Lab 1

Lab 1

DATE: 15/12/23 PAGE: 1

1) Print 'Hello World'

→
```
class Example
{
    public static void main(String a[])
    {
        System.out.print("Hello World");
    }
}
```

Output:
Hello World

2) Addition

→
```
class Example
{
    public static void main(String a[])
    {
        int x=10;
        int y=20;
        int sum=x+y;
        System.out.print(sum);
    }
}
```

Output:
30

3) Subtraction

```
class Example
{
    public static void main(String a[])
    {
        int x=10, y=5, diff;
        diff=x-y;
        System.out.print(diff);
    }
}
```

Output:
5

4) Multiplication

```
class Example
{
    public static void main(String a[])
    {
        int x=5, y=6, prod;
        prod=x*y;
        System.out.print(prod);
    }
}
```

Output
30

5) Division

DATE: PAGE: 3

→ class Example
{
public static void main (String a[])
{
int x=10, y=5, quotient;
quotient = x/y;
System.out.print(quotient);
}
}

Output:
2

1) Check if a number is prime

→ class Example
{
public static void main (String a[])
{
int n=21, 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");
}
}
}

Output:
29 is a prime number

7) Fibonacci Series

→ class Example
{
public static void main (String a[])
{
int n=10, term1=0, term2=1;
System.out.println("Fibonacci series till" + n + "terms");
for (int i=1; i<=n; i++)
{
System.out.print(term1 + " ");
int term3 = term1 + term2;
term1 = term2;
term2 = term3;
}
}
}

Output:
Fibonacci series till 5 terms:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34

8) Roots of a quadratic equation

→ class Example
{
public static void main (String arg [])

```
double a=1, b=-7, c=10;
double root1, root2;
double determinant = b*b - 4*a*c;
if (determinant > 0)
{
    root1 = (-b + Math.sqrt(determinant));
    root2 = (-b - Math.sqrt(determinant));
    System.out.print("root1 = " + root1 + " and
                      root2 = " + root2);
}
else if (determinant == 0)
{
    root1 = root2 = -b/(2*a);
    System.out.print("root1 = root2 = " + root1);
}
else
{
    double real = -b/(2*a);
    double imaginary = Math.sqrt(-determinant)/(2*a);
    System.out.print("root1 = " + real + "+" + imaginary + "i");
    System.out.print
    System.out.print("root2 = " + real + "-" + imaginary + "i");
}
}
}
```

Output:

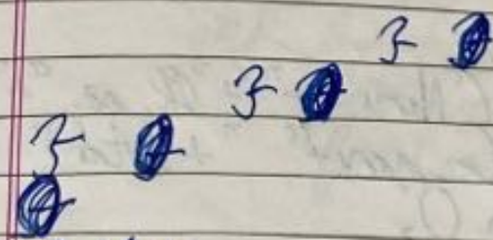
root1 = 2 and root2 = 5

Lab 2

1) Write a program to overload the method print that prints sum of n natural numbers when one variable is passed, and prints the prime numbers in a given range when 2 parameters are passed.

→
```
class Overload {
    void print(int n) {
        int sum = 0;
        for (int i = 1; i <= n; i++) {
            sum = sum + i;
        }
        System.out.println("Sum of " + n + " natural
                numbers is " + sum);
    }
    void print(int m, int n) {
        System.out.println("Prime numbers in the range are");
        for (int i = m; i <= n; i++) {
            int flag = 0;
            for (int j = 2; j <= i/2; j++) {
                if (i % j == 0) {
                    flag = 1;
                    break;
                }
            }
            if (flag == 0)
                System.out.println(i);
        }
    }
}
```

```
class Overload Demo {
    public static void main (String[] args) {
        Overload o = new Overload ();
        o.print (5);
        o.print (7,13);
```



Output:

Sum of 5 natural numbers is 15
Prime numbers in the range are:
7
11
13

2) Write a java program to create a class Grocery that has the variables c_name and c_phno. Create a method to accept 3 parameters to specify quantity of dal, quantity of pulses and quantity of sugar. Also the method to return the total price. Display the name, ph-no and total bill of 3 customers.

→

```
class Grocery {
    String c_name;
    String c_ph;
    double total;
    Grocery (String c_name, String c_ph) {
        this.c_name = c_name;
        this.c_ph = c_ph;
    }
```

```
void calc(double q_dal, double q_pulses,
          double q_sugar) {
    total = q_dal*100 + q_pulses*80 + q_sugar*80;
}
void display()
{
    System.out.println("Name "+" "+"Ph. no. "+" "+"Total");
    System.out.println(c_name+" "+total);
    System.out.println();
}
}
class Grocery {
    public static void main(String[] args) {
        Grocery g1=new Grocery("Rana","8060302010");
        Grocery g2=new Grocery("Shana","7689732510");
        Grocery g3=new Grocery("Bhana","9632587412");
        g1.calc(2,2,1);
        g1.display();
        g2.calc(3,5,2);
        g2.display();
        g3.calc(1,1,0,5);
        g3.display();
    }
}
```

Output:

Name	Phone	total
Rana	806	410
Shana	721	800
Bhana	967	305

5) Write a java program to calculate roots of a quadratic equation. Use appropriate methods to take input and calculate the roots.

```
class Overload {
    void print(int n) {
        int sum=0;
        for (int i=1; i<=n; i++) {
            sum=sum+i;
        }
        System.out.println("Sum of "+n+" natural numbers is "+sum);
    }
}
```

```
import java.util.Scanner;
class Quad {
    int a, b, c;
    double root1, root2, d;
    Scanner s=new Scanner(System.in);
    void input()
    {
        System.out.println("Quadratic equation is in the form: ax^2+bx+c");
        System.out.print("Enter a:");
        a=s.nextInt();
        System.out.print("Enter b:");
        b=s.nextInt();
        System.out.print("Enter c:");
        c=s.nextInt();
    }
    void discriminant() {
```

```
    d=(b*b)-(4*a*c);
}
void calculateRoots() {
    if (d>0)
    {
        System.out.println("Roots are real and unequ
        root1=(-b+Math.sqrt(d))/(2*a);
        root2=(-b-Math.sqrt(d))/(2*a);
        System.out.println("First root is:"+root1);
        System.out.println("Second root is:"+root2);
    }
    else if (d==0)
    {
        System.out.println("Roots are real and equa
        root1=(-b+Math.sqrt(d))/(2*a);
        System.out.println("Root:"+root1);
    }
    else
    {
        System.out.println("No real solutions. Roots
                            are imaginary");
        double real=-b/(2*a);
        double imaginary=Math.sqrt(-d)/(2*a);
        System.out.println("The equation has two
        complex roots:"+real+"+"+imaginary+"i
        and"+real+"-"+imaginary+"i");
    }
}
}
class main {
    public static void main(String[] args) {
```

```
Quad q = new Quad();
q.input();
q.discriminant();
q.calculateRoots();
}
}
```

Output:

Enter variable 1

-4

4

roots are equal

r=2

8 8 12 24

Lab 1

1) Create a class Book that contains four members: name, author, price and num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

→
```
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 = "Author name:" + this.author + "\n";
        price = "Price:" + this.price + "\n";
        numPages = "number of pages:" + this.numPages + "\n
```

```
        return name + author + price + numPages;
    }
}
class Main
{
    public static void main(String args[])
    {
        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, numPages);
        }
        for(int i=0; i<n; i++)
```

System.out.println("Book "+(i+1)+":\n"+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
Enter the price of the book: 1000
Enter the number of pages of the book: 500

Book 2:
Enter the name of the book: Tales_Of_Akbar_And_Birbal
Enter the author of the book: Birbal
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 2:
Book name: Tales_Of_Akbar_And_Birbal
Author: Birbal
Price: 900
Number of pages: 400

2)

DATE: PAGE: 15

2) Write a Java program to create a class Student with members USN, name, marks (6 subjects). Include methods to accept student details and marks, also include a method to calculate the percentage and display appropriate details. (Array of student object to be created)

→
```
import java.util.Scanner;
class Student
{
    String USN;
    String name;
    double[] marks = new double[6];
    // Method to accept student details and marks
    void inputDetails()
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        USN = 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();
        }
    }
    // Method to calculate the percentage
    double calculatePercentage()
    {
        double totalMarks = 0;
        for(double mark : marks)
        {
```

```
        totalMarks += mark;
    }
    return (totalMarks/6);
}
// Method to display student details
void displayDetails()
{
    System.out.println("\n Student Details:");
    System.out.println("USN:" + USN);
    System.out.println("Name:" + name);
    System.out.println("Percentage:" + calculatePercentage() +
                       "%");
}
}
class oop
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of students:");
        int numberOfStudents = scanner.nextInt();
        // Create an array of Student objects
        Student[] students = new Student[numberOfStudents];
        // Input details for each student
        for(int i=0; i< numberOfStudents; i++)
        {
            System.out.println("\n Enter details for
                               Student " + (i+1) + ":");
            students[i] = new Student();
            students[i].inputDetails();
        }
        // Display details for each student
```

```
        for (Student student : students)
        {
            student.displayDetails();
        }
    }
}
```

Output:

```
Enter the number of students: 2

Enter details for student 1:
Enter USN: 1BM22CS285
Enter name: ABC
Enter marks for 6 subjects:
subject 1: 78
subject 2: 87
subject 3: 90
subject 4: 99
subject 5: 79
subject 6: 80

Enter details for student 2:
Enter USN: 1BM22CS400
Enter name: XYZ
Enter marks for 6 subjects:
subject 1: 55
subject 2: 67
subject 3: 76
subject 4: 80
subject 5: 97
subject 6: 76
```

DATE:

PAGE:

Student details:
USN: 1BM22CS285
Name: ABC
Percentage: 85.5%

Student details:
USN: 1BM22CS400
Name: XYZ
Percentage: 71.16666666666667%

19/1/24

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~~ 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;
    shape (int length, int width)
    {
        this.length = length;
        this.width = width;
    }
    abstract void printArea();
}
class rectangle extends shape
{
    rectangle (int length, int width)
    {
        super (length, width);
    }
    void printArea ()
    {
        int area = length * width;
        System.out.println("Rectangle area:" + area);
    }
```

```
}
class triangle extends shape
{
    Triangle (int length, int width)
    {
        super (length, width)
    }
    void printArea()
    {
        double area = 0.5 * length * width;
        System.out.println("Triangle area:" + area);
    }
}
class circle extends shape
{
    circle (int length)
    {
        super (length, 0);
    }
    void printArea()
    {
        double area = Math.PI * length * length;
        System.out.println("circle area:" + area);
    }
}
class Main
{
    public static void main (String args[])
    {
        Rectangle r = new Rectangle (5, 10);
        r.printArea();
```

```
Triangle t = new Triangle (4,6);
t.printArea();
Circle c = new Circle (3);
c.printArea();
    }
}
```

Output:
Rectangle area: 50
Triangle area: 12.
Circle area: 28.26

2) Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders 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, account number and type of account. From this ~~class~~ derive the classes Cur-acct and Sav-acct 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.

for

Lab 5

D) WAP that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age < 0. In Son class, implement a constructor that cases both Father and son's age and throws an exception if son's age is >= father's age.

→ Father-Son Age Relationship

```
class Father
{
    public int age;
    Father(int age)
    {
        if (age > 0)
        {
            throw new IllegalArgumentException("Age cannot
                                                be -ve");
        }
        this.age = age;
    }
}
public class Son extends Father
{
    public int SonAge;
    public Son(int fatherAge, int sonAge)
    {
```

```
super(fatherAge);
if(sonAge >= fatherAge)
{
    throw new IllegalArgumentException("Son's age
                                cannot be >= father's age");
}
this.sonAge = sonAge;
}
}
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        try
        {
            SOP("Enter father's age:");
            int fatherAge = scanner.nextInt();
            SOP("Enter son's age:");
            int sonAge = scanner.nextInt();
            Son son = new Son(fatherAge, sonAge);
            SOP("Father's age:" + son.age);
            SOP("Son's age:" + son.sonAge);
        }
        catch(IllegalArgumentException e)
        {
            SOP("Exception:" + e.getMessage());
        }
        scanner.close();
    }
}
```

```
}
}
```

Output:

```
Enter father's age: 20
Enter son's age: 40
Exception: Son's age cannot be >= father's age
```

2) BMSCE - CSE thread:

→

```
public class Main
{
    psvm (String [] Args)
    {
        Thread thread1 = new Thread (new Runnable ()
        {
            public void new()
            {
                while (true)
                {
                    SOP ("BMSCE");
                    try
                    {
                        Thread.sleep (10000);
                    }
                    catch (InterruptedException e)
                    {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}
```

```
});
Thread thread2 = new Thread(new Runnable()
{
    public void new()
    {
        while (true)
        {
            SOP("CSE");
            try
            {
                thread.sleep(2000);
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    }
});
thread1.start();
thread2.start();
}
}
```

3) CIE package.

→

```
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;
}
}
import java.util.Arrays;
public class Internals
{
    public int[] internalMarks;
    public Internals(int[] internalMarks)
    {
        this.internalMarks = internalMarks;
    }
}

package SEE;
import CIE.Personal;
public class External extends Personal
{
    public int[] seeMarks;
    public External(String usn, String name, int sem,
                    int[] seeMarks)
    {
        super(usn, name, sem);
        this.seeMarks = seeMarks;
    }
}
```

```
package for Main;
import java.util.Arrays;
import CIE.Internals;
import CIE.~~External~~Personal;
import ~~CIE~~ SEE.External;
public class Main
{
    public static void main(String[] args)
    {
        int n=3;
        Student[] students = new Student[n];
        for(int i=0; i<n; i++)
        {
            int[] internalMarks = {80, 75, 90, 85, 95};
            int[] seeMarks = {70, 80, 75, 90, 85};
            students[i] = new Student(new Personal("USN"+i,
                "Student"+i, 1), new Internals(internalMarks
            students[i] = new External("USN"+i, "Student"+i,
                seeMarks);
        }
        for(int i=0; i<students.length; i++)
        {
            Student student = students[i];
            SOP("Student:" + student.personal.name);
            SOP("Internal Marks:" + Arrays.toString(student.
                internals.internalMarks));
            SOP("SEE Marks:" + Arrays.toString(student.see
                seeMarks));
            SOP();
        }
}
```

```
static class Student
{
    public Personal personal;
    public Internals internals;
    public External see;
    public Student(Personal personal, Internals internals)
    {
        this.personal = personal;
        this.internals = internals;
    }
}
}
```

Output:

Student: Student 0
Internal Marks: [80, 75, 90, 85, 95]
SEE Marks: [70, 80, 75, 90, 85]

Student: Student 1
Internal Marks: [80, 75, 90, 85, 95]
SEE Marks: [70, 80, 75, 90, 85]

Student: Student 2
Internal Marks: [80, 75, 90, 85, 95]
SEE Marks: [70, 80, 75, 90, 85]

Lab 4

2) Develop a Java program to create a class Bank that maintains two kinds of account for it's customers, one called saving account and the other current account. The saving account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders 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, account number and type of account. From this derive the classes Cur-acct and Sav-acct 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) Permit withdrawal and update the balance.

Check for the minimum balance, impose penality if necessary and update the balance.

→ import java.util.Scanner;

DATE: PAGE: 33

```
class Account
{
    String customerName;
    long accountNumber;
    String accountType;
    double balance;
    Account(String customerName, long accountNumber,
            String accountType, double balance)
    {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = balance;
    }
    void deposit(double amount)
    {
        balance += amount;
        SOP("Deposit successful. Updated balance:" + balance);
    }
    void displayBalance()
    {
        SOP("Account Number:" + accountNumber);
        SOP("Balance:" + balance);
    }
}

class CurAcct extends Account
{
    double minBalance;
    double serviceCharge;
    CurAcct(String customerName, long accountNumber,
    double balance, double minBalance, double serviceCharge)
```

```
{ double balance
  super (customer Name, account Number, "Current",
         balance);
  this.minBalance = minBalance;
  this.serviceCharge = serviceCharge;
}
void withdraw (double amount)
{
  if ((balance - amount) >= minBalance)
  {
    balance -= amount;
    SOP("Withdrawal successful. Updated
        balance:" + balance);
  }
  else
  {
    SOP("Insufficient funds. Withdrawal not
        allowed.");
  }
}
void checkMinimumBalance()
{
  if (balance < minBalance)
  {
    balance -= serviceCharge;
    SOP("Minimum balance not maintained. Service
    charge imposed. Updated balance: " + balance);
  }
}
}
```

```
class SavAcct extends Account
{
    double interestRate;
    SavAcct(String customerName, long accountNumber,
            double balance, double interestRate)
    {
        super(customerName, accountNumber, "Savings",
              balance);
        this.interestRate = interestRate;
    }
    void computeInterest()
    {
        double interest = balance * (interestRate/100);
        balance += interest;
        SOP("Interest computed and deposited. Updated
             balance: " + balance);
    }
    void withdraw(double amount)
    {
        if (balance >= amount)
        {
            balance -= amount;
            SOP("Withdrawal successful. Updated balance:"
                + balance);
        }
        else
        {
            SOP("Insufficient funds. Withdrawal not
                 allowed.");
        }
    }
}
```

```
class Bank
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        // Example usage
        SavAcct currentAccount = new SavAcct("John Doe",
                                    123456789, 1000, 500, 20);
        currentAccount.deposit(500);
        currentAccount.displayBalance();
        currentAccount.withdraw(700);
        currentAccount.checkMinimumBalance();
        SOP();
        SavAcct savingsAccount = new SavAcct("Jane
                                    Smith", 987654321, 2000, 5);
        savingsAccount.deposit(1000);
        savingsAccount.displayBalance();
        savingsAccount.compliteInterest();
        savingsAccount.withdraw(300);
        scanner.close();
    }
}
```

Output:
Enter name, account no., balance, minimum balance, service charge:
John
123456789
2000
500
20

Deposit successful - Updated balance: 3000
Account number: 123456789
Balance: 3000
interest computed
Withdrawal successful. Updated balance: 2700

Lab 6

1) Creating label, button and text field 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 a 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);
        tf = new TextField();
        tf.setBounds(60, 50, 170, 20);
        Button b = new Button("click me");
        b.setBounds(100, 120, 80, 30);
        b.addActionListener(this);
        f.add(b); f.add(tf);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setVisible(true);
```

```
}
public void actionPerformed (ActionEvent e)
{
    tf.setText("Welcome");
}
public void windowClosing (WindowEvent e)
{
    System.exit(0);
}
public static void main (String args[])
{
    new EventHandling();
}
}
```

Programs on IO:

1) Example 1

→

```
import java.io.*;
public class ByteArrayInput
{
    public static void main (String [] args) throws IOException
    {
        byte [] buf = {35, 36, 37, 38};
        ByteArrayInputStream byt = new ByteArrayInputStream(buf);
        int k = 0;
        while ((k = byt.read()) != -1)
        {
            char ch = (char) k;
            System.out.println("ASCII value of character is: " + k + "
                Specified character is: " + ch);
```

777

4) Example 2

```
import java.io.*;
public class ByteArrayInput
{
    public static void main(String[] args) throws IOException
    {
        byte[] buf = {35, 36, 37, 38};
        ByteArrayInputStream byt = new ByteArrayInputStream(buf);
        int k=0;
        while ((k=byt.read())!=-1)
        {
            char ch=(char)k;
            SOP("ASCII value of character is:"+k+"
                Special character is:"+ch);
        }
    }
}
```

5) Example 3

```
public class FileEx
{
    public static void main(String a[]) throws IOException
    {
        FileInputStream fin=new FileInputStream("Example.txt");
        int content;
        SOP("Remaining bytes that can be read:"+fin.available());
        content=fin.read();
        SOP((char)content+" ");
        SOP(content+" ");
```

```
SOP("Remaining bytes that can be read:"+
    fin.available());
SOP("Remaining bytes that can be read:"+
    fin.available());
}
}
```

2) Example 4

→

```
import java.io.FileInputStream;
import java.io.IOException;
public class FileEx2
{
    public static void main(String a[]) throws IOException
    {
        FileInputStream fin=new FileInputStream("Example.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)b;
            SOP(c);
        }
    }
}
```

1) Output: "Employee Info"

Employee ID:
[ ] [Submit]

2) Output:

[Welcome]

[Click Me]

3) Output: ASCII value: 35 Character is: #
ASCII value: 36 Character is: $
ASCII value: 37 Character is: %
ASCII value: 38 Character is: &

4) Output: Success