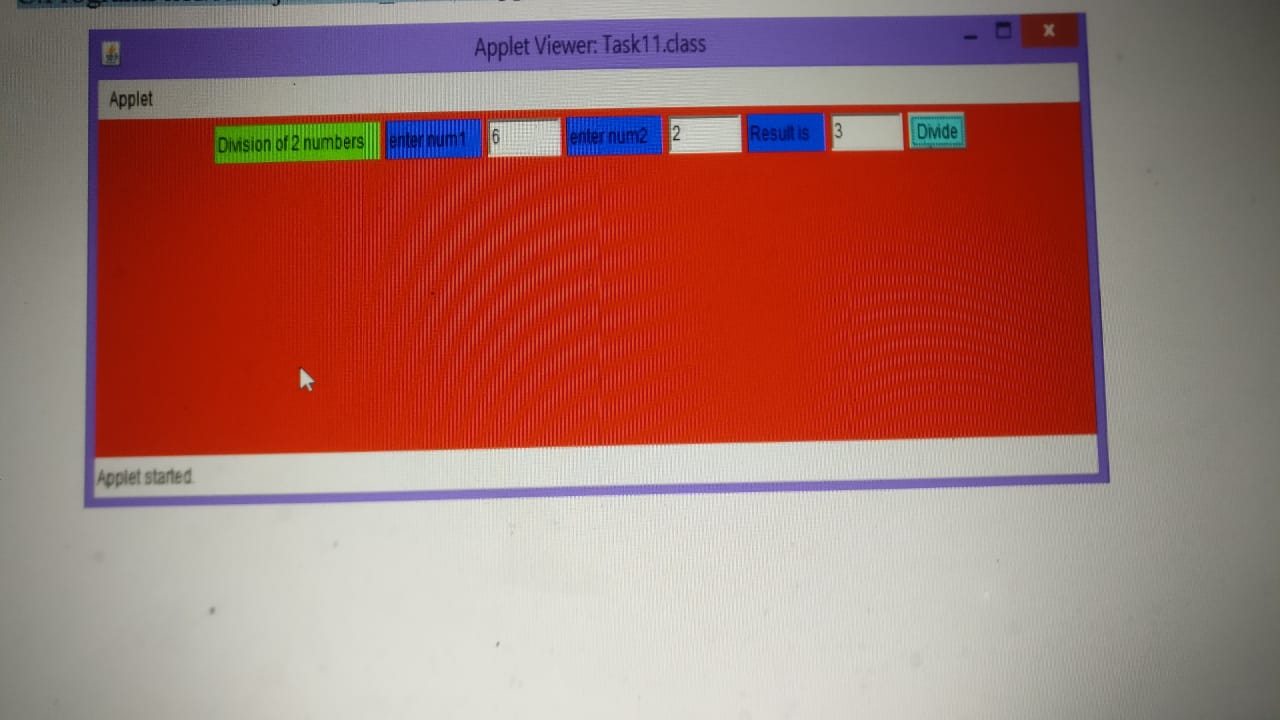
}

}

}

Output:

C:ProgramFiles/Java/jdk1.7..0\_09/bin>javac Task11.java C:ProgramFiles/Java/jdk1.7..0\_09/bin>appletviewer Task11.java



Result: Thus , the above program was successfully executed without errors using that displaying the exception in a message dialog box.

Task-12

1. Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green. When a radio button is selected, the light is turned on, and only one light can be on at a time. No light is on when the program starts.

Aim: To write a java program that simulates a traffic light.

Description:

Create an applet by the name TrafficLight. Use CheckBox and CheckBoxGroup controls to allow a user to select one of the options. Initialize the controls and add them to the Applet in the init() method.

Implement ItemListener’s interface method ItemStateChanged to respond to user interaction. To do this call the repaint() method inside the ItemStateChanged method. This will call the paint method to apply changes requested by the user. In the paint, the method gets the color selected by the user and applies the color to ovals which represent traffic bulbs.

Algorithm:

Step-1: Start the program.

Step-2: Create a Button Group such that it contains three radio buttons Red, Yellow, Green when red is clicked you want to display “Stop”

Step-3: when orange is clicked you want to display “ready”.

Step-4: When the green button is clicked Display “Go” .

Step-5: Create a sample Traffic signal demo using rectangle and ovals using 2D graphics.

Step-6: Stop.

Program:

import java.awt.\*;

import javax.swing.\*;

import java.applet.Applet;

import java.awt.event.\*;

public class Task11 extends Applet implements ActionListener

{

TextField t1,t2,t3;

Button b;

Label L1,L2,L3,L4;

String s;

Task11 e;

public void init()

{

e=this;

//setLayout(new GridLayout(3,2));

t1=new TextField(5);

t2=new TextField(5);

t3=new TextField(5);

L1=new Label("enter num1");

L2=new Label("enter num2");

L3=new Label("Result is");

L4=new Label("Division of 2 numbers");

b=new Button("Divide");

add(L4);

add(L1);

add(t1);

add(L2);

add(t2);

add(L3);

add(t3);

add(b);

b.addActionListener(this);

}

public void paint(Graphics g)

{

setBackground(Color.red);

setForeground(Color.black);

L1.setBackground(Color.blue);

L2.setBackground(Color.blue);

L3.setBackground(Color.blue);

L4.setBackground(Color.green);

//L4.setAlignment(Label.CENTER);

b.setBackground(Color.cyan);

}

public void actionPerformed(ActionEvent ae)

{

try

{

int num1=Integer.parseInt(t1.getText());

int num2=Integer.parseInt(t2.getText());

s=""+(num1/num2);

t3.setText(s);

}

catch(ArithmeticException a)

{

JOptionPane.showMessageDialog(null,"Divide by zero");

}

catch(NumberFormatException b)

{

JOptionPane.showMessageDialog(null,"Number FormateException");

}

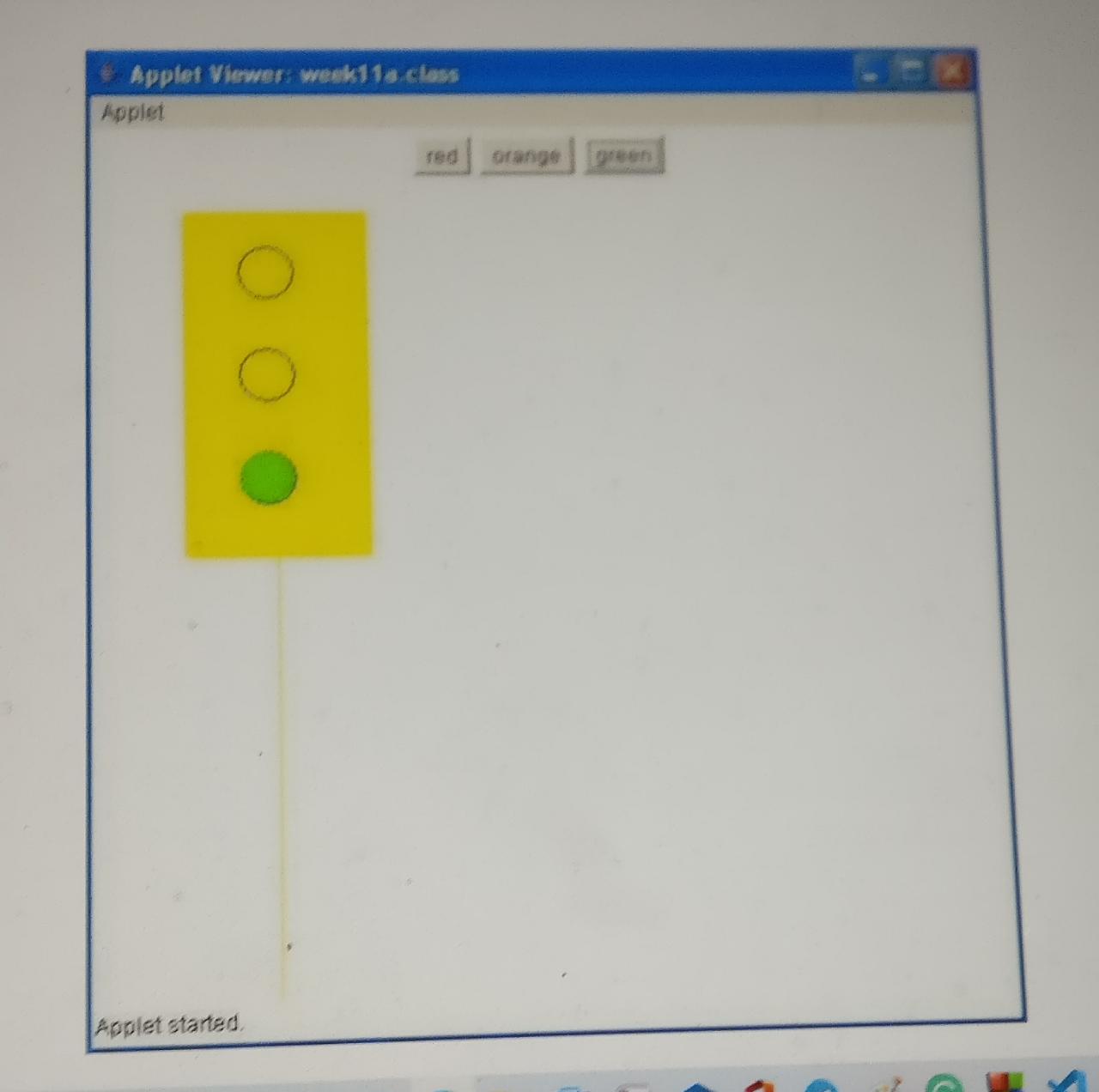
}

}

Output:

Javac task12a.java

Appletviewer task12a.java



Result: Thus, the above program was successfully executed without errors using that simulates a traffic light.

1. Write a Java program that allows the user to draw lines, rectangles and ovals.

Aim: To write a Java program that allows the user to draw lines, rectangles, and ovals.

Description:

In this tutorial, we are going to see how to draw lines, rectangles, and circles in JFrame. Java offers us an easy way to draw graphics using Graphics class in the AWT package which allows us to draw primitive geometric types like lines, circles, etc… This tutorial explains the different functions of the Graphics class used to draw shapes.

Draw a line:

Graphics class provides the **Graphics. draw line(int x1, int y1, int x2, int y2)** method to draw a line on the screen. While x1 is the x-coordinate of the first point in the line and y1 is the y-coordinate of the first point in the line. Similarly, x2 and y2 are the co-ordinates of the second point in the line.

(x1,y1) (x2,y2)

P1 p2

Draw a rectangle:

Graphics class provides the **Graphics.drawRect(int x, int y, int width, int height)** method for drawing a rectangle or square. The first two parameters specify the starting point and the last two parameters specify the width and height of the rectangle or square. The width and height of the square must be identical.

(x,y)

Height

width

Draw a Oval:

**drawOval**(int x, int y, int height, int width); This method will draw an oval at specified x and y position with given height and width. g2. fillOval(int x, int y, int height, int width); This method will fill an oval at specified x and y position with given height and width.

Algorithm:

Step-1: Start the program.

Step-2: drawLine( int x1, int y1, int x2, int y2 ), Draws a line between the point (x1, y1) and the point (x2, y2).

Step-3: drawRect( int x, int y, int width, int height),

Draws a rectangle of the specified width and height. The top-left corner of the rectangle has the coordinates (x, y). Only the outline of the rectangle is drawn using the Graphics object's color the body of the rectangle is not filled with this color.

Step-4: fillRect( int x, int y, int width, int height ),

Draws a filled rectangle with the specified width and height. The top-left corner of the rectangle has the coordinate (x, y). The rectangle is filled with the Graphics object's color.

Step-5: clearRect( int x, int y, int width, int height ),

Draws a filled rectangle with the specified width and height in the current background color. The top-left corner of the rectangle has the coordinate (x, y). This method is useful if the programmer wants to remove a portion of an image.

Step-6: drawRoundRect( int x, int y, int width, int height, int arcWidth, int arc height ),

Draws a rectangle with rounded corners in the current color with the specified width and height. The arcWidth and arc height determine the rounding of the corners. Only the outline of the shape is drawn.

Step-6: fillRoundRect( int x, int y, int width, int height, int arcWidth, int arc height )

Draws a filled rectangle with rounded corners in the current color with the specified width and height. The arcWidth and arc height determine the rounding of the corners.

Step-7: draw3DRect( int x, int y, int width, int height, boolean b ),

Draws a three-dimensional rectangle in the current color with the specified width and height. The top-left corner of the rectangle has the coordinates (x, y). The rectangle appears raised when b is true and lowered when b is false. Only the outline of the shape is drawn.

Step-8: fill3DRect( int x, int y, int width, int height, boolean b )

Draws a filled three-dimensional rectangle in the current color with the specified width and height. The top-left corner of the rectangle has the coordinates (x, y). The rectangle appears raised when b is true and lowered when b is false.

Step-9: drawOval( int x, int y, int width, int height ),

Draws an oval in the current color with the specified width and height. The bounding rectangle's top-left corner is at the coordinates (v, y). The oval touches all four sides of the bounding rectangle at the center of each side .

Step-10: fillOval( int x, int y, int width, int height ),

Draws a filled oval in the current color with the specified width and height. The bounding rectangle's top-left corner is at the coordinates (x, y). The oval touches all four sides of the bounding rectangle at the center of each side.

Step-11: Stop.

Program:

import java.applet.\*;

import java.awt.event.\*;

import java.awt.\*;

/\*<applet code="task12b.class" height=310 width=400> </applet>\*/

public class task12b extends Applet implements ActionListener

{

Button b[]=new Button[10];

int in;

public void init()

{

b[0]=new Button("Line");

b[1]=new Button("Rectangle");

b[2]=new Button("FilledRectangle");

b[3]=new Button("RoundedRectangle");

b[4]=new Button("FilledRoundedRectangle");

b[5]=new Button("Oval");

b[6]=new Button("FilledOval");

b[7]=new Button("Arc");

b[8]=new Button("FilledArc");

b[9]=new Button("Polygon");

for(int i=0;i<10;i++)

{

add(b[i]);

b[i].addActionListener(this);

}

}

public void actionPerformed(ActionEvent ae)

{

for(int j=0;j<10;j++)

{

if(ae.getSource()==b[j])

{

in=j+1;

break;

}

}

repaint();

}

public void paint(Graphics g)

{

if(in==1)

g.drawLine(150, 150, 250,300);

if(in==2)

g.drawRect(150, 150, 60,50);

if(in==3)

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

if(in==4)

g.drawRoundRect(150, 150, 60, 50, 15, 15);

if(in==5)

g.fillRoundRect(150, 150, 60, 50, 15, 15);

if(in==6)

g.drawOval(150, 150, 60, 50);

if(in==7)

g.fillOval(150, 150, 60,50);

if(in==8)

g.drawArc(150, 150, 60, 50, 0, 75);

if(in==9)

g.fillArc(150, 150, 60, 50, 0,75);

if(in==10)

{

int xpoints[ ] = {50, 200,250, 250,200};

int ypoints[ ] = {250, 200,250,300,300};

int num = 5;

g.drawPolygon(xpoints, ypoints, num);

}

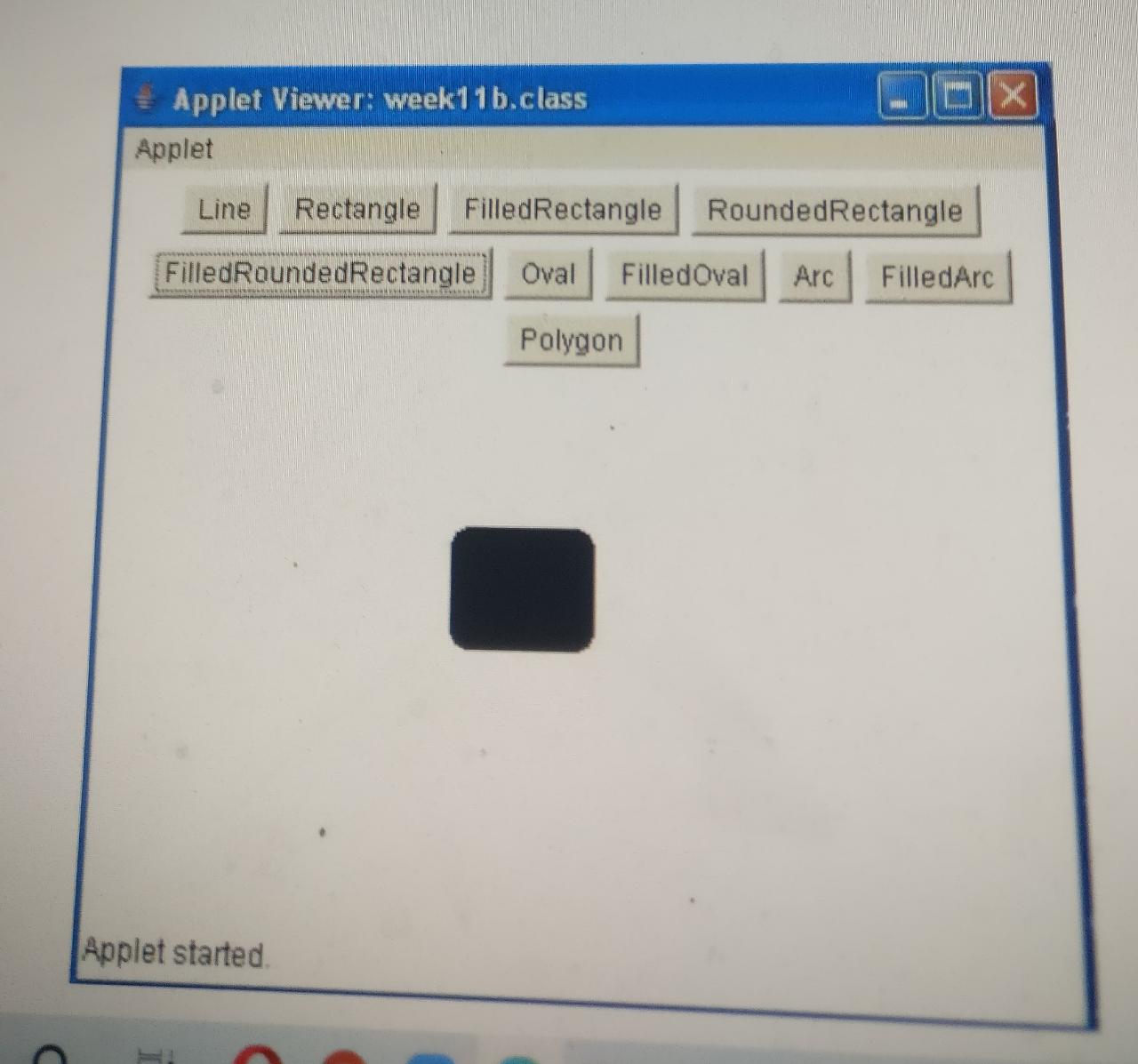
}

}

Output:

D:\>javac task12b.java

D:\>appletviewer task12b.java



Result: Thus, the above program was successfully executed without errors using that draw lines, rectangles, and ovals