Book Database

```
import java.util.Scanner;
class Books
{
    String name;
    String author;
    int price;
    int numpages;
    Books() { }
    Books (String name, String author, int price,
           int numPages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numpages = numPages;
    }
    public String toString()
    {
        String name, author, price, numpages;
        name = "Book name:" + this.name + "\n";
        author = "Auth name :" + this.author + "\n";
        price = "price:" + this.price + "\n";
        numpages = "number of pages:" + this.numpages + "\n";
        return name + author + price + numpages;
    }
}
```

```
{
    scanner s = new scanner (system.in);
    int n;
    string name;
    string author;
    int price;
    int numpages;
    System.out.print("Enter the number of books:");
    n = s.nextInt();

    Books b[];
    b = new Books[n];
for(int i = 0; i<n; i++)
{
    System.out.println("Book "+(i+1)+":");
    System.out.print("Enter name of book:");
    name = s.next();
    System.out.print("Enter author:");
    author = s.next();
    System.out.Print("Enter price:");
    price = s.nextInt();
    System.out.print("Enter no. of pages:");
    numPages = s.nextInt();
b[i] = new Books(name, author, price,
```

```
for (int i=0; i<n; i++)
    System.out.print("Books:"+(i+1)+":"+b[i]);
}
}
```

OUTPUT

Enter the number of book : 2

Book 1:

Enter the name of the book : Jungle book

Enter the author of the book : Rudyard - kipling.

Price of the book : 1000

number of pages of the book : 500

Book 2:

Enter the name of the book : Tales of Akbar - Ambi -

Enter the author of the book : Ambi

Enter the price of the book : 900

Enter the number of pages of the book : 400

Book 1:

Book name : Jungle_book

Author : Rudyard - kipling

Price : 1000

number of pages : 500

Book 1:

Book name : Tales_of_Akbar_Ambi_Bubal

Author : Bubal

Price = 900

STUDENT DATABASE

```
import java.util.scanner
class student {
String USN ;
String name ;
double [] marks = new double [6];

void inputDetails ( ) {
Scanner scanner = new Scanner (System.in);

System.out.print ("Enter USN:");
USN = scanner.nextLine();

System.out.print ("Enter Name: ");
name = scanner.nextLine ( );

System.out.println ("Enter marks for 6 subjects:");
for (int i=0 ; i<6 ; i++) {
System.out.print ("subject "+ (i+1)+":");
marks [i] = scanner.nextDouble ( );
}

}
```

Lab - 4

1. Develop a Java program to create an abstract class named shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain the method printArea() that prints the area of the given shape.

```
abstract class Shape {
    int length;
    int width;

    public Shape(int length, int width) {
        this.length = length;
        this.width = width;
    }

    public abstract void printArea();
}

class Rectangle extends Shape {
    public Rectangle(int length, int width) {
        super(length, width);
    }

    public void printArea() {
        int area = length * width;
        System.out.println("Rectangle Area: "
        + area);
    }
}
```

```
    public Triangle (int length, int width) {
        super (length, width);
    }

    public void printArea() {
        double area = 0.5 * length * width;
        System.out.println("Triangle area:" + area);
    }
}

class circle extends shape {
    public circle (int radius) {
        super (radius, 0);
    }

    public void print area() {
        double area = Math.PI * length * length;
        System.out.println("Circle Area: " + area);
    }
}

public class main {
    public static void main (String[] args) {
        Rectangle r = new Rectangle (5, 10);
        r.print area ();

        Triangle t = new Triangle (4, 6);
        t.print area ();
```

c. print area();

}

}

OUTPUT

Rectangle Area : 50.00

Triangle Area : 12.00

Circle Area : 28.26

18/1/24

2. Develop a java program to create a class Bank that maintains two kinds of account for its customers, one called savings and current acc. The saving acc provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book ~~fig~~ facility but no interest current holder should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class account that stores customer name, acc no. and type of acc. From this device, the classes current and sav. acc to make them more specific to their requirements, include the necessary methods in order to achieve the following tasks:

a). Accept deposit from customer and update the balance.

b). Display the balance.

c). Compute and deposit interest.

d) ... withdrawal and update the balance

```
import java.util.Scanner;
class Account {
    String customerName;
    int accountNumber;
    String accountType;
    double balance;

    public Account(String customerName, int accountNumber, String accountType, double balance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = balance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit successful. Updated Balance: " + balance);
    }

    public void displayBalance() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }
}

class SavAcct extends Account {
    double interestRate;

    public SavAcct(String customerName, int accountNumber, double balance) {
```

```
super (customerName, accountNumber, "saving", balance);
this.interestRate = 0.05;
}

public void depositInterest () {
    double interest = balance * interestRate;
    balance += interest;
    System.out.println ("Interest deposited.
    updated balance :" + balance);
}

public void withdraw (double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println ("Withdrawal successful.
        updated balance :" + balance);
    } else {
        System.out.println ("Insufficient funds for
                withdrawal.");
    }
}
}

class CurAcct extends Account {
    double minBalance;
    double serviceCharge;

public CurAcct (String customerName, int AccountNumber, double Balance, double minBalance,
    double serviceCharge) {
super (customerName, accountNumber, "current", balance);
this.minBalance = minBalance;
this.serviceCharge = serviceCharge;
}
```

```
public void withdraw (double amount) {
    if (balance - amount >= minBalance) {
        balance -= amount;
        System.out.println("Withdrawal successful.
    Updated balance: " + balance);
    } else {
        System.out.println("Insufficient funds for
    withdrawal. Service charge of " + service
    charge + " imposed.");
    balance -= serviceCharge;
    System.out.println("Updated balance after
    service charge: " + balance);
    }
  }
}

public class Bank {
    psvm {
    Scanner sc = new Scanner(System.in);
    int choice, name, acc, bal
    while (1) {
System.out.println(menu);
System.out.println("press 1 for savings acc");
System.out.println("press 2 for cur acc");
System.out.println("choice");
case 1;
CurAcct currentAccount = new CurAcct
("Isha Doe", 123456789, 1000, 500, 20);
```

```
        currentAccount.deposit(500);
        currentAccount.displayBalance();
        currentAccount.withdraw(700);
currentAccount.checkMinimumBalance();
        System.out.println();
        SavAcct savingsAccount=new SavAcct("JaneSmith",
        987654321, 2000, 5);

            savingsAccount.deposit(1000);
            savingsAccount.displayBalance();
            savingsAccount.computeInterest();
            savingsAccount.withdraw(300);

            scanner.close();
        }

}
```

LAB-5

1. Father Son Age Relationship

```
class WrongAge Exception extends Exception {
    public WrongAge Exception (String message) {
        super (message);
    }
}

class Father {
    int age;
    public Father (int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAge Exception ("Age cannot
            be negative");
        }
        this.age = age;
    }
}

class Son extends Father {
    int SonAge;
    public Son (int FatherAge, int SonAge)
        throws WrongAge Exception {
        super (FatherAge);
        if (SonAge >= FatherAge) {
            throw new WrongAge Exception
("Son's age cannot be greater than or
equal to father's age");
        }
    }
}
```

```
        this.sonAge = sonAge;
    }
}

public class InheritanceExceptionHandling {
    public static void main(String[] args) {
        try {
            int fatherAge = Integer.parseInt(args[0]);
            int sonAge = Integer.parseInt(args[1]);
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Father's Age: " + fatherAge);
            System.out.println("Son's age: " + sonAge);
        } catch (NumberFormatException e) {
            System.out.println("Please enter valid
            ages as integers.");
        } catch (WrongAgeException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

3.

```
class DisplayThread extends Thread {
    private final String message;
    private final int interval;
    public DisplayThread(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }
}
```

```
@ Override
public void run() {
    try {
        while (true) {
            System.out.println(message);
            Thread.sleep(interval * 1000);
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
}

public class TwoThreadsDemo {
    public static void main(String[] args) {
        DisplayThread thread1 = new DisplayThread
            ("BMS College Of Engineering", 10);
        DisplayThread thread2 = new DisplayThread
            ("CSE", 2);

        thread1.start();
        thread2.start();
    }
}
```

Create a package CIE which has two classes: Student and Internals class personal has members like USN, name, sem and class Internals has an array that stores the internal marks scored in 5 courses of current semester of the student. Create another package SEE which has the class external which extends class student. This class has an array that stores the SEE marks scored in 5 courses. Import the 2 packages that declares the final marks of a student in all 5 courses.

```
package CIE
public class personal
        public String USN;
        public String name;
        public int sem;
        public personal (String usn, String name, int sem)
        { this. usn = usn;
          this. name = name;
          this. sem = sem; ) }

package CIE;
public class 'internals extends personal
public int [] internal marks;
public internals (String USN, String name, int sem,
                  int [] internal marks)
        { super (usn, name, sem);
          this. internalMarks = internalMarks; } }
```

```
package SEE;
import CIE.personal;
public class External extends Personal
{ public int [] external marks;
  public external (String USN, String name, int sem,
  int [] external marks)
{ super (USN, name, sem);
  this.externalMarks = externalMarks; } }

import java.util.Scanner;
import java.util.Arrays;
import CIE.internals;
import CIE.personals;
import SEE.External;
public class Main
{
  public static void main (String args [])
{
  Scanner S = new Scanner (System.in);
  System.out.println ("Enter the number of Student:");
  int n = S.nextInt ();
  Internals [] internalsData = new Internals [n];
  External [] externalsData = new External [n];
for (i=0; i<n; i++)
{
System.out.println ("Enter details for Student "+(i+1));
System.out.println ("Enter USN:");
String USN = S.nextLine();
System.out.println ("Enter name:");
String name = S.nextLine();
System.out.println ("Enter semester");
int sem = S.nextInt(); }
```

```
System.out.println("Enter internal marks for 5 subjects:");
int[] internalMarks = new int[5];
for (int j=0; j<5; j++)
{ SOP ("Subject" + (j+1));
  internalMarks[j] = s.nextInt();
}
SOP ("Enter External Marks for 5 subjects:");
int[] externalMarks = new int[5];
for (int j=0; j<5; j++) {
      externalMarks[j] = s.nextInt();
}
InternalsData[i] = new Internals(USN, name, sem,
                    internalMarks);
ExternalsData[i] = new Externals(USN, name, sem,
                    externalMarks);
}
for (int i=0; i<n; i++) {
  SOP ("Student" + (i+1));
  SOP ("Internal marks:" + arrayToString
  (InternalsData[i].internalMarks));
SOP ("External Marks" + arrayToString
                    (ExternalData[i].externalMarks));
SOP ("Total Marks (CIE+SEE): " + calculateTotal
                    Marks());}
Public static int calculateTotalMarks
(int[] internalMarks, int[] externalMarks)
```

```
{ int total = 0;
  for (int i = 0 ; i < internalMarks.length ; i++)
  {
    total += internalMarks[i] + externalMarks[i];
    return total; }
}
```

OUTPUT :-

```
enter the number of students : 2
→ enter details for student 1
  enter USN : 1BM22CS295
  enter name : sulaiman Ahmad
  enter semester : 1
  enter Internal Marks for 5 subjects
  subject 1 : 45
  subject 2 : 48
  subject 3 : 43
  subject 4 : 49
  subject 5 : 50

enter external Marks for 5 subjects
subject 1 : 45
subject 2 : 44
subject 3 : 48
subject 4 : 49
subject 5 : 47
```

→ enter details for student 2

Enter USN : 1RM22CS296

Enter name : Sufiah

Enter semesta : 1

Enter Internal Marks for 5 subjects

subject 1 : 44

sub 2 : 43

sub 3 : 47

sub 4 : 41

sub 5 : 49

Enter external Marks for 5 subjects

sub 1 : 46

sub 2 : 42

sub 3 : 48

sub 4 : 44

sub 5 : 49

Student 1 :

Internal Marks : [47, 45, 43, 49, 50]

External Marks : [47, 44, 45, 49, 47]

Total Marks (CIE+SEE) : 468

Student 2 :

External Marks : [44, 43, 47, 41, 49]

External Marks : [46, 42, 48, 44, 49]

Total Marks (CIE + SEE) 441

1. creating label, button and TextField in a Frame using AWT.

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

public class AWTExample extends WindowAdapter {

    Frame F;
    AWTExample() {
    f = new Frame();
    f.addWindowListener(this);
    Label l = new Label("Employee id:");
    Button b = new Button("Submit");
    TextField t = new TextField();
    l.setBounds(20, 80, 80, 30);
    t.setBounds(20, 100, 80, 30);
    b.setBounds(100, 100, 80, 30);
    f.add(b);
    f.add(l);
    f.add(t);
    f.setSize(400, 300);
    f.setTitle("Employee Info");
    f.setLayout(null);
    f.setVisible(true);
}
public void windowClosing(WindowEvent e) {
    System.exit(0);
}
```

```
public static void main (String [] args) {
AWTExample awt_obj = new AWTExample();
}
}
```

2. Create a button and add an action listener for mouse click.

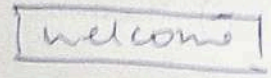
```
import java.awt.*;
import java.awt.event.*;
public class EventHandling extends WindowAdapter
implements ActionListener {
Frame f ;
TextField tf;
EventHandling() {
f = new Frame ();
f.addWindowListener(this);
TextField();
tf.setBounds(60,50,170,20);
Button b = new Button ("Click Me");
b.setBounds(100,120,30,30);
b.addActionListener(this);
f.add(b);
f.add(tf);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
```

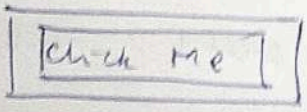
```
{ if. setText ("Welcome "); }

public void windowClosing (WindowEvent e)
{ System.exit (0); }

public static void main (String args[])
{
    new EventHandling (); }
}
```

OUTPUT :





6).
```
import java.io.*;
public class ByteArrayInput
{
  P sum (String args[]) throws IO exception
  {
    byte [] buf = { 35, 36, 37, 38 };
    Byte Array Input Stream byt = new
                BA Input Stream (buf);

    int k = 0;
    while ( ch = byt.read() ! = -1 )
    {
      char ch = (char) k;
      SO P ("ASCII value :" + k + "
                    char is :" + ch);
    }.
```

OUTPUT:-

ASCII value : 35 character is #
" : 36 " " is $
" : 37 " " is %
" : 38 " " is &

Q1. import java.io.*;
public class ByteArray_ex
{
psvm (String[] args) throws exception
{
FileOutputStream fout1 = new FileOutputStream ("example.txt");
FileOutputStream fout2 = new FileOutputStream ("example2.txt");
ByteArrayOutputStream bout = new ByteArrayOutputStream ();
bout.write (65);
bout.writeTo (fout1);
bout.writeTo (fout2);
bout.flush ();
bout.close ();
SOP ("Success");
}
}

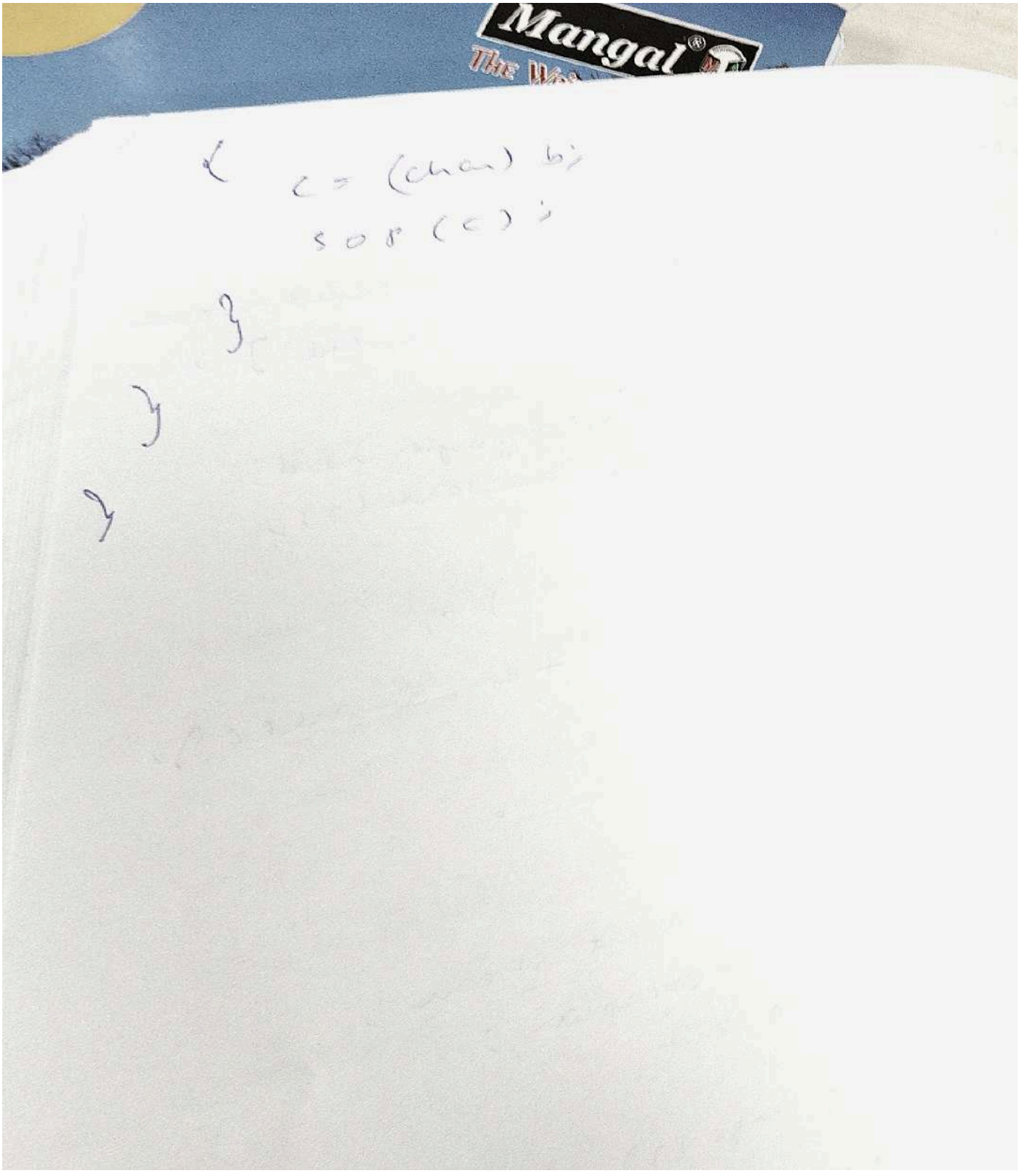
OUTPUT:

Success

8). public class FileEx
{ psvm (String args[]) throws IO exception
{
FileInputStream fin = new FileInputStream
("example.txt");

int content;
System.out.println("Remaining Bytes read"
+ fin.available());
content = fin.read();
SOP (char)content + " ");
SOP ("Remaining bytes:" + fin.available());
}
}

9). import java.io.FileInputStream;
import java.io.IOException;
public class FileOut
{
psvm (String args[]) throws IOException
{
FileInputStream fin = new FileInputStream("input.txt");
byte [] bytes = new byte [20];
int i;
char c;
i = fin.read (bytes);
SOP ("Number of bytes read:" + i);
SOP ("Bytes Read");
for (byte b: bytes)




```
{   c = (char) bi
    sop(c);
  }
 }
}
```

2]. WAP to print "Hello World"

```
class example {
    public static void main (String args[])
    {
        SOP ("Hello world");
    }
}
```

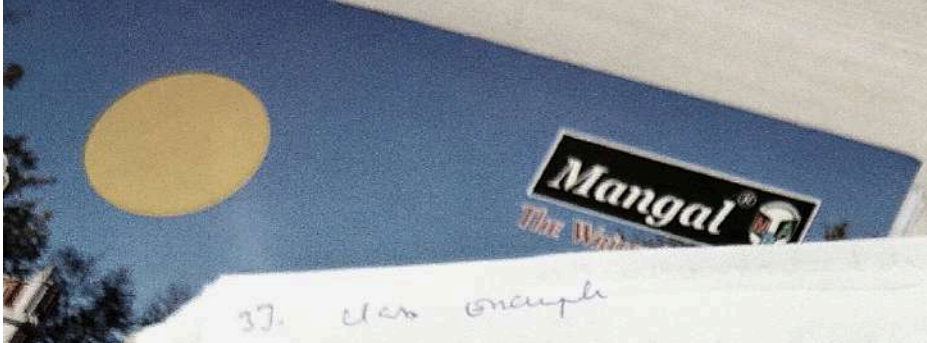
OUTPUT

Hello world.

2]

```
class Except
{
    PSVM (String args[])
    {
        int x = 10;
        int y = 20;
        int sum = x+y;
        SOP (sum);
    }
}
```

OUTPUT : 30



37. class Example

```
{
    psvm (String args[])
    {
        int x = 10, y = 5, diff;
        diff = x - y;
        SOP (diff);
    }
}
```

OUTPUT

5

class Example

```
{
    public static void main (String args[])
{
    int x = 7, y = 6, mod;
    mod = x * y;
    SOP (mod);
}
}
```

OUTPUT

30

4). check if number is prime
class example

```
{
    psvm (String args[])
    {
        int n = 29, flag = 0;
        for (int i = 2 ; i <= n/2; i++)
        {
            if (n%i == 0)
            {
                flag = 1; break;
            }
        }
        if (flag == 0)
        { System.out.println(n+" is a prime
                                number");
        }
        else
        { SOP (n+" is not a prime number");
        }
    }
```

OUTPUT:

29 is a prime number.