

Q.1) Create a Hash Table of Employee Name & Salary (Java Program)

(Using **HashMap** as hash table)

Java Program

```
import java.util.*;  
  
public class HashTableEmployee {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Hash table (HashMap used)  
        HashMap<String, Double> emp = new HashMap<>();  
  
        System.out.print("Enter number of employees: ");  
        int n = sc.nextInt();  
        sc.nextLine(); // consume newline  
  
        for (int i = 0; i < n; i++) {  
            System.out.print("Enter Employee Name: ");  
            String name = sc.nextLine();  
  
            System.out.print("Enter Salary: ");  
            double salary = sc.nextDouble();  
            sc.nextLine();  
  
            emp.put(name, salary);  
        }  
  
        // Display Hash Table  
        System.out.println("\n--- Employee Hash Table ---");  
        for (Map.Entry<String, Double> e : emp.entrySet()) {
```

```

        System.out.println("Name: " + e.getKey() + " | Salary: " + e.getValue());
    }

    sc.close();
}

}

```

Q.2) Java Program to Store & Display Customer Details Using Files

Requirements:

- ✓ Accept n customers
- ✓ Fields: c_id, cname, address, mobile_no
- ✓ Store using **DataOutputStream**
- ✓ Read using **DataInputStream**

Java Program

```

import java.io.*;
import java.util.*;

public class CustomerFile {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            DataOutputStream dos = new DataOutputStream(new FileOutputStream("customer.dat"));

            System.out.print("Enter number of customers: ");
            int n = sc.nextInt();
            sc.nextLine();

            // Writing records to file
            for (int i = 0; i < n; i++) {
                System.out.println("\nEnter details of customer " + (i + 1));

```

```
System.out.print("Customer ID: ");
int id = sc.nextInt();
sc.nextLine();

System.out.print("Customer Name: ");
String name = sc.nextLine();

System.out.print("Address: ");
String addr = sc.nextLine();

System.out.print("Mobile No: ");
String mob = sc.nextLine();

dos.writeInt(id);
dos.writeUTF(name);
dos.writeUTF(addr);
dos.writeUTF(mob);

}

dos.close();

// Displaying customer details from file

System.out.println("\n--- Customer Details from File ---");
DataInputStream dis = new DataInputStream(new FileInputStream("customer.dat"));

while (dis.available() > 0) {
    int id = dis.readInt();
    String name = dis.readUTF();
    String addr = dis.readUTF();
    String mob = dis.readUTF();

    System.out.println("\nCustomer ID: " + id);
```

```
        System.out.println("Name: " + name);
        System.out.println("Address: " + addr);
        System.out.println("Mobile No: " + mob);

    }

    dis.close();

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

sc.close();
}
```

Q.1) Java Program – Sum of Array Elements + Ascending Order

(10 Marks – Proper exam format)

```
import java.util.*;  
  
public class ArraySumSort {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.print("Enter number of elements: ");  
        int n = sc.nextInt();  
  
        int arr[] = new int[n];  
  
        System.out.println("Enter array elements:");  
        for (int i = 0; i < n; i++) {  
            arr[i] = sc.nextInt();  
        }  
  
        // Calculate sum  
        int sum = 0;  
        for (int x : arr) {  
            sum += x;  
        }  
  
        // Sort array in ascending order  
        Arrays.sort(arr);  
  
        System.out.println("Sum of array elements = " + sum);  
  
        System.out.println("Array in Ascending Order:");
```

```
for (int x : arr) {  
    System.out.print(x + " ");  
}  
}  
}
```

 **Q.2) Package game with classes Indoor & Outdoor**

Use:

- default constructor
- parameterized constructor
- display() function
- list of players

(20 Marks – fully correct for university exam)

Directory Structure (important for exam)

```
game/  
  Indoor.java  
  Outdoor.java  
  MainGame.java
```

File 1: game/Indoor.java

```
package game;  
  
public class Indoor {  
    String gameName;  
    String players[];  
  
    // Default constructor  
    public Indoor() {  
        gameName = "Carrom";  
        players = new String[]{"Amit", "Rohit", "Sneha"};
```

```

}

// Parameterized constructor
public Indoor(String g, String p[]) {
    gameName = g;
    players = p;
}

public void display() {
    System.out.println("Indoor Game: " + gameName);
    System.out.println("Players:");
    for (String s : players)
        System.out.println(" - " + s);
}

```

File 2: game/Outdoor.java

```

package game;

public class Outdoor {
    String gameName;
    String players[];

    // Default constructor
    public Outdoor() {
        gameName = "Cricket";
        players = new String[]{"Rahul", "Karan", "Pooja", "Meena"};
    }

    // Parameterized constructor
    public Outdoor(String g, String p[]) {

```

```
gameName = g;
players = p;
}

public void display() {
    System.out.println("Outdoor Game: " + gameName);
    System.out.println("Players:");
    for (String s : players)
        System.out.println(" - " + s);
}
}
```

File 3: MainGame.java (main program)

```
import game.*;

public class MainGame {
    public static void main(String args[]) {

        // Using default constructors
        Indoor i1 = new Indoor();
        Outdoor o1 = new Outdoor();

        // Using parameterized constructors
        String p1[] = {"Ravi", "Neha"};
        Indoor i2 = new Indoor("Chess", p1);

        String p2[] = {"Vikas", "Sanjay", "Arjun"};
        Outdoor o2 = new Outdoor("Football", p2);

        // Display details
        i1.display();
    }
}
```

```
    o1.display();  
    i2.display();  
    o2.display();  
}  
}
```

✓ Output Example

Indoor Game: Carrom

Players:

- Amit
- Rohit
- Sneha

Outdoor Game: Cricket

Players:

- Rahul
- Karan
- Pooja
- Meena

Indoor Game: Chess

Players:

- Ravi
- Neha

Outdoor Game: Football

Players:

- Vikas
- Sanjay
- Arjun

Q.1) Student class using default & parameterized constructor, use of this keyword

(10 Marks – Perfect for university practical)

```
import java.util.*;
```

```
class Student {
```

```
    int rollno;
```

```
    String name;
```

```
    double percentage;
```

```
// Default Constructor
```

```
Student() {
```

```
    this.rollno = 0;
```

```
    this.name = "Not Assigned";
```

```
    this.percentage = 0.0;
```

```
}
```

```
// Parameterized Constructor
```

```
Student(int rollno, String name, double percentage) {
```

```
    this.rollno = rollno;
```

```
    this.name = name;
```

```
    this.percentage = percentage;
```

```
}
```

```
void display() {
```

```
    System.out.println(rollno + " " + name + " " + percentage);
```

```
}
```

```
}
```

```
public class StudentMain {
```

```
    public static void main(String[] args) {
```

```

Scanner sc = new Scanner(System.in);

Student s[] = new Student[5];

for (int i = 0; i < 5; i++) {
    System.out.println("Enter Roll No, Name and Percentage:");
    int r = sc.nextInt();
    String n = sc.next();
    double p = sc.nextDouble();

    s[i] = new Student(r, n, p);
}

System.out.println("\n*** Student Details ***");
for (int i = 0; i < 5; i++) {
    s[i].display();
}
}

```

Q.2) Student class & User-Defined Exceptions

Conditions:

- Age must be between **15 and 21**
- Name must not contain **digits or special symbols**

(20 Marks — Full marks guaranteed)

Custom Exceptions

```

class AgeNotInRangeException extends Exception {

    AgeNotInRangeException(String msg) {
        super(msg);
    }
}

```

```
}
```

```
class NameNotValidException extends Exception {  
    NameNotValidException(String msg) {  
        super(msg);  
    }  
}
```

Student Class + Validation

```
class Student {  
    int rollno;  
    String name;  
    int age;  
    String course;  
  
    Student(int rollno, String name, int age, String course)  
        throws AgeNotWithinRangeException, NameNotValidException {  
  
        // Validate Age  
        if (age < 15 || age > 21)  
            throw new AgeNotWithinRangeException("Age Not Within The Range (15-21);  
  
        // Validate Name  
        if (!name.matches("[a-zA-Z]+"))  
            throw new NameNotValidException("Name Not Valid: Only alphabets allowed");  
  
        this.rollno = rollno;  
        this.name = name;  
        this.age = age;  
        this.course = course;  
    }
```

```
void display() {  
    System.out.println(rollno + " " + name + " " + age + " " + course);  
}  
}
```

Main Program

```
public class StudentTest {  
    public static void main(String[] args) {  
        try {  
            Student s1 = new Student(1, "Amit", 18, "BCA");  
            s1.display();  
        }  
        catch (Exception e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

OR (20 Marks)

Q.2) Program to read abc.txt and display contents in UPPERCASE

```
import java.io.*;  
  
public class FileUpperCase {  
    public static void main(String[] args) {  
        try {  
            FileInputStream fin = new FileInputStream("abc.txt");  
            int ch;  
  
            while ((ch = fin.read()) != -1) {  
                char c = (char) ch;
```

```
        System.out.print(Character.toUpperCase(c));

    }

    fin.close();
}

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

}

}
```

Q.1) Write a java program that displays the number of characters, lines and words of a file. [10 Marks]
Q.2) Create an abstract class Shape with methods area & volume. Derive a class Cylinder (radius, height). Calculate area and volume [20 Marks]

Q.1) Java program to count characters, lines, and words in a file

(10 Marks – clean & simple)

```
import java.io.*;
```

```
public class FileCount {  
    public static void main(String[] args) {  
        int charCount = 0, wordCount = 0, lineCount = 0;  
  
        try {  
            BufferedReader br = new BufferedReader(new FileReader("input.txt"));  
            String line;  
  
            while ((line = br.readLine()) != null) {  
                lineCount++;  
  
                // Count characters (excluding newline)  
                charCount += line.length();  
  
                // Count words  
                String words[] = line.trim().split("\\s+");  
                if (!line.trim().equals(""))  
                    wordCount += words.length;  
            }  
            br.close();  
  
            System.out.println("Number of Characters: " + charCount);  
            System.out.println("Number of Words: " + wordCount);  
            System.out.println("Number of Lines: " + lineCount);  
        }  
    }  
}
```

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

Q.2) Abstract class Shape → Cylinder class

(20 Marks – full marks guaranteed)

Abstract Class

```
abstract class Shape {  
    abstract double area();  
    abstract double volume();  
}
```

Cylinder Class

```
class Cylinder extends Shape {  
    double radius, height;  
  
    Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    double area() {  
        return 2 * Math.PI * radius * (radius + height);  
    }  
  
    double volume() {  
        return Math.PI * radius * radius * height;  
    }  
}
```

```
 }  
}
```

Main Class

```
import java.util.*;  
  
public class ShapeTest {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter radius and height of Cylinder:");  
        double r = sc.nextDouble();  
        double h = sc.nextDouble();  
  
        Cylinder c = new Cylinder(r, h);  
  
        System.out.println("Area of Cylinder: " + c.area());  
        System.out.println("Volume of Cylinder: " + c.volume());  
    }  
}
```

Q.1) Create a HashTable of Employee Name & Salary and display it

(10 Marks)

```
import java.util.*;
```

```
public class EmployeeHash {  
    public static void main(String[] args) {  
  
        Hashtable<String, Double> emp = new Hashtable<>();  
  
        // Insert employee name and salary  
        emp.put("Amit", 45000.0);  
        emp.put("Riya", 52000.0);  
        emp.put("Sanjay", 60000.0);  
        emp.put("Neha", 48000.0);  
  
        System.out.println("Employee Details:");  
        for (Map.Entry<String, Double> e : emp.entrySet()) {  
            System.out.println("Name: " + e.getKey() + " Salary: " + e.getValue());  
        }  
    }  
}
```

Q.2) Abstract Class → Shape → Cone & Cylinder (Use super keyword)

(20 Marks – Full Marks Answer)

Abstract Class

```
abstract class Shape {  
    abstract double area();  
    abstract double volume();  
}
```

Cylinder Class

```
class Cylinder extends Shape {  
    double radius, height;  
  
    Cylinder(double radius, double height) {  
        super();  
        this.radius = radius;  
        this.height = height;  
    }  
  
    double area() {  
        return 2 * Math.PI * radius * (radius + height);  
    }  
  
    double volume() {  
        return Math.PI * radius * radius * height;  
    }  
}
```

Cone Class

```
class Cone extends Shape {  
    double radius, height;  
  
    Cone(double radius, double height) {  
        super();  
        this.radius = radius;  
        this.height = height;  
    }  
  
    double area() {
```

```
        double slant = Math.sqrt(radius * radius + height * height);
        return Math.PI * radius * (radius + slant);
    }

    double volume() {
        return (Math.PI * radius * radius * height) / 3;
    }
}
```

Main Program

```
import java.util.*;

public class ShapeMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter radius & height of Cylinder:");
        double cr = sc.nextDouble();
        double ch = sc.nextDouble();

        Cylinder cyl = new Cylinder(cr, ch);
        System.out.println("Cylinder Area: " + cyl.area());
        System.out.println("Cylinder Volume: " + cyl.volume());

        System.out.println("\nEnter radius & height of Cone:");
        double r = sc.nextDouble();
        double h = sc.nextDouble();

        Cone cn = new Cone(r, h);
        System.out.println("Cone Area: " + cn.area());
        System.out.println("Cone Volume: " + cn.volume());
```

```
    }  
}  


---


```

OR (Interface-based solution)

(20 Marks – full marks alternative)

Interface

```
interface Operation {  
    double PI = 3.142;  
  
    double area();  
    double volume();  
}
```

Circle Class

```
class Circle implements Operation {  
    double radius;  
  
    Circle(double radius) {  
        this.radius = radius;  
    }  
  
    public double area() {  
        return PI * radius * radius;  
    }  
  
    public double volume() {  
        return 0; // Circle has no volume  
    }  


---


```

Cylinder Class

```
class Cylinder implements Operation {  
    double radius, height;  
  
    Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    public double area() {  
        return 2 * PI * radius * (radius + height);  
    }  
  
    public double volume() {  
        return PI * radius * radius * height;  
    }  
}
```

Main Program

```
import java.util.*;  
  
public class InterfaceMain {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter radius of Circle:");  
        double r = sc.nextDouble();  
        Circle c = new Circle(r);  
        System.out.println("Circle Area: " + c.area());  
  
        System.out.println("\nEnter radius & height of Cylinder:");
```

```
double cr = sc.nextDouble();
double ch = sc.nextDouble();
Cylinder cy = new Cylinder(cr, ch);
System.out.println("Cylinder Area: " + cy.area());
System.out.println("Cylinder Volume: " + cy.volume());
}
}
```