SONA COLLEGE OF ARTS AND SCIENCE
Affiliated to Periyar University, Salem

B.Sc Computer Science

PROGRAMMING IN JAVA
PRACTICAL RECORD

COMPUTER SCIENCE

2022 – 2023

SONA COLLEGE OF ARTS AND SCIENCE
Affiliated to Periyar University, Salem

B.Sc., Computer Science

III Year

PROGRAMMING IN JAVA
PRACTICAL RECORD

Name :

Register Number :

SONA COLLEGE OF ARTS AND SCIENCE

Affiliated to Periyar University, Salem

This is to Certify that this Practical Record " ***PROGRAMMING IN JAVA*** *Practical "is a bona-fide work done by* ____________________ *Reg.No:*__________________ *submitted to the Department of Computer science, during the academic year* **2022 - 2023**

SUBJECT IN-CHARGE **HEAD OF THE DEPARTMENT**

Submitted for University Practical Examination held on ________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Ex.no: 01	METHOD OVERLOADING
Date:	

AIM:

To write a java program to find the Area of Square, Rectangle and Circle using Method Overloading.

ALGORITHM:

Step1: Start the Program.

Step2: Create a class with main( ) method.

Step3: Overload the method to calculate the area of square, rectangle and circle.

Step4: Create an object for the class and call the methods.

Step5: Stop the program.

PROGRAM:

```
import java.io.*;
class JavaExample
{
   void calculatearea(float x)
   {
      System.out.println("Area of the square: "+x*x+" sq units");
   }
   void calculatearea(float x, float y)
   {
      System.out.println("Area of the rectangle: "+x*y+" sq units");
   }
   void calculatearea(double r)
   {
      double area = 3.14*r*r;
      System.out.println("Area of the circle: "+area+" sq units");
   }
   public static void main(String args[ ])
   {
          JavaExample obj = new JavaExample( );
         obj.calculatearea(6.1f);
         obj.calculatearea(10,22);
         obj.calculatearea(6.1);
    }  }
```

OUTPUT:

```
>javac JavaExample.java
>java JavaExample
Area of the square: 37.21 sq units

Area of the rectangle: 220.0 sq units

Area of the circle: 116.8394 sq units
```

RESULT:

Thus the method overloading was successfully performed using three different methods.

Ex.no: 02	COMMAND LINE ARGUMENTS
Date:	

AIM:

To write a java program to sort the list of numbers using Command Line Arguments

ALGORITHMS:

Step1: Start the program.

Step2: Create a class with main( ) method.

Step3: Create an array and store the value passed from the command line arguments.

Step4: Sort the elements of the array by comparing all the values with each other.

Step5: Display the sorted array elements.

Step6: Stop the program.

PROGRAM:

```
public class cmd
  {
        public static void main(String[ ] args)
        {
        int a[ ]=new int[20]; int j, temp, i;
        for(i=0; i<args.length; i++)
        {
                a[i]=Integer.valueOf(args[i]);
        }
        System.out.println("Elements in the unsorted array are: ");
        for(i=0; i<args.length; i++)
        {
                System.out.println(a[i]+"\t"); }
                for(i=0; i<args.length; i++)
                {
                for(j=0; j<args.length-i-1; j++)
                {
                                if(a[j] > a[j+1])
                {
                        temp=a[j]; a[j]=a[j+1];
                        a[j+1]=temp;
                }
                }
                }
                        System.out.println("Sorted array elements are: ");
                for(i=0; i<args.length; i++)
                {
                        System.out.println(a[i] + "\t")
                }
              }
        }
```

OUTPUT:

```
>javac cmd.java
>java cmd 10 50 15 12
Elements in the unsorted array are: 10
50
15
12
Sorted array elements are: 10
12
15
50
```

RESULT:

Thus the elements was sorted successfully using Command Line Arguments.

Ex.no: 03	**MATRIX MULTIPLICATION**
Date:	

AIM:

To write a java program to sort the list of numbers using Command Line Arguments

ALGORITHMS:

Step1:	Start the program.
Step2:	Create a class with main( ) method.
Step3:	Read the input for order of two matrices.
Step4:	Read the input of two matrices.
Step5:	Multiply the two matrices and store the result in another matrix.
Step6:	Print the elements in the resultant matrix.
Step7:	Stop the program.

PROGRAM:

```
import java.util.Scanner;
class matrix
{
    public static void main(String args[ ])
     {
        int m, n, p, q, sum = 0, c, d, k;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of rows and columns of first matrix");
        m = in.nextInt( );
        n = in.nextInt( );
        int first[ ][ ] = new int[m][n];
        System.out.println("Enter elements of first matrix");
        for (c = 0; c < m; c++)
              for (d = 0; d < n; d++)
                    first[c][d] = in.nextInt( );
        System.out.println("Enter the number of rows and columns of second matrix");
        p = in.nextInt( );
        q = in.nextInt( );
        if (n != p)
            System.out.println("The matrices can't be multiplied with each other.");
        else
          {
               int second[ ][ ] = new int[p][q];
                int multiply[ ][ ] = new int[m][q];
               System.out.println("Enter elements of second matrix");
               for (c = 0; c < p; c++)
                     for (d = 0; d < q; d++)
                              second[c][d] = in.nextInt( );
               for (c = 0; c < m; c++)
                 {
                         for (d = 0; d < q; d++)
                           {
                               for (k = 0; k < p; k++)
                                 {
                                     sum = sum + first[c][k]*second[k][d];
                                 }
                          multiply[c][d] = sum;
                          sum=0;
                  }
             }
            System.out.println("Product of the matrices:");
            for (c = 0; c < m; c++)
              {
                  for (d = 0; d < q; d++)
                      System.out.print(multiply[c][d]+"\t");
                      System.out.print("\n");
              }
          }
     }
}
```

OUTPUT:

```
      >javac matrix.java
      >java matrix
Enter the number of rows and columns of first matrix : 2

Enter the elements of first matrix : 3
5
7
5

Enter the number of rows and columns of second matrix : 2

Enter the elements of second matrix : 3
4
9
8

Product of the matrices :
54      52
66      68
```

RESULT:

Thus the matrix multiplication was successfully executed and produced the resultant matrix.

Ex.no: 04	**BANK ACCOUNT MANAGEMENT**
Date:	

AIM:

To write a java program for performing bank operations.

ALGORITHM:

Step1: Start the program.

Step2: Create a class with data members such as depositor name, account number, type of account and balance amount.

Step3: Create methods such as assign initial values, deposit, withdraw, balance checking and display the details.

Step4: Create an object for the class in the main( ) method.

Step5: Call the methods to do a particular operation.

Step6: Stop the program.

PROGRAM:

```
import java.util.*;
class bk
{
    private String name, acttype;
    private int acno, balance;
    public Scanner in;
    public bk( )
    {
          acno=0; balance=0;
          in=new Scanner(System.in);
    }
   public void get( )
    {
      System.out.println("Enter acno, name, type of account, balance");
      acno=in.nextInt( );
      name=in.next( );
      acttype=in.next( );
      balance=in.nextInt( );
    }
  public void deposit( )
    {
       int dep;
       System.out.println("Enter the amount to be deposited");
      dep=in.nextInt( );
      balance=balance+dep;
     }
   public void withdraw( )
     {
        int withdraw;
        System.out.println("Enter the amount to be drawn");
         withdraw=in.nextInt( );
        if(balance<withdraw)
         {
                System.out.println("Required balance not available");
          }
        else
            balance=balance-withdraw;
       }
     public void display( )
      {
         System.out.println("Account holder Name="+ name +" Balance=" + balance);
      }
  }
class bank
 {
      public static void main(String args[ ])
         {
            bk ob=new bk( );
            ob.get( );
            ob.deposit( );
            ob.withdraw( );
            ob.display( );
          }
   }
```

OUTPUT:

```
>javac bank.java
>java bank
Enter acno, name, type of account, balance
101
siva
savings
1500
Enter the amount to be deposited 500
Enter the amount to be drawn 200
Account holder Name=siva Balance=1800
```

RESULT:

Thus the program was successfully executed and the details about the particular account has been displayed.

Ex.no: 05	**USER – DEFINED PACKAGE**
Date:	

AIM:

To write a java program that import the user-defined package and access the member variable of classes that contained by the package.

ALGORITHM:

Step1: Start the process.

Step2: Import the necessary packages.

Step3: Create a user defined package call mypack with data member and method.

Step4: Import the user defined package call mypack in another java file.

Step5: Create a class called xxx to display the string and variable value.

Step6: Save file and run the program and display the output.

Step7: Terminate the program.

PROGRAM:

```
package mypack;
public class abc
 {
       public int a=10;
       public void getdata(String s)
        {
            System.out.println("the string is" +s);
        }
 }

import mypack.abc;
class xxx
 {
     public static void main(String args[])
        {
            abc t=new abc();
            t.getdata("shan");
```

```
                System.out.println("the variable is" + t.a);
            }
        }
```

OUTPUT:

```
>javac abc.java
>javac xxx.java
>java xxx
the string is shan
the variable is 10
```

RESULT:

Thus the package was imported and accessed successfully.

Ex.no: 06	**EXCEPTION HANDLING USING TRY AND MULTIPLE CATCH BLOCKS**
Date:	

AIM:

To perform a program to handle the Exception using try and multiple catch blocks.

ALGORITHM:

Step1: Start the program.

Step2: Inside the main function create a perform try operation.

```
{
  int a[]=new int[5];
  a[5]=30/0;
}
```

Step3: Perform the Exception using try and multiple catch blocks operations in the program. catch(Arithmetic Exception e),
catch(ArrayIndexOutOfBoundsException e) &
catch(Exception e).

Step4: If the exception is occur it will print the appropriate statements in the above methods.

Step5: Display the output in the command prompt.

Step6: Stop the program

PROGRAM:

```
public class multiplecatchblock
 {
        public static void main(String[] args)
        {
            try
            {
                 int a[]=new int[5];
                 a[5]=30/0;
            }
```

```
                catch(Arithmetic Exception e)
                 {
                    System.out.println("ARITHMETIC EXCEPTION OCCURS");
                 }
                catch(ArrayIndexOutOfBoundsException  e)
                 {
                    System.out.println("Array Index of Bounds Exception Occurs");
                 }
                catch(Exception e)
                {
                    System.out.println("Parent Exception Occurs");
                 }
                    System.out.println("rest of the code");
          }
     }
```

OUTPUT:

```
>javac multiplecatchblock.java
>java multiplecatchblock
ARITHMETIC EXCEPTION OCCURS
Rest of the code
```

RESULT:

Thus the exception was handled and displayed successfully.

Ex no: 07	MULTITHREADS
Date:	

AIM:

To perform a program to illustrate the use of multithreads .

ALGORITHM:

Step1:	Start the program.
Step2:	Create a new thread inside main( ) function. Thread thread1 = new Thread("yyy");
Step3:	Perform multithread operations in the program.
Step4:	Create objects for the threads.
Step5:	Stop the program.

PROGRAM:

```
public class GuruThread1 implements Runnable
{
      public static void main(String[] args)
      {
           Thread thread1 = new Thread("yyy");
           Thread thread2 = new Thread("xxx");
           thread1.start();
           thread2.start();
           System.out.println("Thread names are following:");
           System.out.println(thread1.getName());
           System.out.println(thread2.getName());
      }

     @Override
     public void run( )
      {
      }
  }
```

OUTPUT:

```
>javac GuruThread1.java
>java GuruThread1
Thread names are following:
yyy
xxx
```

RESULT:

Thus the program has been successfully executed and output has been displayed.

Ex.no: 08	**STUDENT REGISTRATION FORM USING APPLET**
Date:	

AIM:

To write an applet program for student registration form.

ALGORITHM:

Step1: Start the program.

Step2: Create a class that extends the Applet class.

Step3: Create an object for TextField, Button, Checkbox and Label classes in the init( ).

Step4: Override the actionPerformed( ) method to check and display the events.

Step5: Stop the program.

PROGRAM:

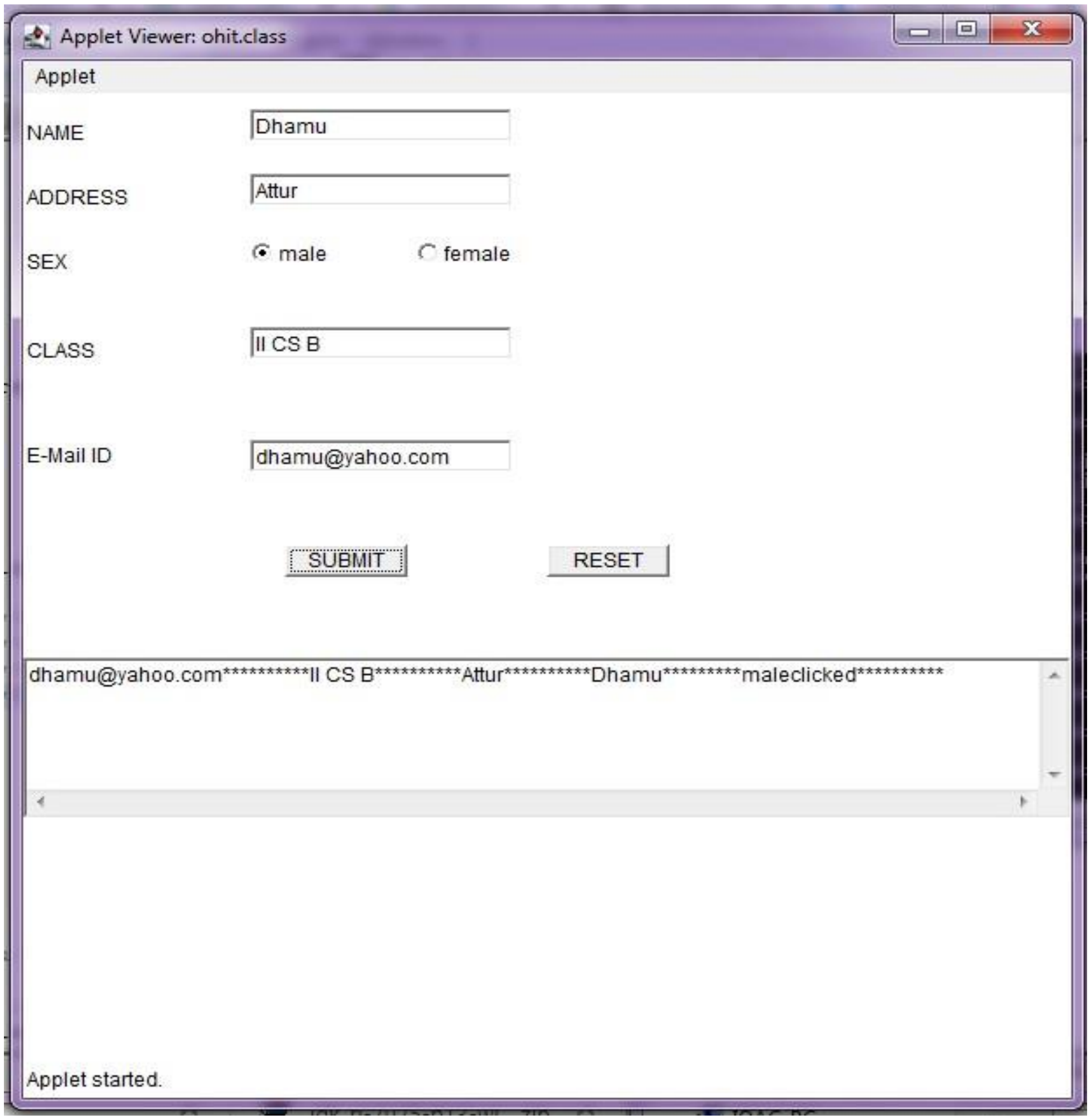
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/* <applet code="regis.class" width=600 height=600>
</applet> */

public class regis extends Applet implements ActionListener,ItemListener
{
        TextField t3,t4,t5,t6,t7;
        Button b1,b2;
        Checkbox c1,c2,c3,c4,m,f;
        CheckboxGroup cbg;
        List l1;
        Label l2,l3,l4,l5,l6;
        TextArea tx1;

        public void init( )
        {
          setLayout(null);
          l2=new Label("NAME");
```

```
l2.setBounds(0,0,50,50);
add(l2);
t3=new TextField(20);
t3.setBounds(130,10,150,20);
add(t3);
 l3=new Label("ADDRESS");
l3.setBounds(0,40,70,50);
add(l3);
t4=new TextField(20);
t4.setBounds(130,50,150,20);
add(t4);

l4=new Label("SEX");
l4.setBounds(0,80,70,50);
add(l4);
cbg=new CheckboxGroup( );
m=new Checkbox("male",false,cbg);
m.setBounds(130,90,75,20);
add(m);
m.addItemListener(this);
f=new Checkbox("female",false,cbg);
f.setBounds(225,90,75,20);
add(f);
f.addItemListener(this);
l5=new Label("CLASS");
l5.setBounds(0,135,50,50);
add(l5);
t5=new TextField(20);
t5.setBounds(130,145,150,20);
add(t5);
l6=new Label("E-Mail ID");
l6.setBounds(0,195,60,60);
add(l6);
t6=new TextField(20);
t6.setBounds(130,215,150,20);
add(t6);

b1= new Button("SUBMIT");
b1.setBounds(150,280,70,20);
add(b1);
b1.addActionListener(this);
b2= new Button("RESET");
b2.setBounds(300,280,70,20);
add(b2);
```

```
            b2.addActionListener(this);
            tx1=new TextArea("",10,20,TextArea.SCROLLBARS_BOTH);
            tx1.setBounds(0,350,600,100);
            add(tx1);

      }

      String selections[ ];
      public void actionPerformed(ActionEvent e)
      {
             if(e.getSource( )==b1)
            {
                  tx1.insert(t3.getText( )+"*********",0);
                  tx1.insert(t4.getText( )+"**********",0);
                  tx1.insert(t5.getText( )+"**********",0);
                  tx1.insert(t6.getText( )+"**********",0);
            }

           String msg= new String("");
           if(e.getSource( )==b2)
            {
                   tx1.setText(msg);
                   t3.setText(msg);
                   t4.setText(msg);
            }
            String outString=new String("you selected");
            if(e.getSource( )==b1)
            {
              selections=l1.getSelectedItems( );
              for(int loop=0; loop<selections.length; loop++)
               {
                   outString +=" " + selections[loop];
                }
               tx1.insert(outString,0);
              }
       }
       public void itemStateChanged(ItemEvent e)
       {
            tx1.insert(((Checkbox)e. getItemSelectable( )).getLabel( ) + "clicked**********",0);
        }
    }
```

OUTPUT:

```
 >javac regis.java
> start appletviewer
> appletviewer regis.java
```



Applet Viewer: ohit.class

Applet

NAME: Dhamu

ADDRESS: Attur

SEX: (● male) ○ female

CLASS: II CS B

E-Mail ID: dhamu@yahoo.com

SUBMIT RESET

```
dhamu@yahoo.com**********II CS B**********Attur**********Dhamu*********maleclicked**********
```

Applet started.

RESULT:

Thus the student registration form was successfully created using Applet.

Ex.no: 09	**GRAPHICS METHOD USING APPLET**
Date:	

AIM:

To write an applet program to display line, rectangle, oval and text using graphics method.

ALGORITHM:

Step1: Start the program.

Step2: Create a class that extends Applet class.

Step3: Override the paint( ) method.

Step4: Call the method to display the line, rectangle, oval and text.

Step5: Stop the program.

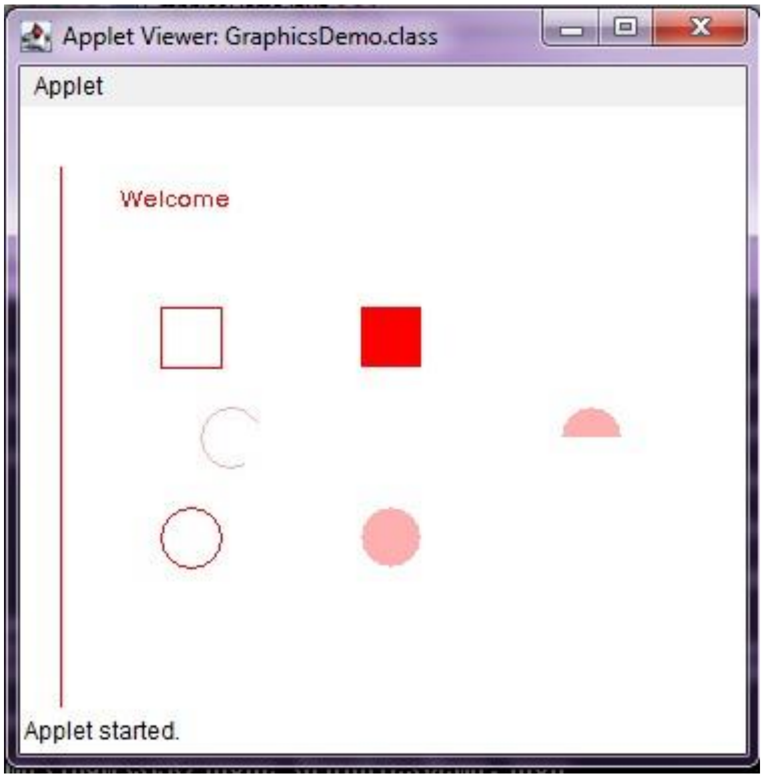
PROGRAM:

```
import java.applet.Applet;
import java.awt.*;
/*
<applet code="GraphicsDemo.class" width="300" height="300">
</applet>
*/
public class GraphicsDemo extends Applet
{
        public void paint(Graphics g)
        {
                g.setColor(Color.red);
                g.drawString("Welcome",50, 50);
                g.drawLine(20,30,20,300);
                g.drawRect(70,100,30,30);
                g.fillRect(170,100,30,30);
                g.drawOval(70,200,30,30);
                g.setColor(Color.pink);
                g.fillOval(170,200,30,30);
                g.drawArc(90,150,30,30,30,270);
                g.fillArc(270,150,30,30,0,180);
```

```
            }
    }
```

OUTPUT:

```
>javac GraphicsDemo.java
> start appletviewer
> appletviewer GraphicsDemo.java
```



RESULT:

Thus the line, rectangle, oval, text has been drawn using the graphics method successfully and output has been displayed.

Ex.no: 10	**SEQUENTIAL FILE OPERATIONS**
Date:	

AIM:

To write a java program for accessing a file sequentially.

ALGORITHM:

Step1:	Start the program.
Step2:	Create a class for getting product information.
Step3:	Declare data members such as product code, cost and count.
Step4:	Write methods to get input for the product information.
Step5:	Write methods to return the product information.
Step6:	Create a class with main( ) function.
Step7 :	Create a file to store product information.
Step 8:	Create an object for product information.
Step 9:	Call methods of an object to get product information and store it in the file.
Step 10:	Calculate the product value and display.
Step 11:	Stop the program.

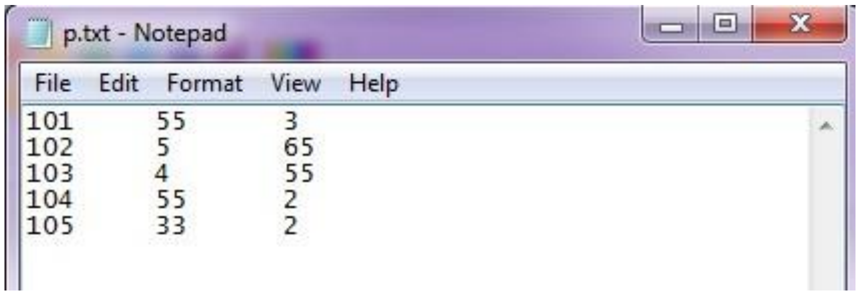
PROGRAM:

```
import java.io.*;
import java.util.*;
class pdt
{
        public int pcode, pcost, pcount;
        Scanner in;
        public pdt( )
        {
```

```
                    in=new Scanner(System.in);
            }
            public void get( )
            {
                    System.out.println(“Enter the product code :”);
                    pcode=in.nextInt( );
                    System.out.println(“Enter the product cost :”);
                    pcost =in.nextInt( );
                    System.out.println(“Enter the number of items :”);
                    pcount=in.nextInt( );
             }
             public int gpcode( )
            {
                    return pcode;
            }
            public int gpcount( )
            {
                    return pcount;
            }
            public int gcost( )
            {
                    return pcost;
            }
     }
    class test
    {
            public static void main(String args[ ] ) throws IOException
            {
                FileWriter fw=new FileWriter("p.txt");
                int i, tot=0;
                pdt pd[ ]=new pdt[5];
                for(i=0; i<5; i++)
                {
                    pd[i]=new pdt( );
                    pd[i].get( );
                    tot=tot+(pd[i].gcost( )*pd[i].gpcount( ));
                    fw.write("\n"+pd[i].gpcode( )+"\t"+pd[i].gcost( )+"\t"+pd[i].gpcount( )+"\n");
                 }
               System.out.println("Total value="+tot);
               fw.close( );
            }
     }
```

OUTPUT:

```
>javac test.java

> java test

Enter the product code: 101
Enter the product cost:  55
Enter the number of items: 3
Enter the product code: 102
Enter the product cost 5
Enter the number of items: 65
Enter the product code: 103
Enter the product cost: 4
Enter the number of items: 55
Enter the product code: 104
Enter the product cost: 55
Enter the number of items: 2
Enter the product code: 105
Enter the product cost: 33
Enter the number of items: 2
Total value=886
```



```
101     55     3
102     5      65
103     4      55
104     55     2
105     33     2
```

RESULT:

Thus the product value has been calculated and stored in a file successfully.