Experiment No. 1

AIM: Implement a java program to calculate gross salary, net salary taking the following data. Input:- empno, empname, ba Process: DA=70% of basic HRA=30% of basic CCA=Rs240/- PF=10% of basic PT= Rs100/-

ALGORITHM:

1)Start

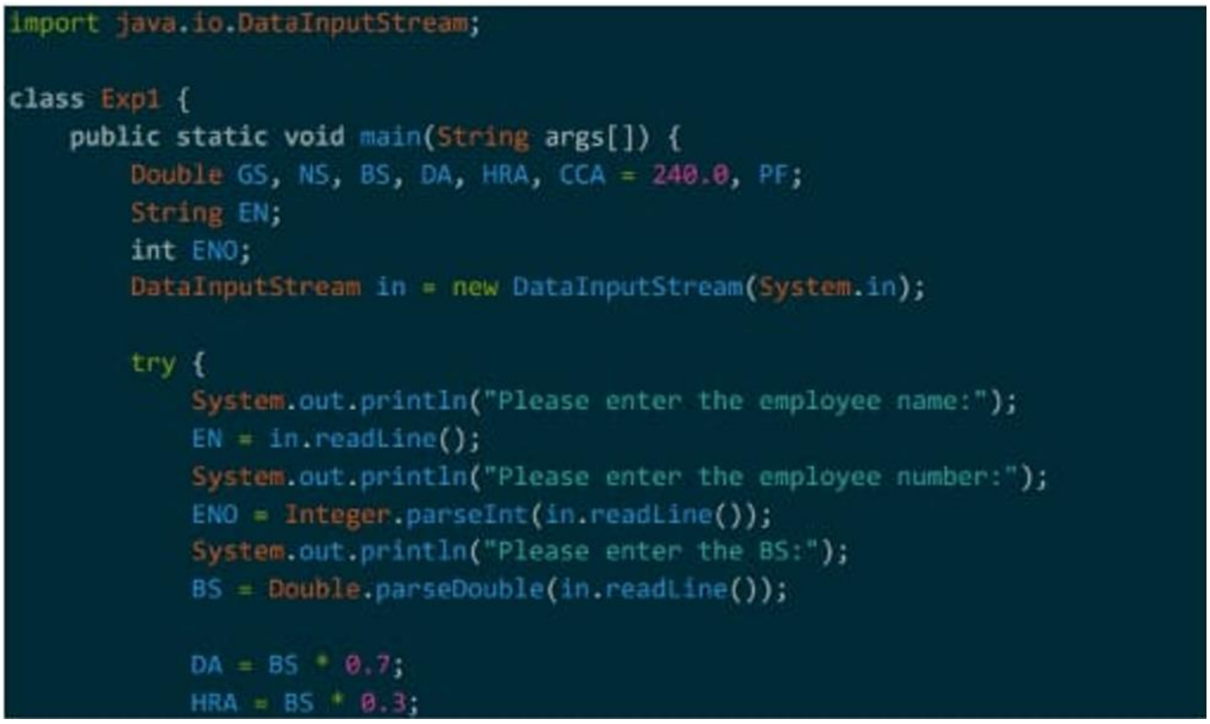
2)Take the input from the user as employee name, id and basic salary

3)Calculate the output from the above parameters DA, HRA, GS, income tax and net salary

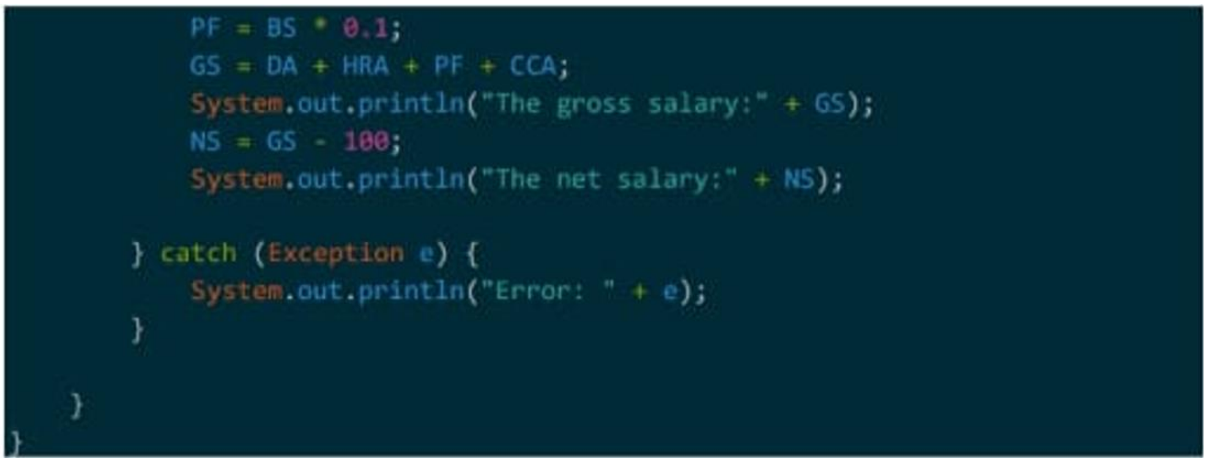
4)Display the output

5)Stop

Source Code:

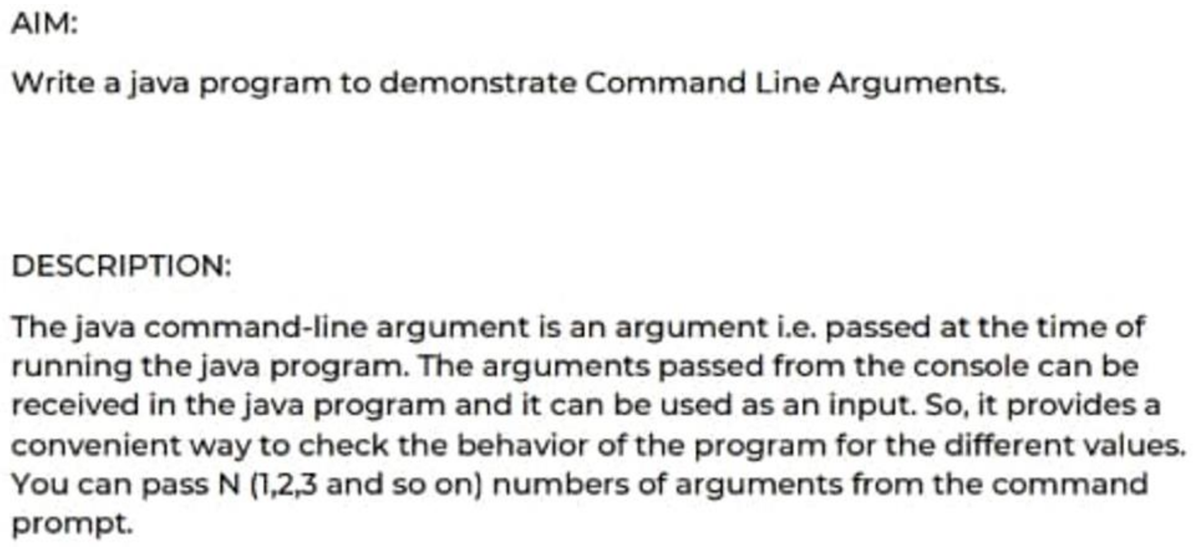


```java
import java.io.DataInputStream;

class Exp1 {
    public static void main(String args[]) {
        Double GS, NS, BS, DA, HRA, CCA = 240.0, PF;
        String EN;
        int ENO;
        DataInputStream in = new DataInputStream(System.in);

        try {
            System.out.println("Please enter the employee name:");
            EN = in.readLine();
            System.out.println("Please enter the employee number:");
            ENO = Integer.parseInt(in.readLine());
            System.out.println("Please enter the BS:");
            BS = Double.parseDouble(in.readLine());

            DA = BS * 0.7;
            HRA = BS * 0.3;
```



```java
            PF = BS * 0.1;
            GS = DA + HRA + PF + CCA;
            System.out.println("The gross salary:" + GS);
            NS = GS - 100;
            System.out.println("The net salary:" + NS);

        } catch (Exception e) {
            System.out.println("Error: " + e);
        }

    }
}
```

Experiment No. 2



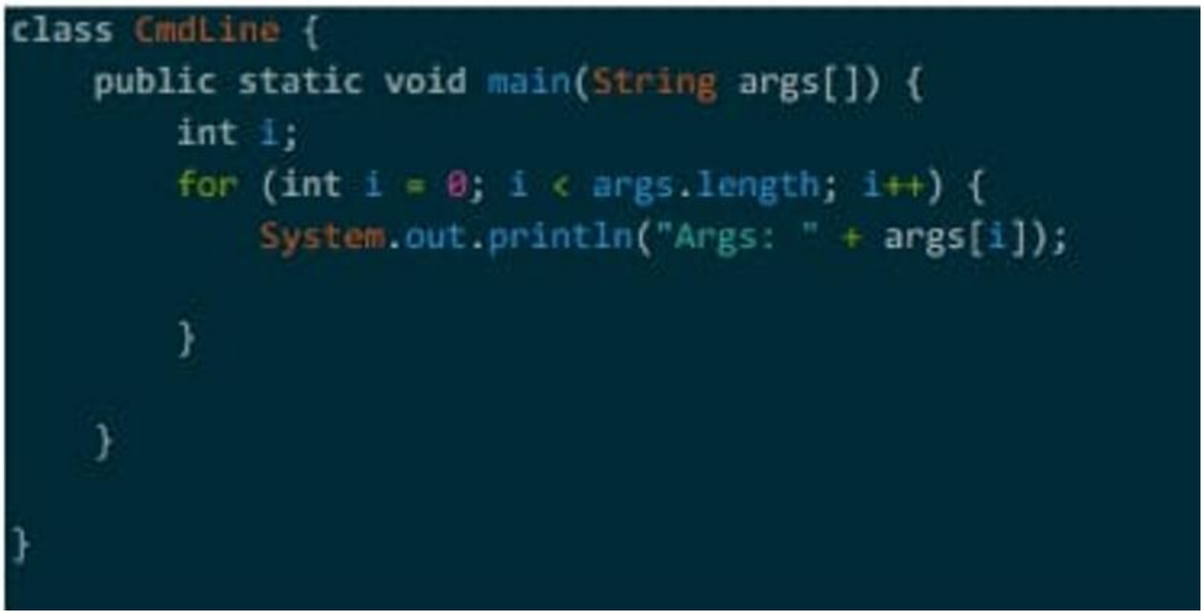
AIM:

Write a java program to demonstrate Command Line Arguments.

DESCRIPTION:

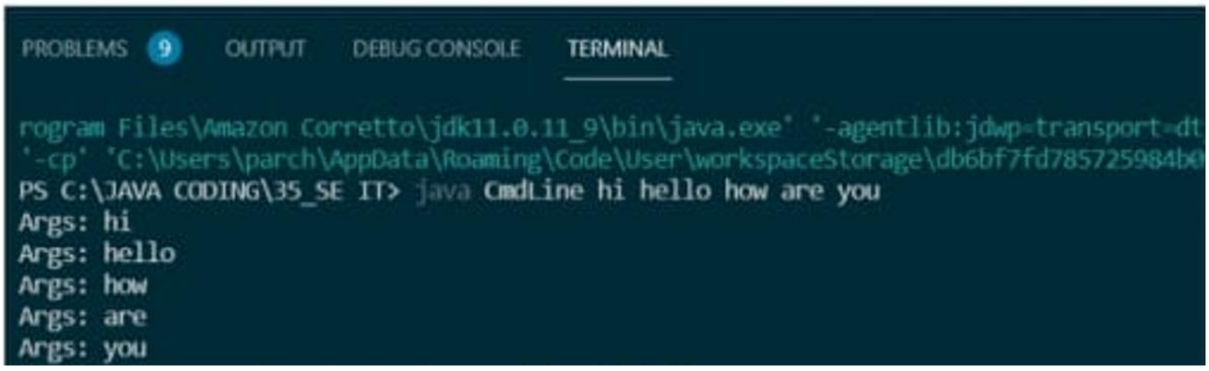
The java command-line argument is an argument i.e. passed at the time of running the java program. The arguments passed from the console can be received in the java program and it can be used as an input. So, it provides a convenient way to check the behavior of the program for the different values. You can pass N (1,2,3 and so on) numbers of arguments from the command prompt.

Source Code:



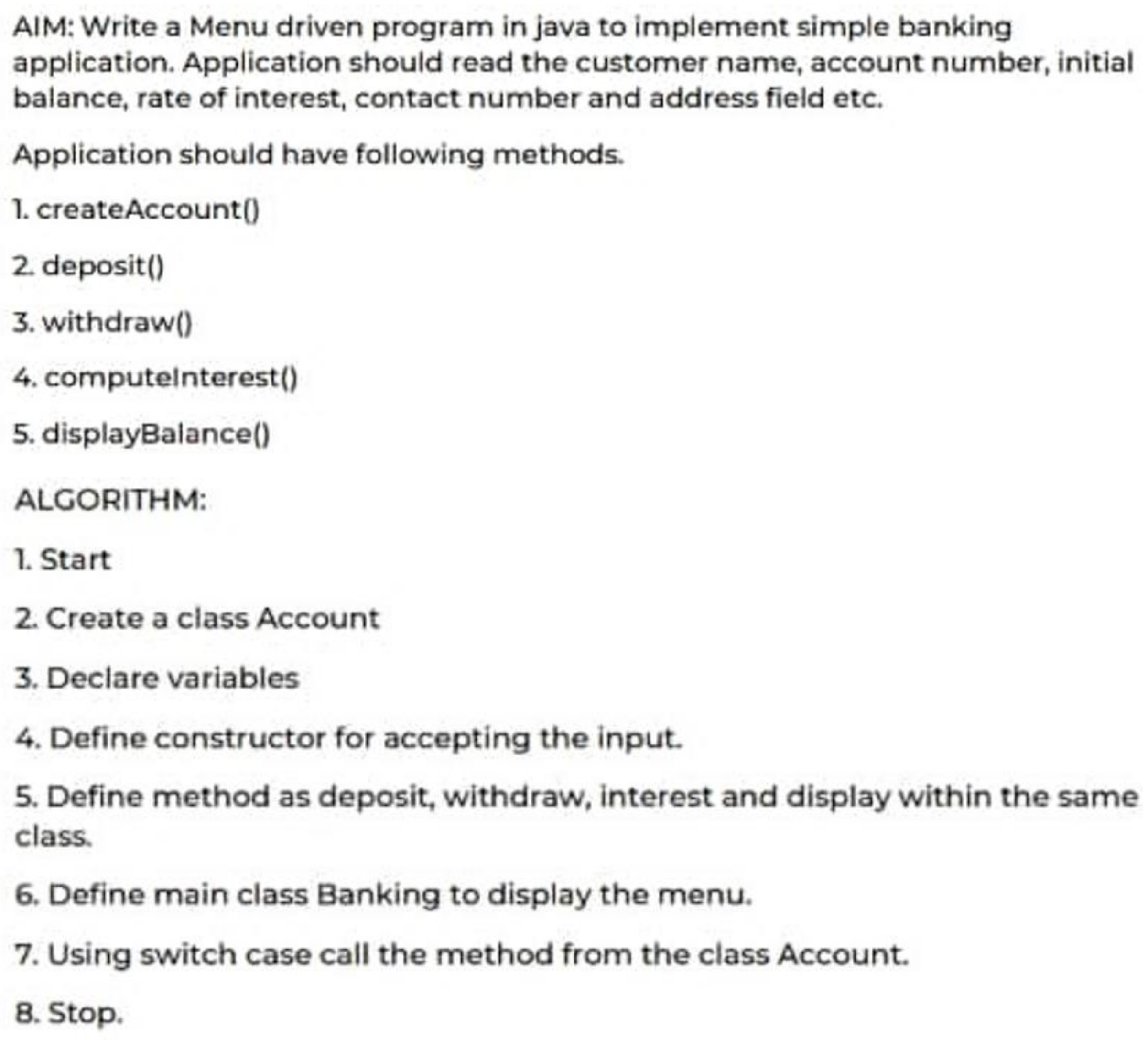
```java
class CmdLine {
    public static void main(String args[]) {
        int i;
        for (int i = 0; i < args.length; i++) {
            System.out.println("Args: " + args[i]);

        }

    }

}
```

Output:



```
rogram Files\Amazon Corretto\jdk11.0.11_9\bin\java.exe' '-agentlib:jdwp=transport=dt
'-cp' 'C:\Users\parch\AppData\Roaming\Code\User\workspaceStorage\db6bf7fd785725984b0
PS C:\JAVA CODING\35_SE IT> java CmdLine hi hello how are you
Args: hi
Args: hello
Args: how
Args: are
Args: you
```

Experiment No. 3



AIM: Write a Menu driven program in java to implement simple banking application. Application should read the customer name, account number, initial balance, rate of interest, contact number and address field etc.

Application should have following methods.

1. createAccount()

2. deposit()

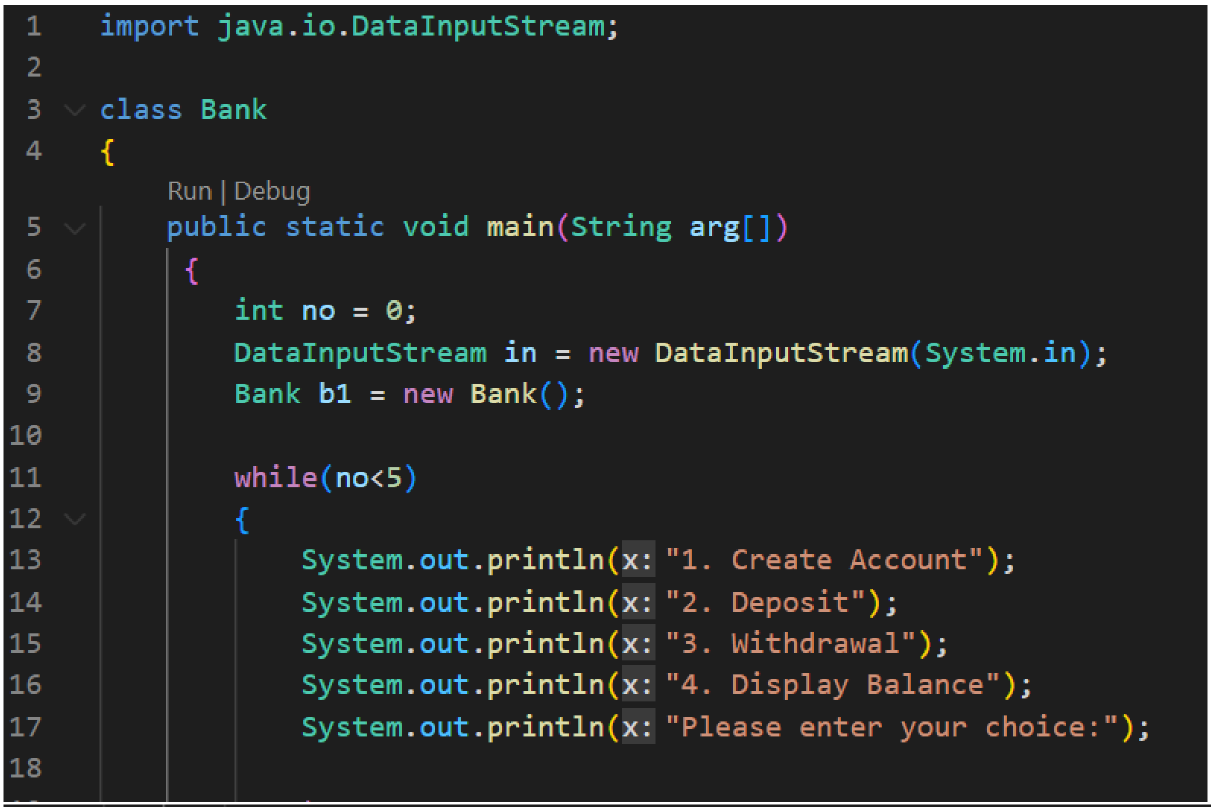
3. withdraw()

4. computeInterest()

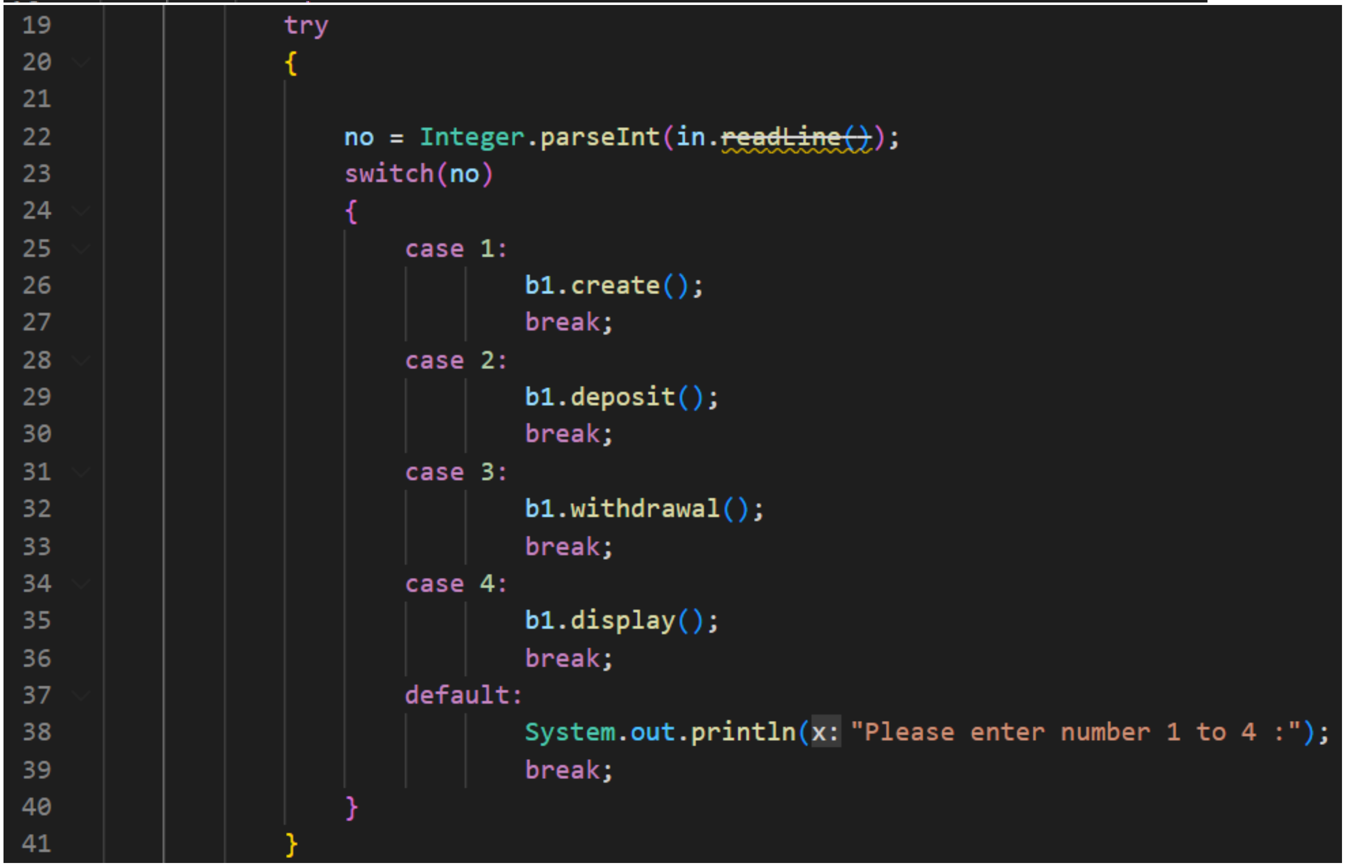
5. displayBalance()

ALGORITHM:

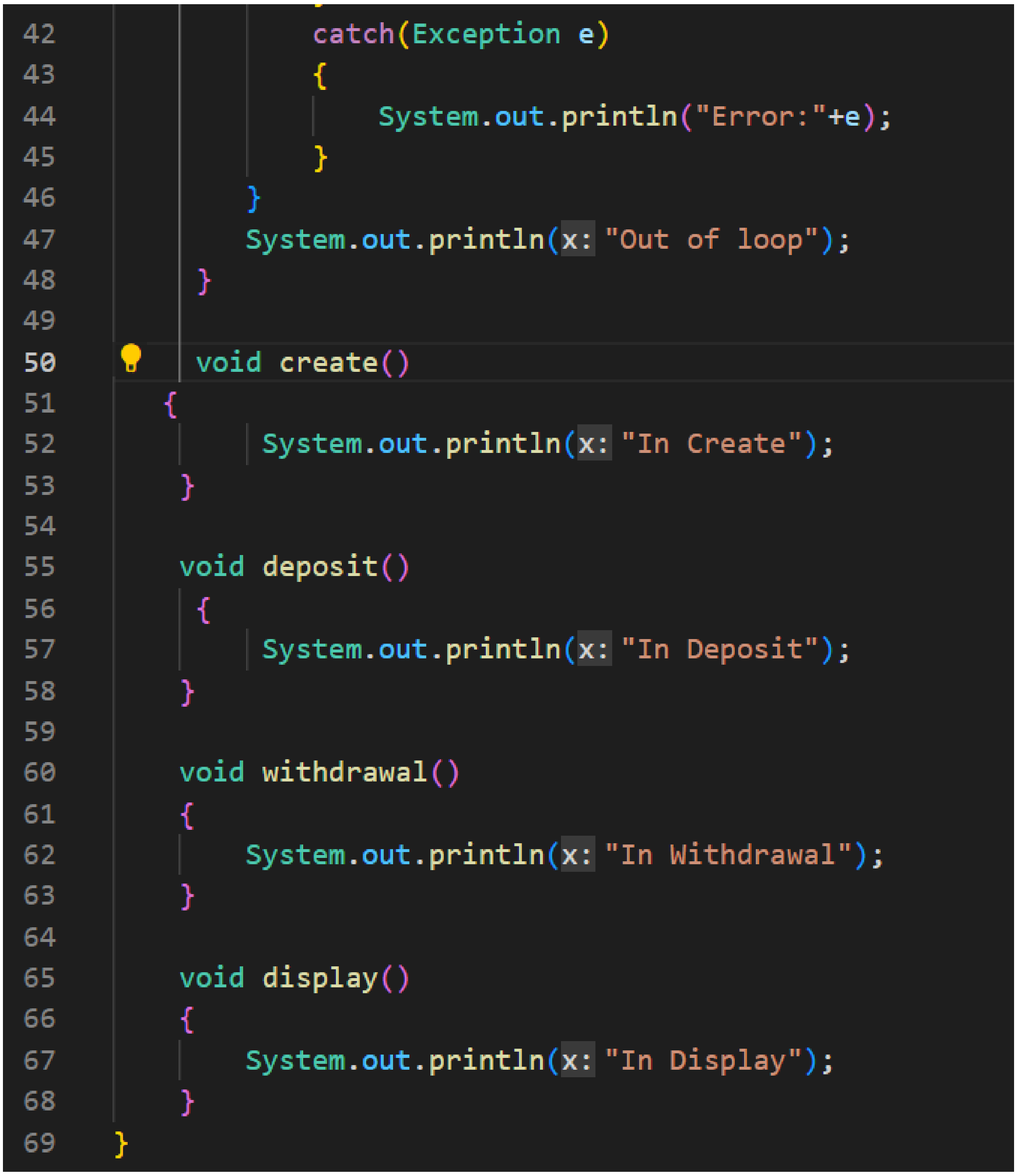
```
1. Start
2. Create a class Account
3. Declare variables
4. Define constructor for accepting the input.
5. Define method as deposit, withdraw, interest and display within the same class.
6. Define main class Banking to display the menu.
7. Using switch case call the method from the class Account.
8. Stop.
```



```java
import java.io.DataInputStream;

class Bank
{
    public static void main(String arg[])
    {
        int no = 0;
        DataInputStream in = new DataInputStream(System.in);
        Bank b1 = new Bank();

        while(no<5)
        {
            System.out.println(x: "1. Create Account");
            System.out.println(x: "2. Deposit");
            System.out.println(x: "3. Withdrawal");
            System.out.println(x: "4. Display Balance");
            System.out.println(x: "Please enter your choice:");

```

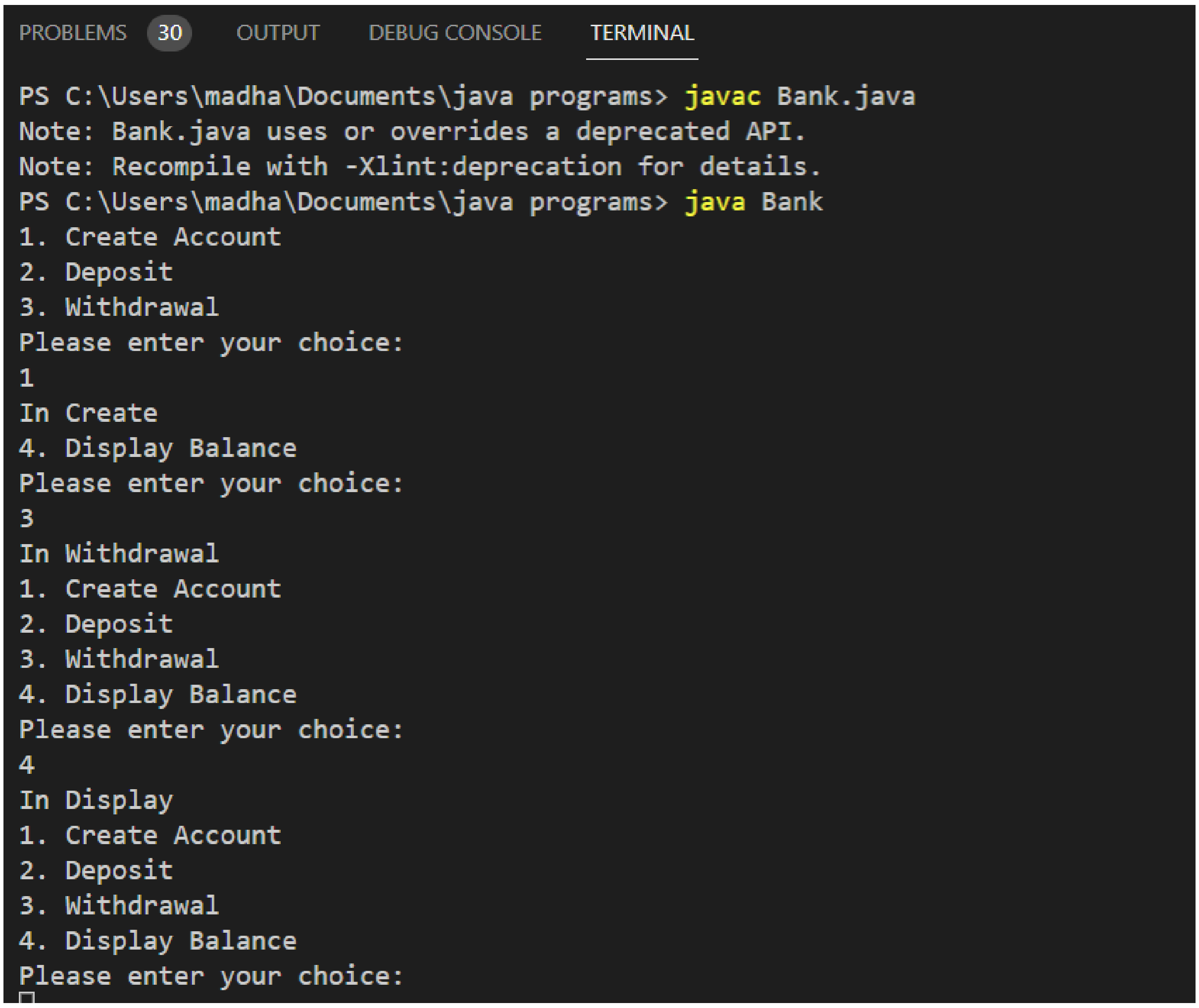


```java
            try
            {

                no = Integer.parseInt(in.readLine());
                switch(no)
                {
                    case 1:
                            b1.create();
                            break;
                    case 2:
                            b1.deposit();
                            break;
                    case 3:
                            b1.withdrawal();
                            break;
                    case 4:
                            b1.display();
                            break;
                    default:
                            System.out.println(x: "Please enter number 1 to 4 :");
                            break;
                }
            }
```



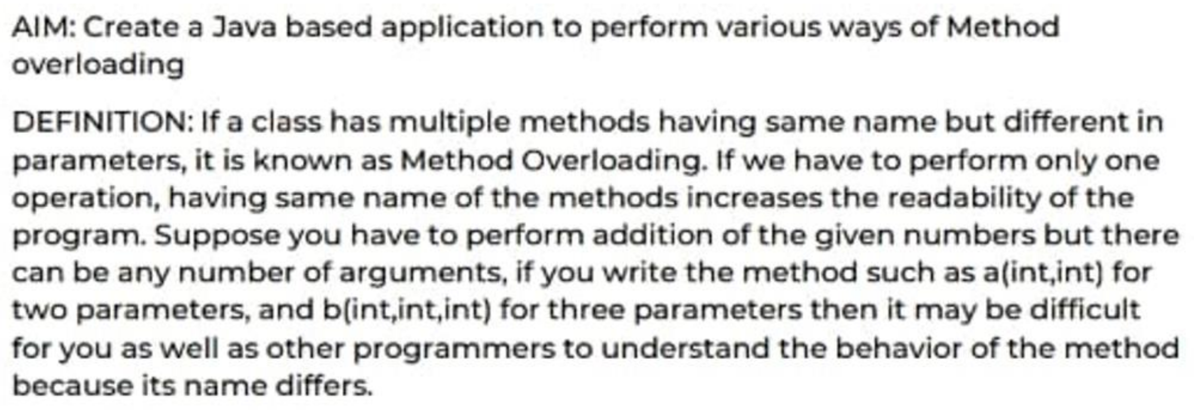
```
            catch(Exception e)
            {
                System.out.println("Error:"+e);
            }
        }
        System.out.println(x: "Out of loop");
     }

     void create()
   {
         System.out.println(x: "In Create");
    }

    void deposit()
     {
         System.out.println(x: "In Deposit");
    }

    void withdrawal()
    {
        System.out.println(x: "In Withdrawal");
    }

    void display()
    {
        System.out.println(x: "In Display");
    }
}
```

Output:



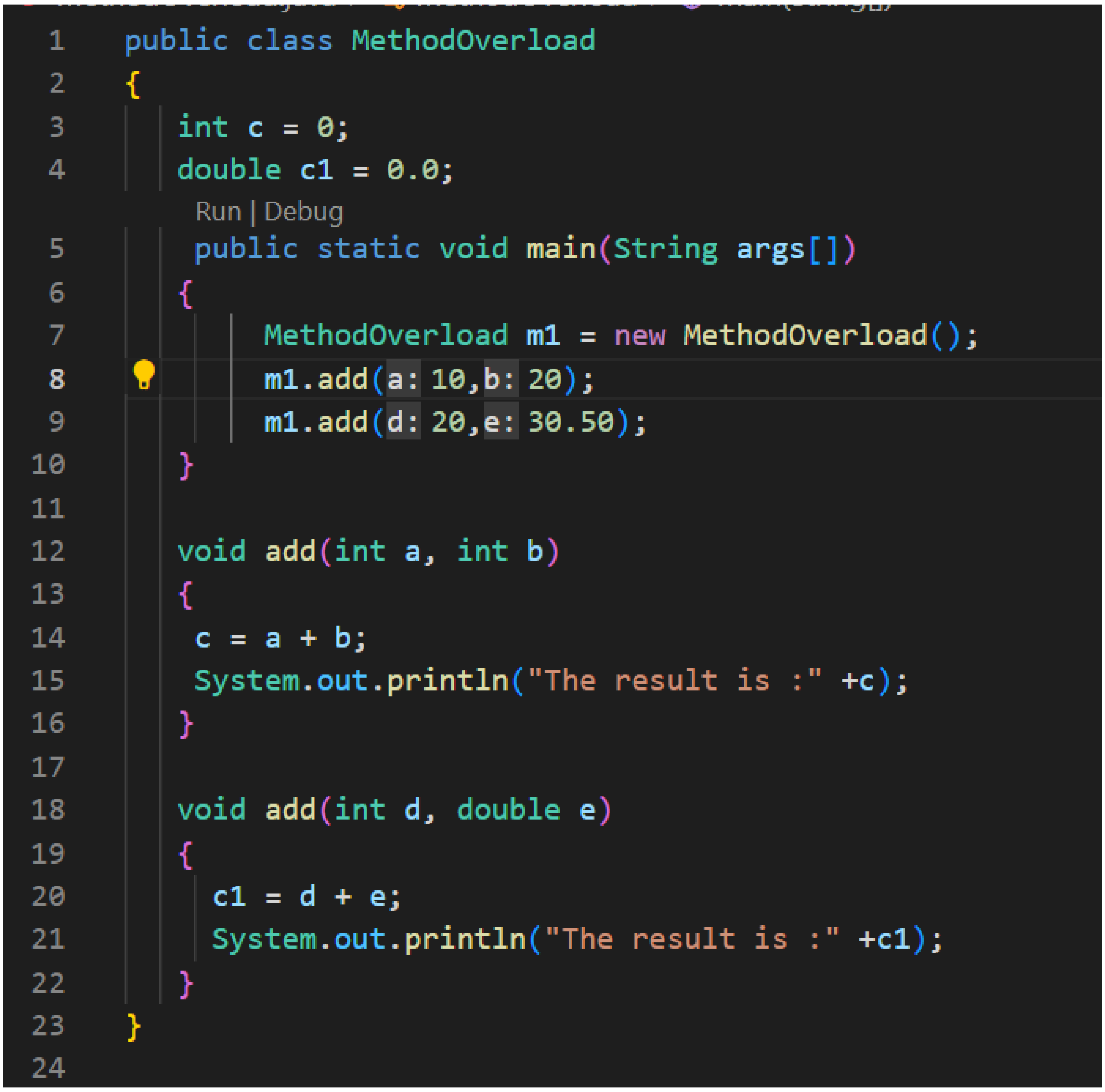
```
PS C:\Users\madha\Documents\java programs> javac Bank.java
Note: Bank.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
PS C:\Users\madha\Documents\java programs> java Bank
1. Create Account
2. Deposit
3. Withdrawal
Please enter your choice:
1
In Create
4. Display Balance
Please enter your choice:
3
In Withdrawal
1. Create Account
2. Deposit
3. Withdrawal
4. Display Balance
Please enter your choice:
4
In Display
1. Create Account
2. Deposit
3. Withdrawal
4. Display Balance
Please enter your choice:
```

Experiment No. 4


AIM: Create a Java based application to perform various ways of Method overloading

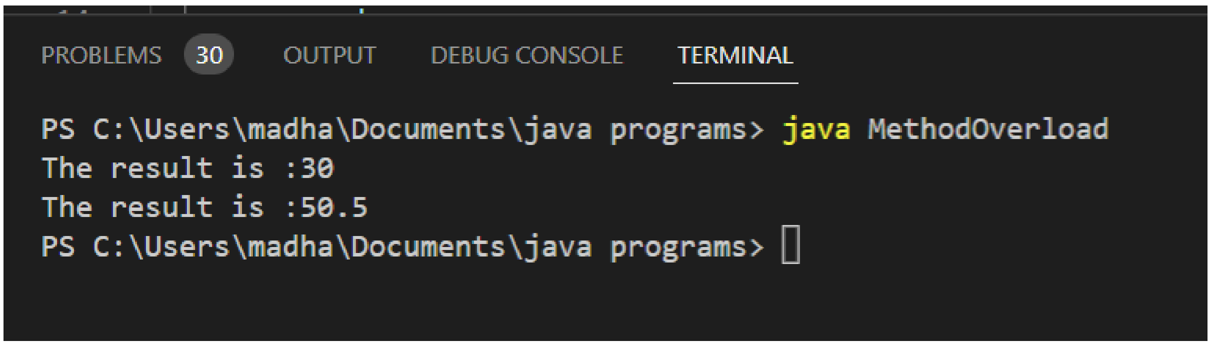
DEFINITION: If a class has multiple methods having same name but different in parameters, it is known as Method Overloading. If we have to perform only one operation, having same name of the methods increases the readability of the program. Suppose you have to perform addition of the given numbers but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

Source Code:



```java
public class MethodOverload
{
    int c = 0;
    double c1 = 0.0;
     Run | Debug
     public static void main(String args[])
    {
         MethodOverload m1 = new MethodOverload();
         m1.add(a: 10,b: 20);
         m1.add(d: 20,e: 30.50);
    }

    void add(int a, int b)
    {
     c = a + b;
     System.out.println("The result is :" +c);
    }

    void add(int d, double e)
    {
      c1 = d + e;
      System.out.println("The result is :" +c1);
    }
}

```

Output:



```
PS C:\Users\madha\Documents\java programs> java MethodOverload
The result is :30
The result is :50.5
PS C:\Users\madha\Documents\java programs> 
```

Experiment No. 5

Aim: Write a java program to illustrate Constructor chaining.

Description: constructor chaining is a process of calling one constructor from another constructor with respect to current object.

Algorithm:

1: start.

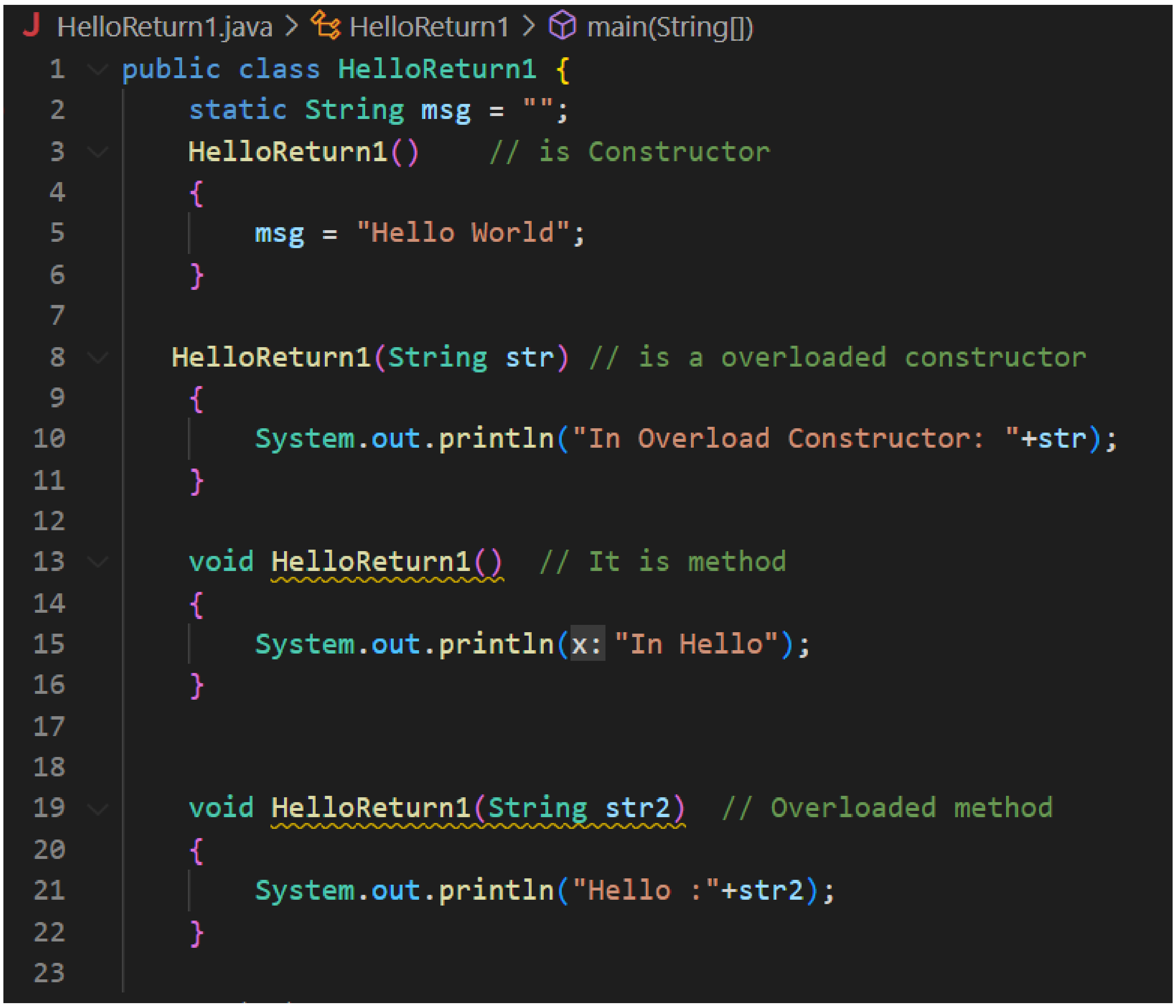
2: create class main.

3: create one constructor of main class.

4: again one constructor create of main class.

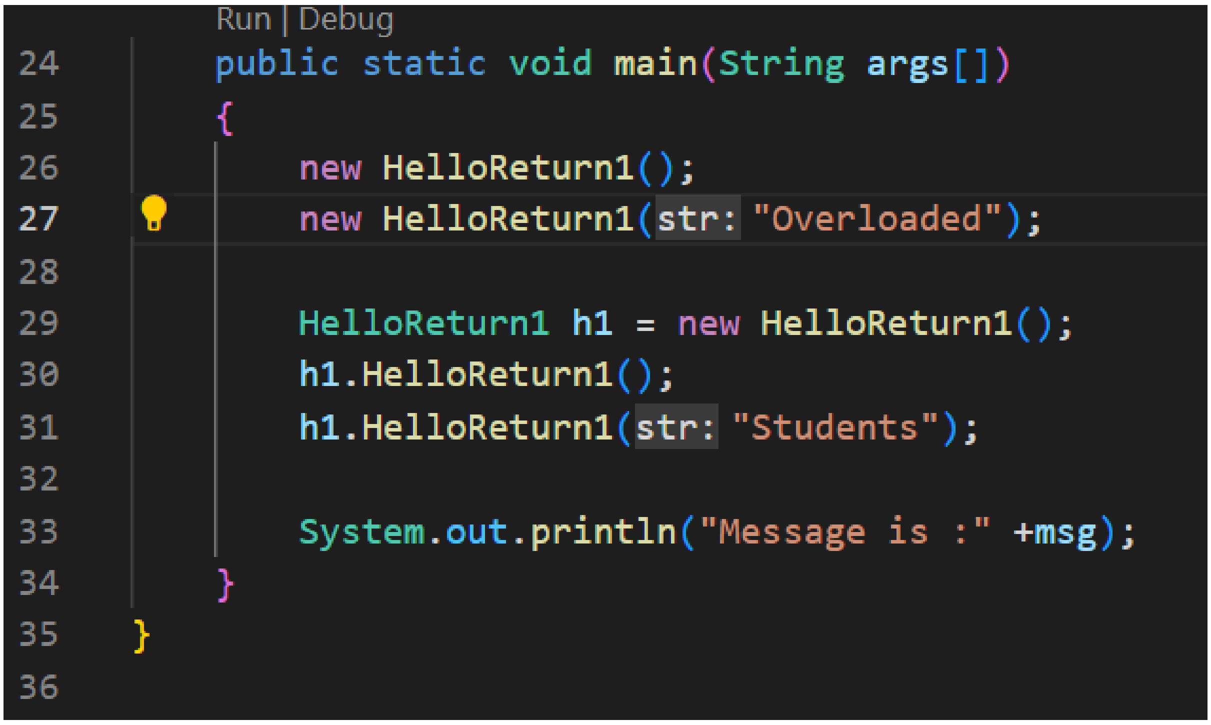
5: create a one method name of class.

6: create main method.



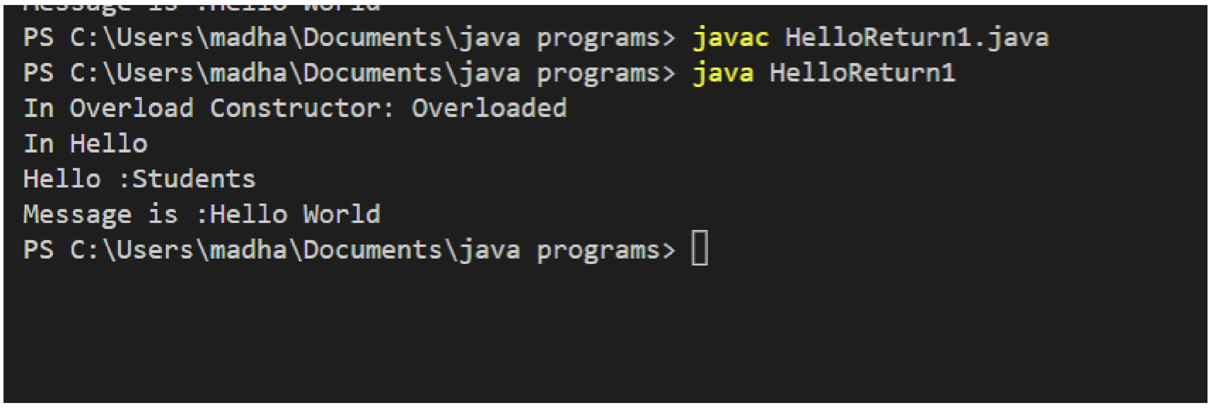
HelloReturn1.java > HelloReturn1 > main(String[])

```java
public class HelloReturn1 {
    static String msg = "";
    HelloReturn1()    // is Constructor
    {
        msg = "Hello World";
    }

   HelloReturn1(String str) // is a overloaded constructor
    {
        System.out.println("In Overload Constructor: "+str);
    }

    void HelloReturn1()   // It is method
    {
        System.out.println(x: "In Hello");
    }


    void HelloReturn1(String str2)  // Overloaded method
    {
        System.out.println("Hello :"+str2);
    }

```



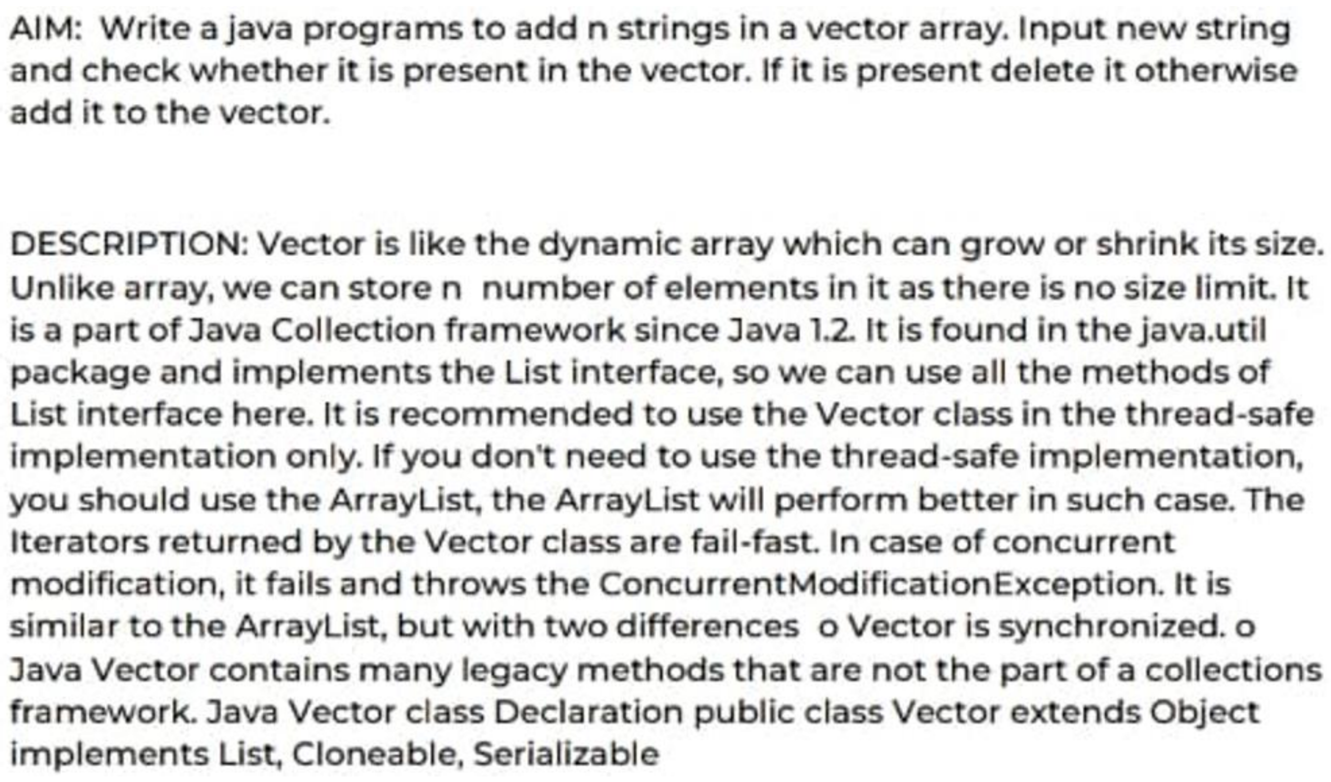
```java
Run | Debug
public static void main(String args[])
{
    new HelloReturn1();
    new HelloReturn1(str: "Overloaded");

    HelloReturn1 h1 = new HelloReturn1();
    h1.HelloReturn1();
    h1.HelloReturn1(str: "Students");

    System.out.println("Message is :" +msg);
}
}
```

Output:



```
PS C:\Users\madha\Documents\java programs> javac HelloReturn1.java
PS C:\Users\madha\Documents\java programs> java HelloReturn1
In Overload Constructor: Overloaded
In Hello
Hello :Students
Message is :Hello World
PS C:\Users\madha\Documents\java programs> 
```

Experiment No. 6



AIM: Write a java programs to add n strings in a vector array. Input new string and check whether it is present in the vector. If it is present delete it otherwise add it to the vector.

DESCRIPTION: Vector is like the dynamic array which can grow or shrink its size. Unlike array, we can store n number of elements in it as there is no size limit. It is a part of Java Collection framework since Java 1.2. It is found in the java.util package and implements the List interface, so we can use all the methods of List interface here. It is recommended to use the Vector class in the thread-safe implementation only. If you don't need to use the thread-safe implementation, you should use the ArrayList, the ArrayList will perform better in such case. The Iterators returned by the Vector class are fail-fast. In case of concurrent modification, it fails and throws the ConcurrentModificationException. It is similar to the ArrayList, but with two differences o Vector is synchronized. o Java Vector contains many legacy methods that are not the part of a collections framework. Java Vector class Declaration public class Vector extends Object implements List, Cloneable, Serializable

Algorithm:

1: start

2: import java.util.*.

3: create class Exp6 and define main method there.

4: create 3 object of the vector

5: add different string and integer in vector object

6: add one another string

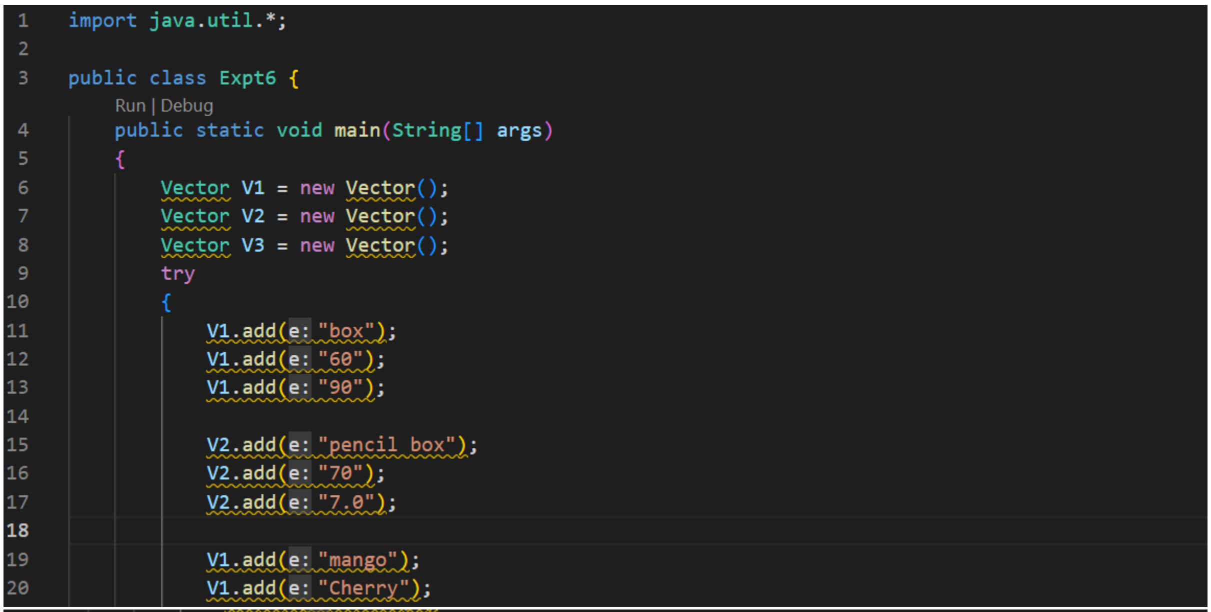
7: print the vector and first element.

8: 3rd object of the vector is empty and we add in the one string.

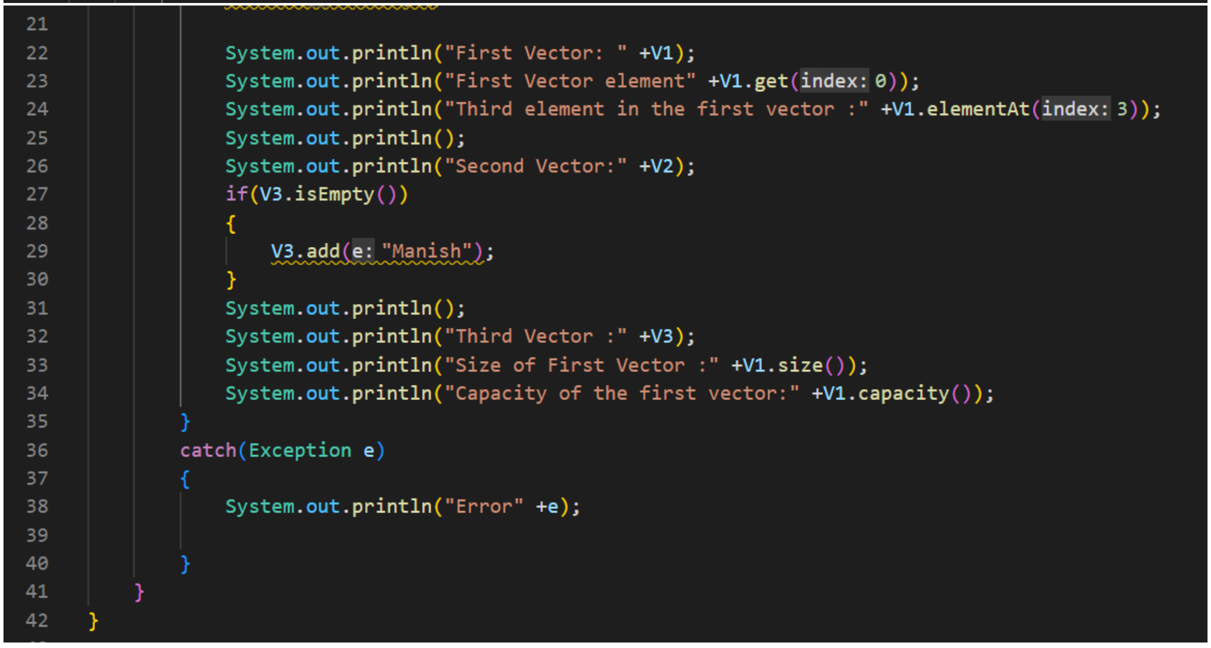
9: and print that string.

10: print sixe and the capacity of the first vector

Source Code:

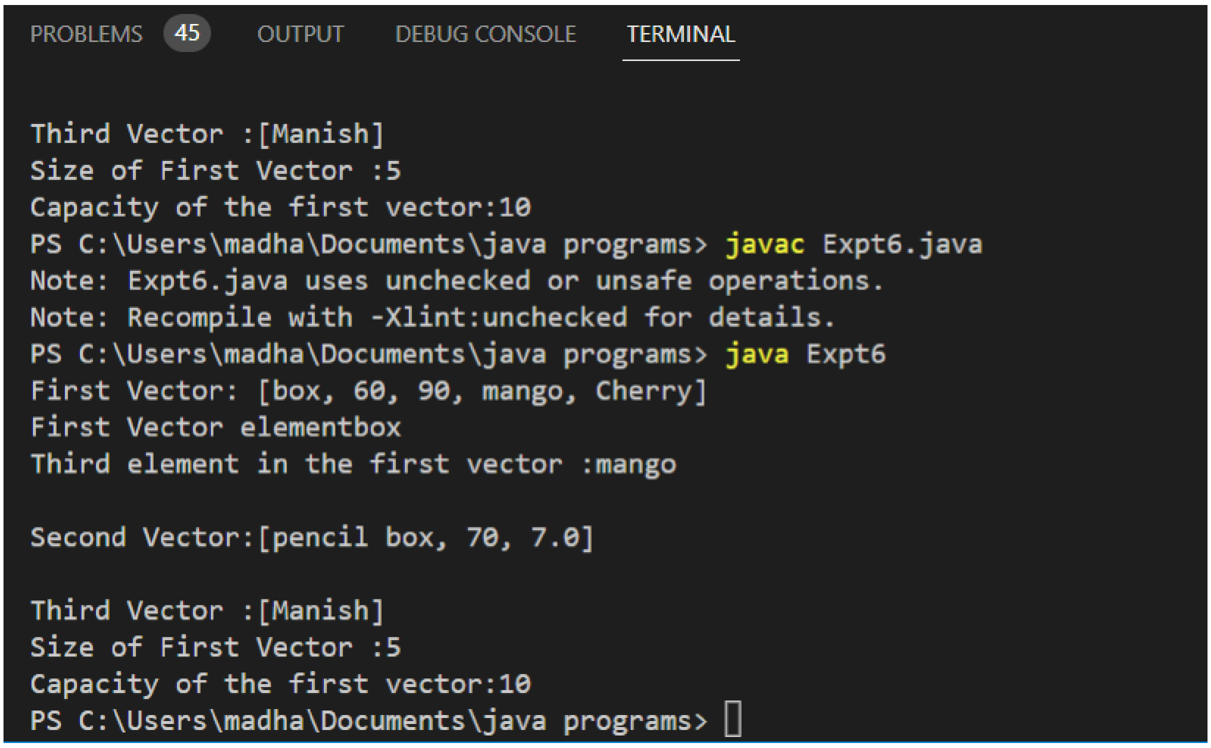


```java
import java.util.*;

public class Expt6 {
    public static void main(String[] args)
    {
        Vector V1 = new Vector();
        Vector V2 = new Vector();
        Vector V3 = new Vector();
        try
        {
            V1.add(e: "box");
            V1.add(e: "60");
            V1.add(e: "90");

            V2.add(e: "pencil box");
            V2.add(e: "70");
            V2.add(e: "7.0");

            V1.add(e: "mango");
            V1.add(e: "Cherry");
```



```java

            System.out.println("First Vector: " +V1);
            System.out.println("First Vector element" +V1.get(index: 0));
            System.out.println("Third element in the first vector :" +V1.elementAt(index: 3));
            System.out.println();
            System.out.println("Second Vector:" +V2);
            if(V3.isEmpty())
            {
                V3.add(e: "Manish");
            }
            System.out.println();
            System.out.println("Third Vector :" +V3);
            System.out.println("Size of First Vector :" +V1.size());
            System.out.println("Capacity of the first vector:" +V1.capacity());
        }
        catch(Exception e)
        {
            System.out.println("Error" +e);

        }
    }
}
```

Output:



```
Third Vector :[Manish]
Size of First Vector :5
Capacity of the first vector:10
PS C:\Users\madha\Documents\java programs> javac Expt6.java
Note: Expt6.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\madha\Documents\java programs> java Expt6
First Vector: [box, 60, 90, mango, Cherry]
First Vector elementbox
Third element in the first vector :mango

Second Vector:[pencil box, 70, 7.0]

Third Vector :[Manish]
Size of First Vector :5
Capacity of the first vector:10
PS C:\Users\madha\Documents\java programs>
```

Experiment No. 7

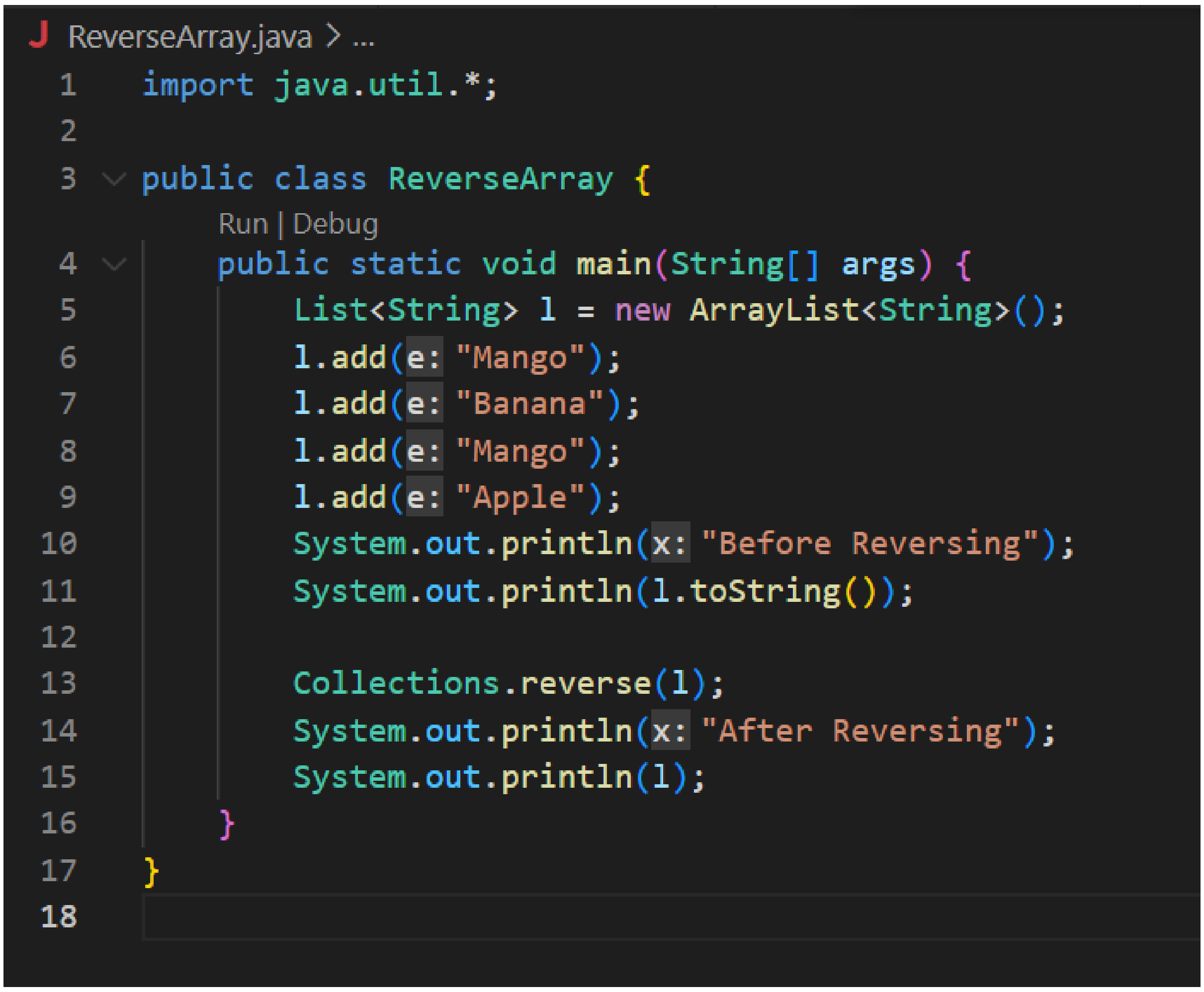
Aim: Print the reverse array list in java writing our own function.

Description: reverseArrayList() method in reverseList class cantains logic for reversing an arrayList with integer object.

Algorithm:

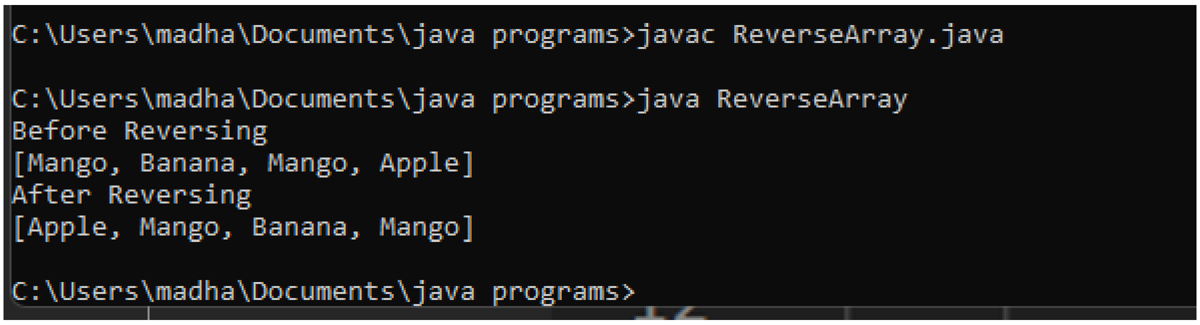
```
1: start
2:create class ReverseList
3: define main method
4: difine collection of ArrayList of the string data type.
5: add some string type.
6: and using collections method we reverse the string list and print it.
```

Source Code:



```java
ReverseArray.java > ...
import java.util.*;

public class ReverseArray {
    Run | Debug
    public static void main(String[] args) {
        List<String> l = new ArrayList<String>();
        l.add(e: "Mango");
        l.add(e: "Banana");
        l.add(e: "Mango");
        l.add(e: "Apple");
        System.out.println(x: "Before Reversing");
        System.out.println(l.toString());

        Collections.reverse(l);
        System.out.println(x: "After Reversing");
        System.out.println(l);
    }
}

```

Output:



```
C:\Users\madha\Documents\java programs>javac ReverseArray.java

C:\Users\madha\Documents\java programs>java ReverseArray
Before Reversing
[Mango, Banana, Mango, Apple]
After Reversing
[Apple, Mango, Banana, Mango]

C:\Users\madha\Documents\java programs>
```

Experiment No 8

Aim:

Create a class Book and define a display method to display book information. Inherit Reference Book and Magazine classes from Book class and override display method of Book class in Reference_Book and Magazine classes. Make necessary assumptions required .

Description:

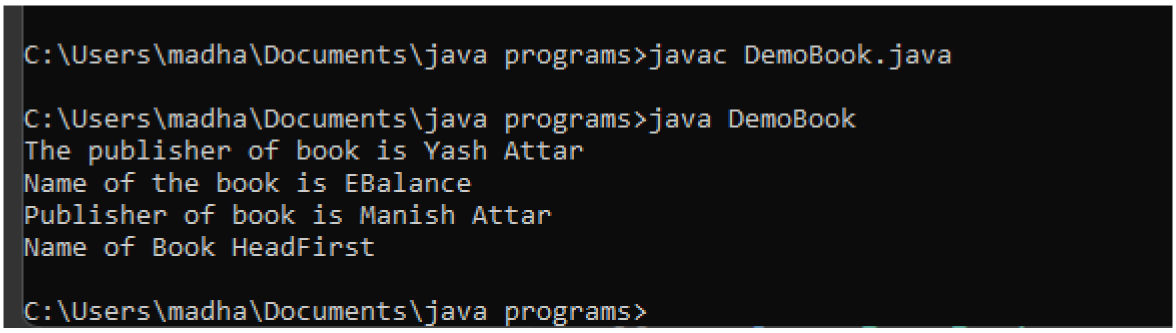
Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object

ALGORITHM

1. Start 2.
2. Create class Book and define method display.
3. 3. Create class Reference_Book extends Book and define method display.
4. 4. Create class Magzine extends Book and define method display.
5. 5. Create main class DemoIntrafaces and create object of each class from a Book class. Call display method of each class.
6. 6. Stop.

```
abstract class Book
{
 abstract void display();
}
class Magzine_book extends Book
{
   void display()
   {
      System.out.println("Publisher of book is Manish Attar");
      System.out.println("Name of Book HeadFirst");
   }
}

class reference_book extends Book
{
   void display()
   {
      System.out.println("The publisher of book is Yash Attar");
      System.out.println("Name of the book is EBalance");
   }
}
class DemoBook
{
   public static void main(String[] args)
   {
      reference_book r = new reference_book();
      Magzine_book b = new Magzine_book();
      r.display();
      b.display();
   }
}
```

Save as: DemoBook.java



```
C:\Users\madha\Documents\java programs>javac DemoBook.java

C:\Users\madha\Documents\java programs>java DemoBook
The publisher of book is Yash Attar
Name of the book is EBalance
Publisher of book is Manish Attar
Name of Book HeadFirst

C:\Users\madha\Documents\java programs>
```

Experiment no 9

Aim: Stack ADT implementation using inheritance (Interface).

Description:

To design a Java application to implement array implementation of stack using the concept of Interface and Exception handling.

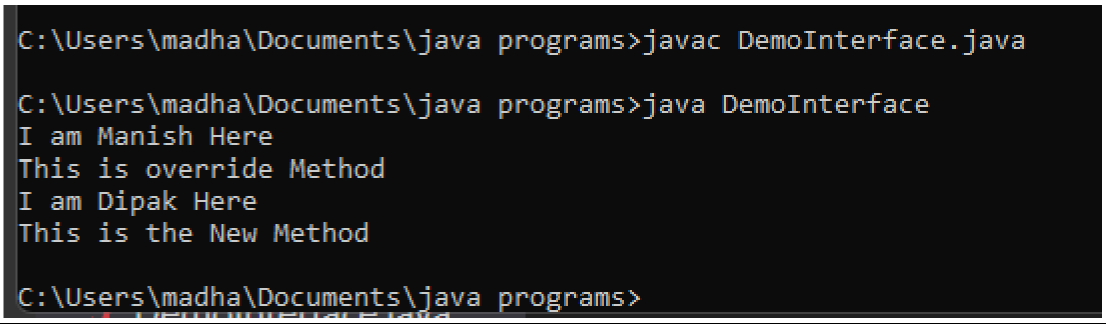
ALGORITHM

1. Start
2. create the interfaces Dipak
3. create display method
4. create man method
5. create method man2
6. create DemoInterfaces  extends man and implements Dipak
7. override the display method in this method
8. create main method
9. create object and call method .

Source Code:

```java
interface Dipak
{
    static final double pi=3.14;
    public void display();
}
   class Man
   {
        double z=200;

        void Man2()
        {
            System.out.println("I am Dipak Here");
            System.out.println("This is the New Method");
        }
   }
class DemoInterface extends Man implements Dipak
{
   public void display()
   {
    System.out.println("I am Manish Here");
    System.out.println("This is override Method");
   }
   public static void main(String[] args)
   {
    DemoInterface dr = new DemoInterface();
    dr.display();
    dr.Man2();
   }
}
```

Output:



```
C:\Users\madha\Documents\java programs>javac DemoInterface.java

C:\Users\madha\Documents\java programs>java DemoInterface
I am Manish Here
This is override Method
I am Dipak Here
This is the New Method

C:\Users\madha\Documents\java programs>
```

Experiment No 10

Write a program to display Calculator

Experiment No. 11

## Aim: User defined exception handling implementation.

## Description:

To write a Java program to implement user defined exception handling.

ALGORITHM:

1.Start

2. import java lang Exception

3. create class MyException and exrends it from Exception

4. print message

5. create class TestMyException and create main class

6. in main method perform divide operation and take if condition and print the message in try block

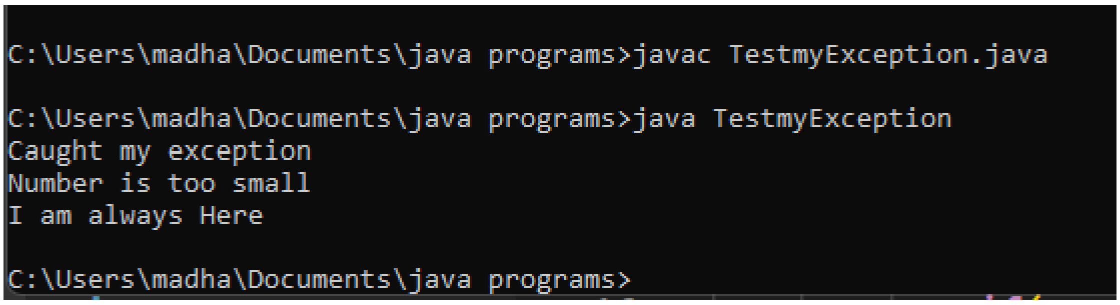
7. and catch exception and print it

8.and take finally block

Source Code:

```
import java.lang.Exception;
class MyException extends Exception
{
   MyException(String Message)
   {
      super(Message);
   }
}
class TestmyException
{
   public static void main(String[] args)
   {
      int x=5, y=1000;
      try{
         float z = (x )/(y);
         if(z < 0.01)
         {
            throw new MyException("Number is too small");
         }
      }
      catch(MyException e)
      {
          System.out.println("Caught my exception");
          System.out.println(e.getMessage());
      }
      finally
      {
          System.out.println("I am always Here");
      }
   }
}
```

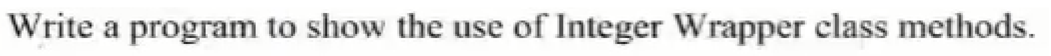
Output:



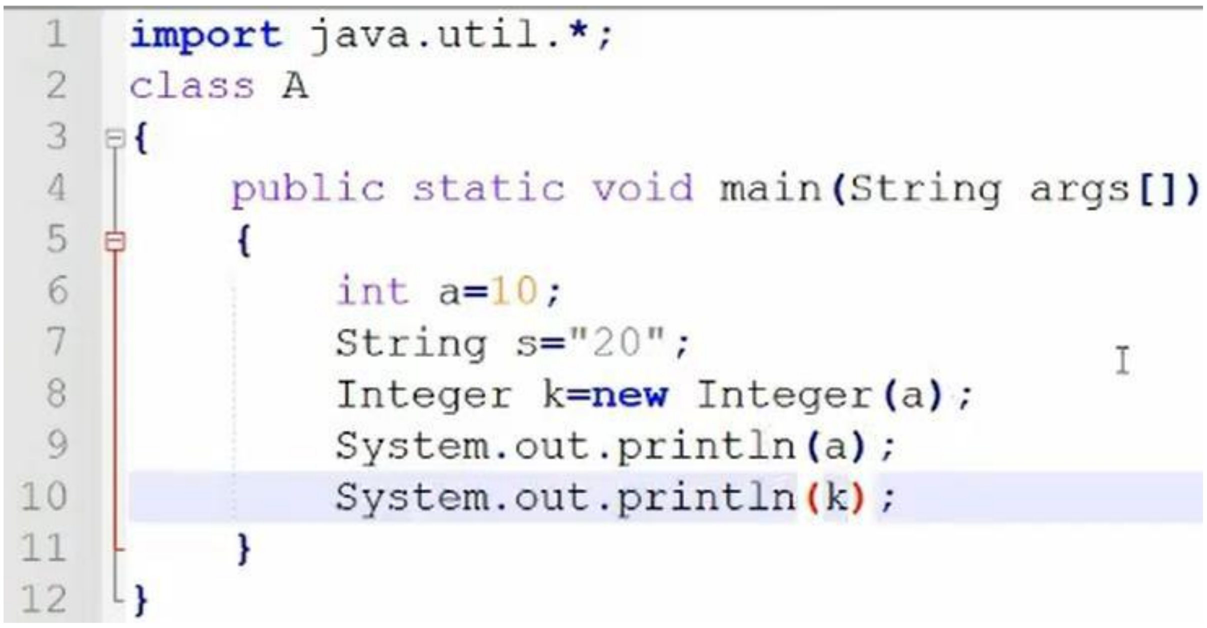
```
C:\Users\madha\Documents\java programs>javac TestmyException.java

C:\Users\madha\Documents\java programs>java TestmyException
Caught my exception
Number is too small
I am always Here

C:\Users\madha\Documents\java programs>
```

---

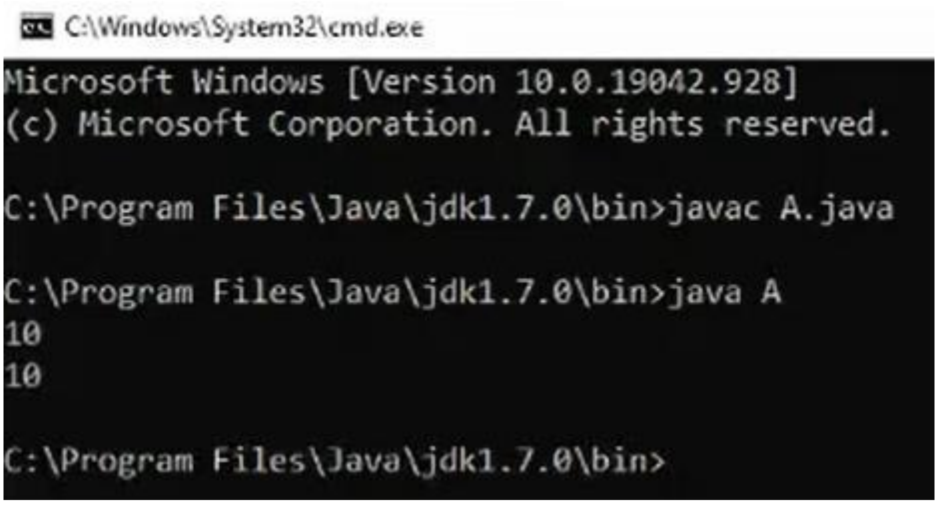
Experiment No 12



Write a program to show the use of Integer Wrapper class methods.

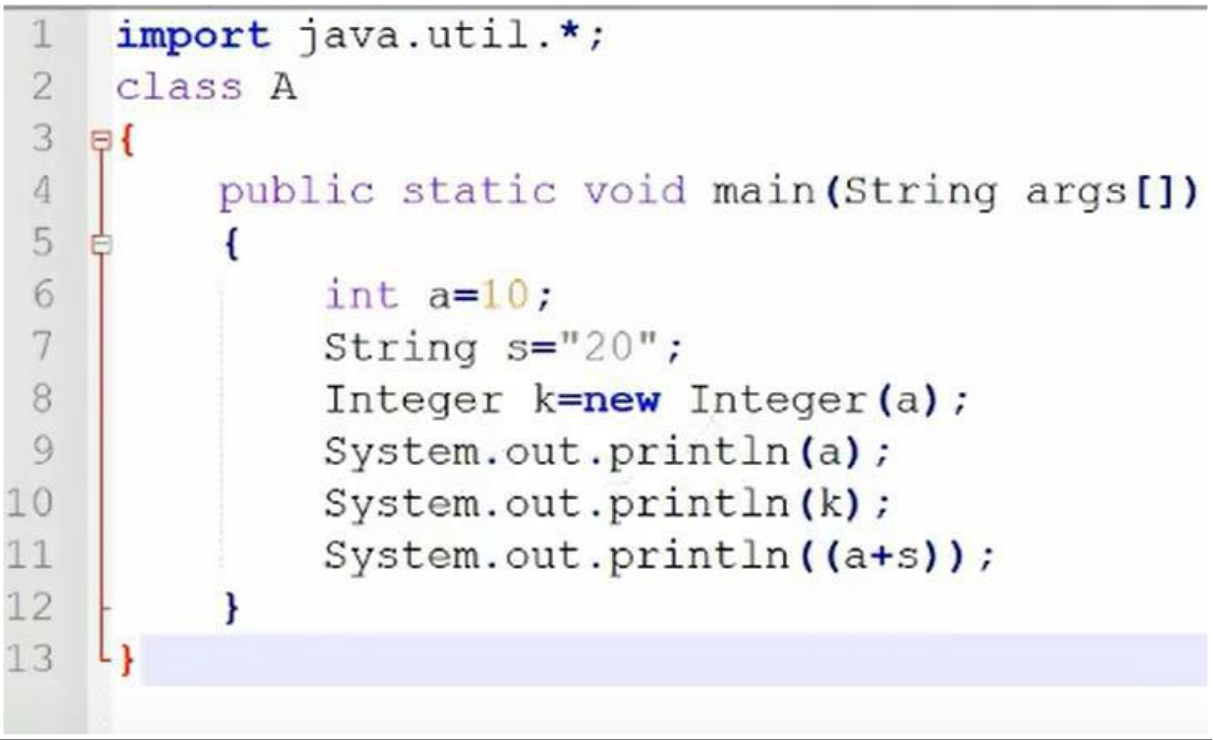


```
import java.util.*;
class A
{
        public static void main(String args[])
        {
                int a=10;
                String s="20";
                Integer k=new Integer(a);
                System.out.println(a);
                System.out.println(k);
        }
}
```



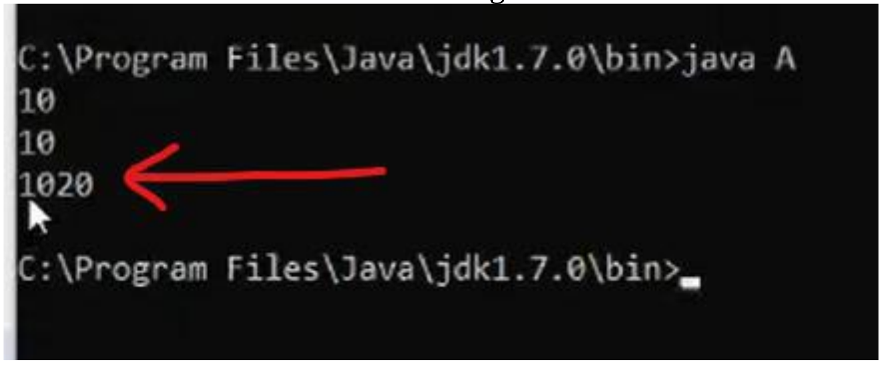
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Java\jdk1.7.0\bin>javac A.java

C:\Program Files\Java\jdk1.7.0\bin>java A
10
10

C:\Program Files\Java\jdk1.7.0\bin>
```

If we are doing addition of two numbers



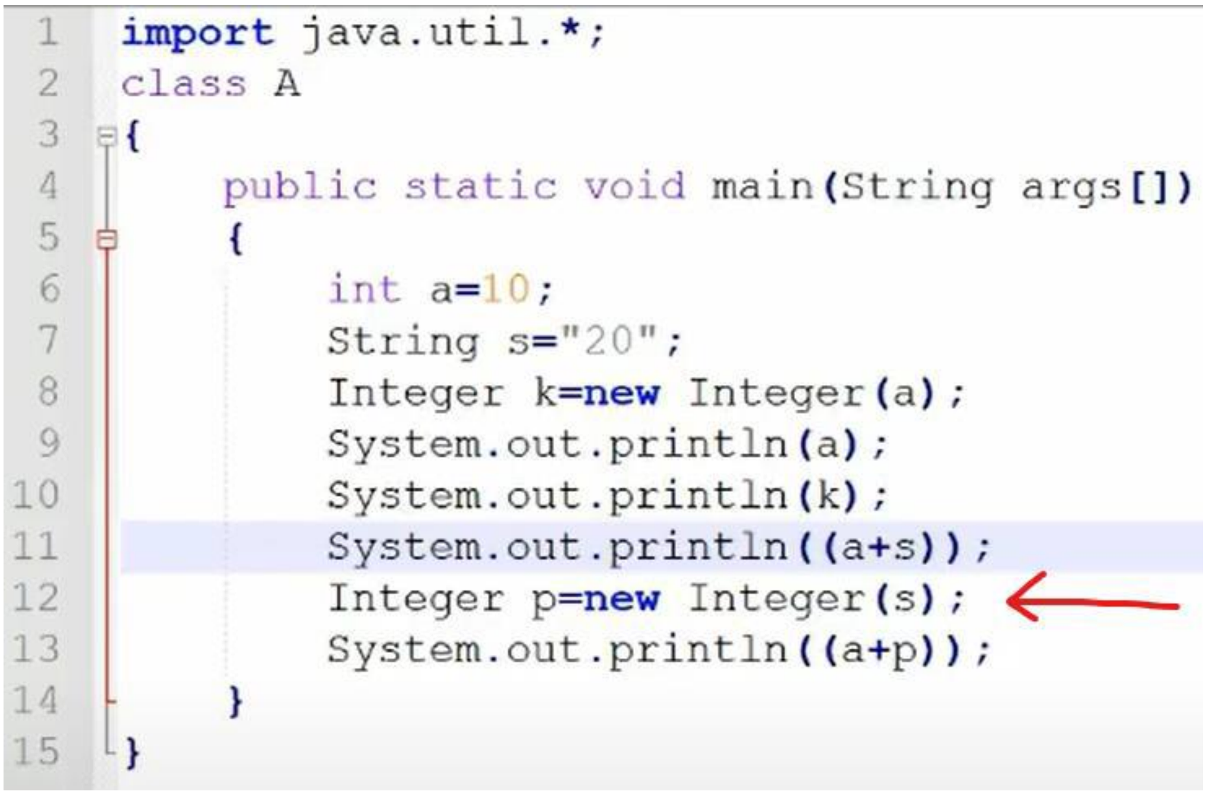
```java
import java.util.*;
class A
{
    public static void main(String args[])
    {
        int a=10;
        String s="20";
        Integer k=new Integer(a);
        System.out.println(a);
        System.out.println(k);
        System.out.println((a+s));
    }
}
```

In this addition not occur, only they join two numbers like string, it consider as string
For that we have to convert s into integer

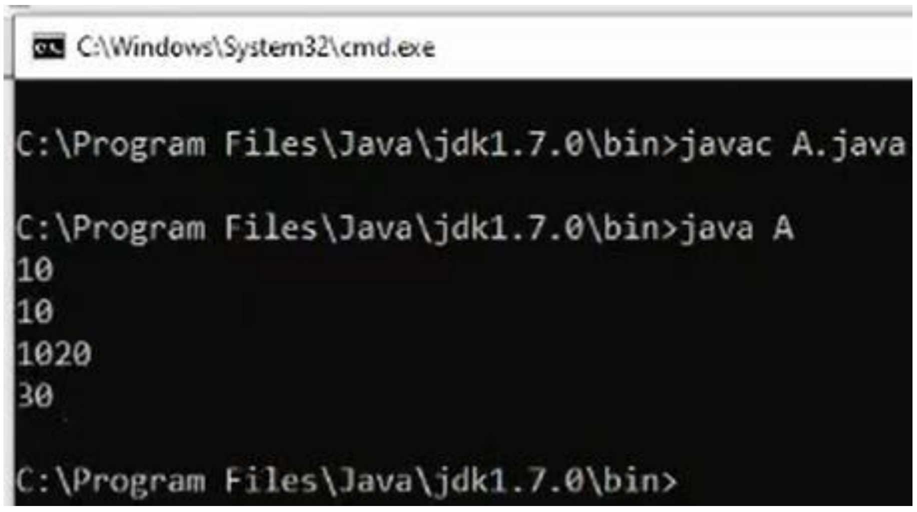


```
C:\Program Files\Java\jdk1.7.0\bin>java A
10
10
1020

C:\Program Files\Java\jdk1.7.0\bin>
```

Here we have convert into integer and store in p object

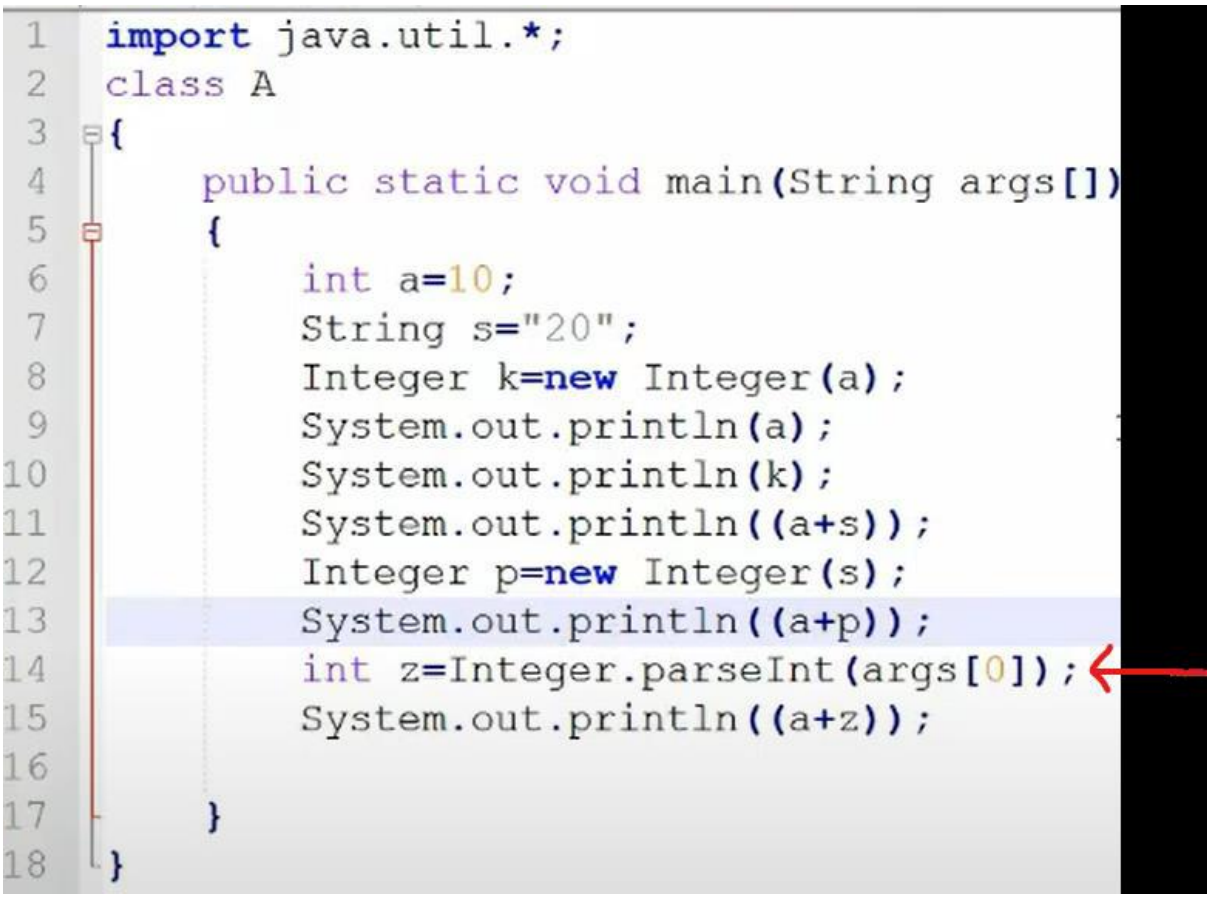


```java
import java.util.*;
class A
{
    public static void main(String args[])
    {
        int a=10;
        String s="20";
        Integer k=new Integer(a);
        System.out.println(a);
        System.out.println(k);
        System.out.println((a+s));
        Integer p=new Integer(s);
        System.out.println((a+p));
    }
}
```

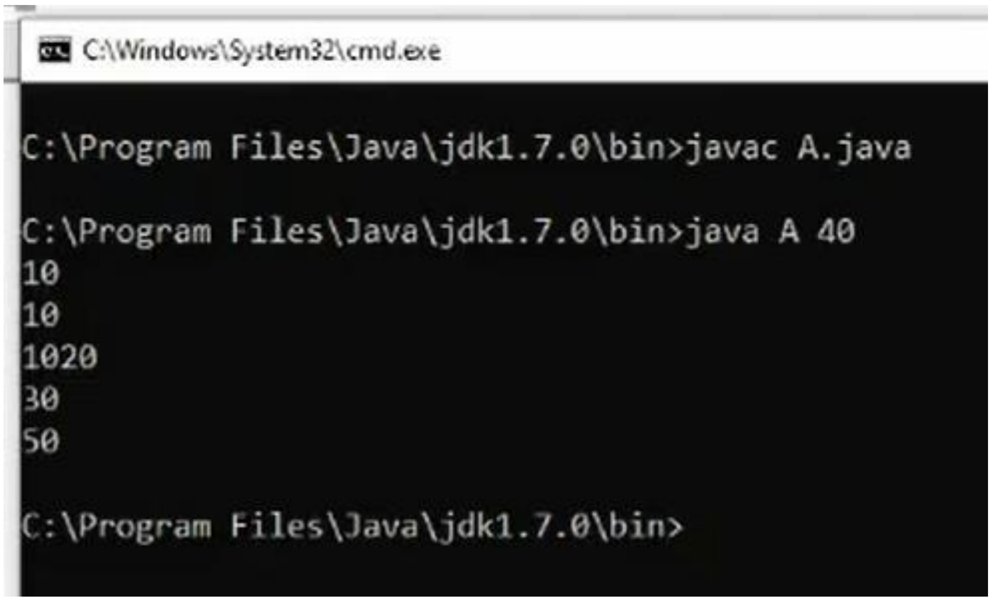


```
C:\Program Files\Java\jdk1.7.0\bin>javac A.java

C:\Program Files\Java\jdk1.7.0\bin>java A
10
10
1020
30

C:\Program Files\Java\jdk1.7.0\bin>
```

If we want to take input from keyboard, we need to use parseInt(args[0])



```java
import java.util.*;
class A
{
     public static void main(String args[])
     {
          int a=10;
          String s="20";
          Integer k=new Integer(a);
          System.out.println(a);
          System.out.println(k);
          System.out.println((a+s));
          Integer p=new Integer(s);
          System.out.println((a+p));
          int z=Integer.parseInt(args[0]);
          System.out.println((a+z));

     }
}
```



```
C:\Program Files\Java\jdk1.7.0\bin>javac A.java

C:\Program Files\Java\jdk1.7.0\bin>java A 40
10
10
1020
30
50

C:\Program Files\Java\jdk1.7.0\bin>
```

---

Experiment No. 13

Package in JAVA

Experiment No. 14

Aim: To write an Applet Program in Java.
Description: An applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.

Life Cycle of an Applet
Four methods in the Applet class gives you the framework on which you build any serious applet −
• **init** − This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.

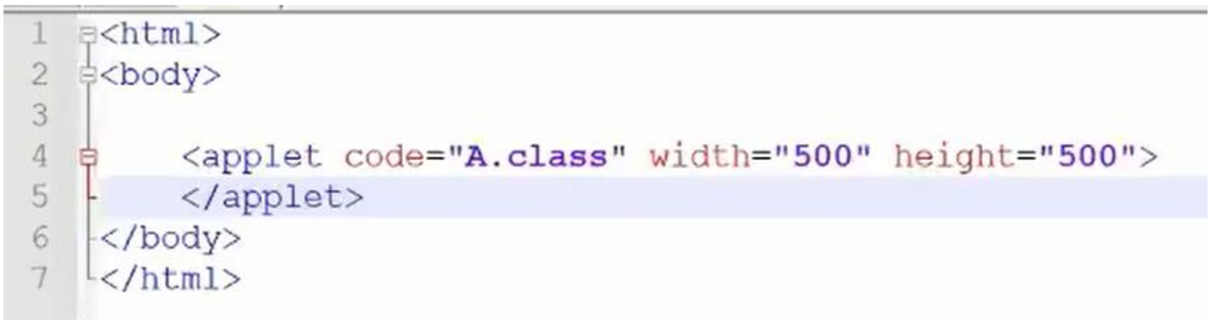
• **start** − This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.

• **stop** − This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.

• **destroy** − This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.

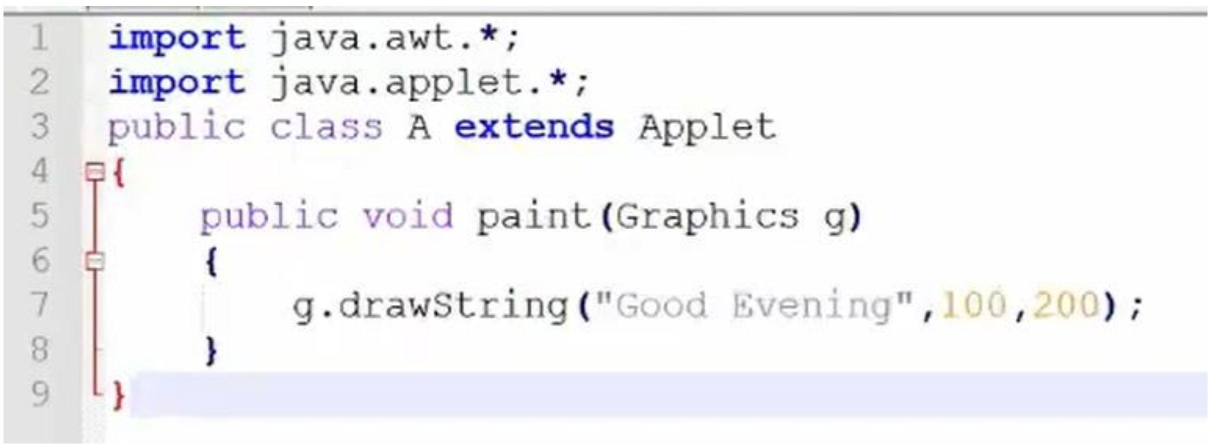
• **paint** − Invoked immediately after the start() method, and also any time the applet needs to repaint itself in browser. The paint() method is actually inherited from the java.awt.

**NOTE :** Applet is not supported for the JDK's with version greater than 9



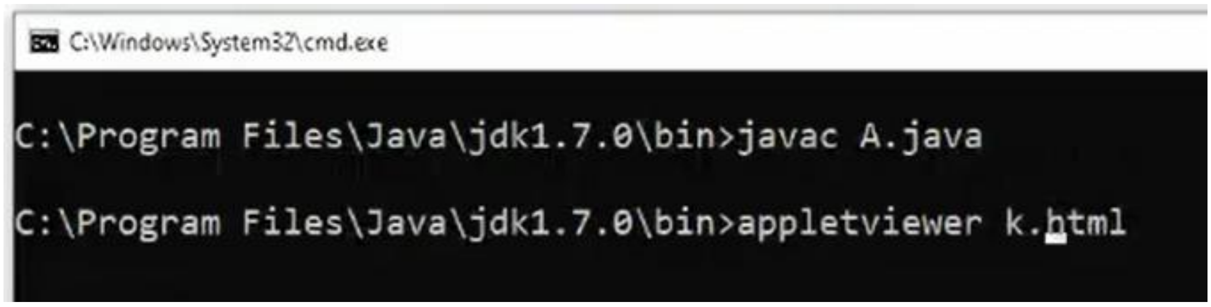
```
<html>
<body>

    <applet code="A.class" width="500" height="500">
    </applet>
</body>
</html>
```

Save as k.html  in same folder of java program

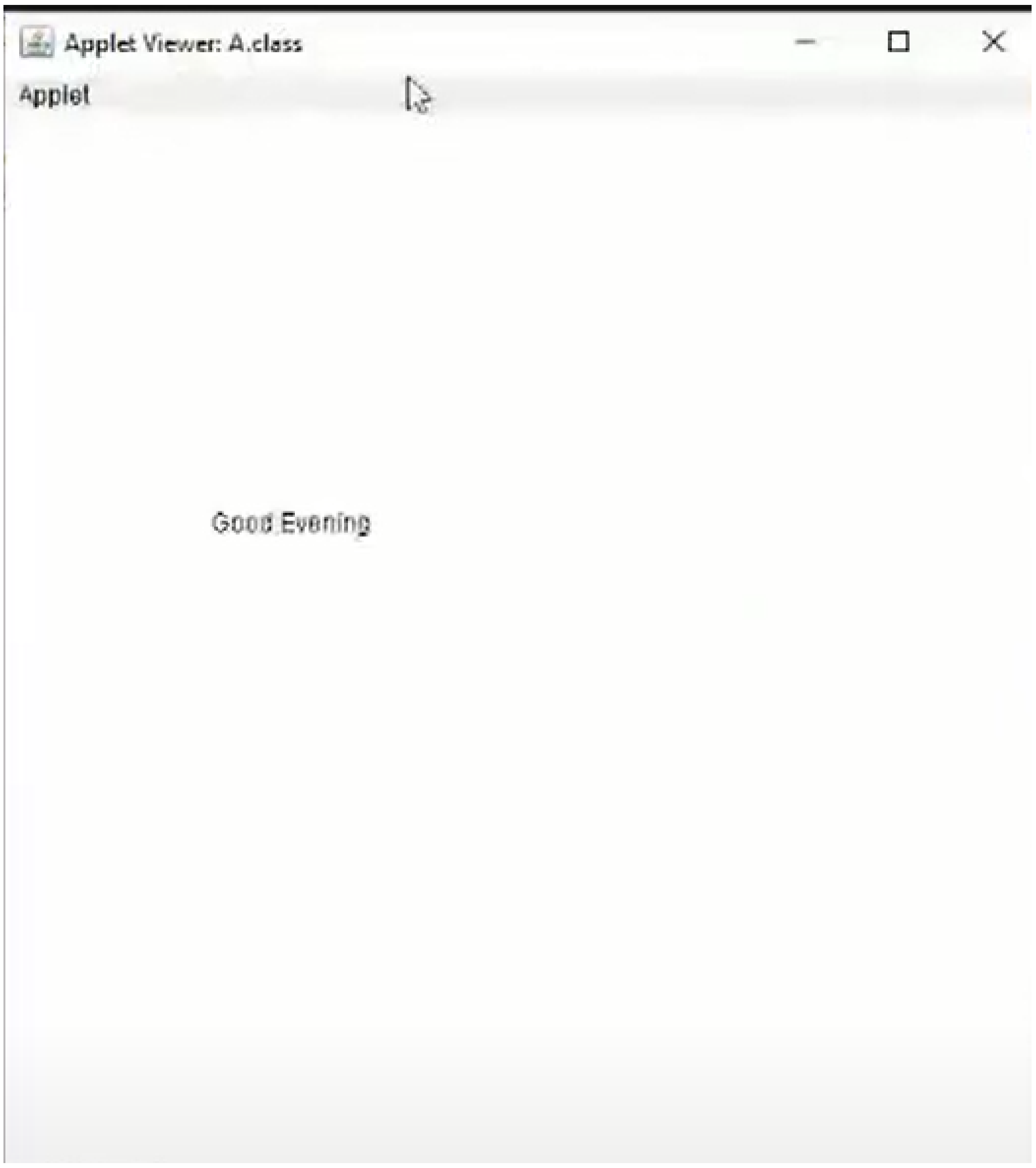


```java
import java.awt.*;
import java.applet.*;
public class A extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Good Evening",100,200);
    }
}
```

Save as A.java



```
C:\Program Files\Java\jdk1.7.0\bin>javac A.java

C:\Program Files\Java\jdk1.7.0\bin>appletviewer k.html
```



```
Good Evening
```

Experiment No. 15

1.File input stream and file output stream

2. Copy the contents of one file into another file