Slip 1

Q1. Write a program to accept a number from user and generate multiplication table of a number.

```
import java.util.Scanner;
public class s1q1
{
public static void main(String[] args)
{
int ans;
Scanner sc=new Scanner(System.in);
System.out.println("Enter a number : ");
int n=sc.nextInt();
for(int i=1;i<=10;i++)
{
ans=n*i;
System.out.println(ans);
}
}
}
```

Q2. Construct a linked List containing names of colours: red, blue, yellow and orange. Then extend the program to do the following: i. Display the contents of the List using an Iterator ii. Display the contents of the List in reverse order using a ListIterator iii. Create another list containing pink and green. Insert the elements of this list between blue and yellow.

```
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.Iterator;

public class ColorList {
   public static void main(String[] args) {
      // Step 1: Create a LinkedList with colors
      LinkedList<String> colors = new LinkedList<>();
      colors.add("red");
      colors.add("blue");
      colors.add("yellow");
      colors.add("orange");

      // Step 2: Display the contents of the list using an Iterator
      System.out.println("Displaying the contents of the list using Iterator:");
      Iterator<String> iterator = colors.iterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      // Step 3: Display the contents of the list in reverse order using a ListIterator
      System.out.println("\nDisplaying the contents of the list in reverse order using ListIterator:");
```

```
        ListIterator<String> listIterator = colors.listIterator(colors.size()); // Start at the end of the list
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }

        // Step 4: Create another list with "pink" and "green"
        LinkedList<String> newColors = new LinkedList<>();
        newColors.add("pink");
        newColors.add("green");

        // Step 5: Insert new colors between "blue" and "yellow"
        int indexOfYellow = colors.indexOf("yellow"); // Find the index of "yellow"
        colors.addAll(indexOfYellow, newColors); // Insert new colors before "yellow"

        // Step 6: Display the modified list
        System.out.println("\nModified list after inserting 'pink' and 'green' between 'blue' and
'yellow':");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

Slip 2

Q1. Write a program to accept ‘n’ integers from the user & store them in an Array List collection. Display the elements of Array List.

```
import java.util.*;

class Qno1 {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of integers:");
        int no = sc.nextInt();

        ArrayList<Integer> al = new ArrayList<>();
        System.out.println("Add integers:");
        for (int i = 0; i < no; i++) {
            int n = sc.nextInt();
            al.add(n);
        }

        System.out.println("Elements in ArrayList:");
        for (int n : al) {
            System.out.println(n);
        }
    }
```

```
}
```

Q2. Define a class MyNumber having one private integer data member. Write a default constructor initialize it to 0 and another constructor to initialize it to a value. Write methods isNegative, isPositive, isOdd, iseven. Use command line argument to pass a value to the object and perform the above operations.

```
class MyNumber
{
int n;
public MyNumber()
{
n=0;
}

public MyNumber(int x)
{
n=x;
}

public void isNegative()
{
if(n<0)
System.out.println("Number is negative");
}

public void isPositive()
{
if(n>=0)
System.out.println("Number is positive");
}

public void isOdd()
{
if(n%2!=0)
System.out.println("Number is odd");
}

public void isEven()
{
if(n%2==0)
System.out.println("Number is even");
}

class s2q2
{
```

```
public static void main(String[] args)
{
int n1=Integer.parseInt(args[0]);
System.out.println(n1);
MyNumber m1=new MyNumber(n1);
m1.isNegative();
m1.isPositive();
m1.isOdd();
m1.isEven();
}
}
}
```

Slip 3

Q1. Write a program to accept the 'n' different numbers from user and store it in array. Also print the sum of elements of the array.

```
import java.util.Scanner;
public class s3q1
{
int sum=0;
Scanner sc=new Scanner(System.in);
System.out.println("Enter the number of array elements : ");
int n=sc.nextInt();
int a[]=new int[n];
for(int i=0;i<a.length;i++)
{
sum=sum+a[i];
}
System.out.println("Sum of array elements :"+sum);
}
}
```

Q2. Write a program to create class Account (accno, accname, balance). Create an array of 'n' Account objects. Define static method “sortAccount” which sorts the array on the basis of balance. Display account details in sorted order.

```
class Account {
   int accno;
   String accname;
   double balance;

   public Account(int accno, String accname, double balance) {
      this.accno = accno;
      this.accname = accname;
      this.balance = balance;
```

```
    }

    public static void sortAccount(Account[] accounts) {
        for (int i = 0; i < accounts.length - 1; i++) {
            for (int j = 0; j < accounts.length - i - 1; j++) {
                if (accounts[j].balance > accounts[j + 1].balance) {
                    // Swap accounts[j] and accounts[j+1]
                    Account temp = accounts[j];
                    accounts[j] = accounts[j + 1];
                    accounts[j + 1] = temp;
                }
            }
        }
    }

    public void display() {
        System.out.println("Account Number: " + accno);
        System.out.println("Account Name: " + accname);
        System.out.println("Balance: " + balance);
        System.out.println();
    }
}

public class Main2 {
    public static void main(String[] args) {
        // Create an array of Account objects
        Account[] accounts = new Account[5];
        accounts[0] = new Account(101, "John Doe", 1500);
        accounts[1] = new Account(102, "Jane Doe", 2000);
        accounts[2] = new Account(103, "Bob Smith", 1200);
        accounts[3] = new Account(104, "Alice Johnson", 1800);
        accounts[4] = new Account(105, "Eve Johnson", 1000);

        // Sort the accounts based on balance
        Account.sortAccount(accounts);

        // Display account details in sorted order
        System.out.println("Account Details (Sorted by Balance):");
        for (Account account : accounts) {
            account.display();
        }
    }
}
```

Slip 4

Q1. Write a program to accept the user name and greets the user by name. Before displaying the user's name, convert it to upper case letters. For example, if the user's name is Raj, then display greet message as "Hello, RAJ, nice to meet you!".

```
import java.util.Scanner;

public class GreetUser {
   public static void main(String[] args) {
      // Step 1: Create a Scanner object to accept input from the user
      Scanner scanner = new Scanner(System.in);

      // Step 2: Prompt the user to enter their name
      System.out.print("Please enter your name: ");
      String name = scanner.nextLine();

      // Step 3: Convert the name to uppercase
      String upperCaseName = name.toUpperCase();

      // Step 4: Display the greeting message with the name in uppercase
      System.out.println("Hello, " + upperCaseName + ", nice to meet you!");

      // Close the scanner
      scanner.close();
   }
}
```

Q2. Write a program which define class Product with data member as id, name and price. Store the information of 5 products and Display the name of product having minimum price (Use array of object).

```
import java.util.Scanner;

class Product {
   int id;
   String name;
   double price;

   // Constructor to initialize Product object
   public Product(int id, String name, double price) {
      this.id = id;
      this.name = name;
      this.price = price;
   }
}
```

```
public class ProductManagement {
   public static void main(String[] args) {
      // Create an array to store 5 Product objects
      Product[] products = new Product[5];

      // Scanner object to take user input
      Scanner scanner = new Scanner(System.in);

      // Step 1: Accept information for 5 products
      for (int i = 0; i < 5; i++) {
         System.out.println("Enter details for product " + (i + 1) + ":");
         System.out.print("Enter product id: ");
         int id = scanner.nextInt();
         scanner.nextLine(); // Consume newline character after integer input
         System.out.print("Enter product name: ");
         String name = scanner.nextLine();
         System.out.print("Enter product price: ");
         double price = scanner.nextDouble();

         // Create and store the product object in the array
         products[i] = new Product(id, name, price);
      }

      // Step 2: Find the product with the minimum price
      Product minPriceProduct = products[0]; // Assume the first product has the minimum price
initially
      for (int i = 1; i < products.length; i++) {
         if (products[i].price < minPriceProduct.price) {
            minPriceProduct = products[i]; // Update minPriceProduct if a lower price is found
         }
      }

      // Step 3: Display the name of the product with the minimum price
      System.out.println("\nThe product with the minimum price is: " + minPriceProduct.name);

      // Close the scanner
      scanner.close();
   }
}
```

Slip 5

Q1. Write a program to accept a number from the user, if number is zero then throw user defined exception —Number is 0, otherwise display factorial of a number.

```
class ZeroNumberException extends Exception {
    public ZeroNumberException() {
        super("Number is 0");
    }
}

public class FactorialCalculator1 {
    public static long calculateFactorial(int number) {
        if (number == 0) {
            return 1; // Factorial of 0 is 1
        }

        long factorial = 1;
        for (int i = 1; i <= number; i++) {
            factorial *= i;
        }
        return factorial;
    }

    public static void main(String[] args) {
        try {
            // Create a Scanner object to read user input
            java.util.Scanner scanner = new java.util.Scanner(System.in);

            // Prompt the user to enter a number
            System.out.print("Enter a number: ");
            int number = scanner.nextInt();

            if (number == 0) {
                throw new ZeroNumberException(); // Throw the custom exception for zero
            } else {
                // Calculate and display the factorial
                long factorial = calculateFactorial(number);
                System.out.println("Factorial of " + number + " is: " + factorial);
            }

            // Close the Scanner
            scanner.close();
        } catch (ZeroNumberException e) {
            System.err.println(e.getMessage()); // Handle the custom exception
        } catch (java.util.InputMismatchException e) {
            System.err.println("Invalid input. Please enter a valid number.");
        }
    }
}
```

Q2. Define a “Point” class having members – x,y (coordinates). Define default constructor and parameterized constructors. Define subclass “ColorPoint” with member as color. Write display method to display the details of Point.

```
// Superclass Point
class Point {
    int x, y;

    // Default constructor
    public Point() {
        this.x = 0;
        this.y = 0;
    }

    // Parameterized constructor
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // Method to display point details
    public void display() {
        System.out.println("Point coordinates: (" + x + ", " + y + ")");
    }
}

// Subclass ColorPoint
class ColorPoint extends Point {
    String color;
```

```
    // Default constructor
    public ColorPoint() {
        super(); // Call superclass default constructor
        this.color = "unknown";
    }

    // Parameterized constructor
    public ColorPoint(int x, int y, String color) {
        super(x, y); // Call superclass parameterized constructor
        this.color = color;
    }

    // Override the display method to include color information
    @Override
    public void display() {
        System.out.println("ColorPoint coordinates: (" + x + ", " + y + "), Color: " + color);
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating Point object using default constructor
        Point p1 = new Point();
        p1.display();

        // Creating Point object using parameterized constructor
        Point p2 = new Point(5, 10);
        p2.display();

        // Creating ColorPoint object using default constructor
        ColorPoint cp1 = new ColorPoint();
```

```
        cp1.display();


        // Creating ColorPoint object using parameterized constructor
        ColorPoint cp2 = new ColorPoint(7, 14, "Red");
        cp2.display();
    }
}
```

Slip 6

Q1. Accept 'n' integers from the user and store them in a collection. Display them in the sorted order. The collection should not accept duplicate elements. (Use a suitable collection). Search for a particular element using predefined search method in the Collection framework.

```
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;

public class UniqueSortedCollection1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of integers (n): ");
        int n = scanner.nextInt();

        Set<Integer> uniqueSortedSet = new TreeSet<>();

        for (int i = 0; i < n; i++) {
            System.out.print("Enter an integer: ");
            int num = scanner.nextInt();
            uniqueSortedSet.add(num);
        }

        System.out.println("Unique integers in sorted order:");
        for (int num : uniqueSortedSet) {
            System.out.print(num + " ");
        }

        System.out.println(); // Newline for readability

        System.out.print("Enter an integer to search: ");
        int searchNum = scanner.nextInt();
```

```
        if (uniqueSortedSet.contains(searchNum)) {
            System.out.println(searchNum + " is found in the collection.");
        } else {
            System.out.println(searchNum + " is not found in the collection.");
        }
    }
}
```

Q2. Write a program which define class Employee with data member as id, name and salary Store the information of 'n' employees and Display the name of employee having maximum salary (Use array of object.

```
import java.io.*;
class Employee
{
    String name;
    int salary;
    Employee(String nm,int sal)
    {
        name=nm;
        salary=sal;
    }
    void show()
    {
        System.out.println("Name : "+name);
        System.out.println("Salary : "+salary);
    }

    public static void main(String[] args) throws IOException
    {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        String n;
        int s,i;

        Employee e[]=new Employee[3];
        int sal[]=new int[3];

        for(i=0;i<3;i++)
        {
            System.out.print("Enter Employee name : ");
            n=br.readLine();
            System.out.print("Enter Employee salary : ");
            s=Integer.parseInt(br.readLine());
            e[i]=new Employee(n,s);
            sal[i]=e[i].salary;
```

```
    }

    System.out.print("Employee details are as follows : ");
    for(i=0;i<3;i++)
    {
      e[i].show();
    }
    int max=sal[0];
    int arr[]=new int[3];
    int j=0;
    for(i=0;i<3;i++)
    {
      if(max>sal[i])
      {
        max=max;
      }
      else
      {
        max=sal[i];
      }
    }
    System.out.println("Maximum salary is : "+max);
  }
}
```

Slip 7

Q1. Create a Hash table containing Employee name and Salary. Display the details of the hash table.

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class EmployeeSalaryTable {
  public static void main(String[] args) {
    // Create a HashMap to store employee names and salaries
    HashMap<String, Double> employeeSalaryMap = new HashMap<>();
    Scanner scanner = new Scanner(System.in);

    // Accept the number of employees
    System.out.print("Enter the number of employees: ");
    int n = scanner.nextInt();
    scanner.nextLine(); // Consume the newline left-over

    // Accept employee details
    for (int i = 0; i < n; i++) {
      System.out.print("Enter employee name: ");
```

```
            String name = scanner.nextLine();
            System.out.print("Enter employee salary: ");
            double salary = scanner.nextDouble();
            scanner.nextLine(); // Consume the newline left-over

            // Store the employee name and salary in the hash table
            employeeSalaryMap.put(name, salary);
        }

        // Display the details of the hash table
        System.out.println("\nEmployee Salary Details:");
        for (Map.Entry<String, Double> entry : employeeSalaryMap.entrySet()) {
            System.out.println("Employee Name: " + entry.getKey() + ", Salary: " + entry.getValue());
        }

        // Close the scanner
        scanner.close();
    }
}
```

Q2. Define a class student having rollno, name and percentage. Define Default and parameterized constructor. Accept the 5 student details and display it. (Use this keyword).

```
import java.util.Scanner;
class Student
{
int rno;
String name;
double per;

public Student()
{
this.rno=1;
this.name="Jidnyasa";
this.per=80.50;
}

public Student(int rno, String name, double per)
{
this.rno=rno;
this.name=name;
this.per=per;
}

void display()
{
System.out.println(rno+" "+name+" "+per)
```

```
}

public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter how many students : ");
int n=sc.nextInt();
for(int i=1;i<=n;i++)
{
System.out.println("Enter student rollno : ");
int srno=sc.nextInt();
System.out.println("Enter student name : ");
String sname=sc.next();
System.out.println("Enter student percentage : ");
double sper=sc.nextDouble();
Student s=new Student(srno,sname,sper);
s.display();
}
}
}
```

Slip 8

Q1. Write a program to reverse a number. Accept number using command line argument. [10 Marks]

```
import java.util.Scanner;
class s8q1
{
public static void main(String[] args)
{
int n=Integer.parseInt(args[0]);
while(n>0)
{
int k=n%10;
System.out.print(k);
n=n/10;
}
}
}
```

Q2. Define a class MyDate (day, month, year) with methods to accept and display MyDate object. Accept date as dd, mm, yyyy. Throw user defined exception “InvalidDateException” if the date is invalid. Examples of invalid dates : 12 15 2015, 31 6 1990, 29 2 2001

```
import java.util.Scanner;
```

```
// User-defined exception class for invalid dates
class InvalidDateException extends Exception {
    public InvalidDateException(String message) {
        super(message);
    }
}

// MyDate class to represent a date
class MyDate {
    private int day;
    private int month;
    private int year;

    // Method to accept date input
    public void acceptDate() throws InvalidDateException {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter date as dd mm yyyy: ");
        day = scanner.nextInt();
        month = scanner.nextInt();
        year = scanner.nextInt();

        // Validate the date
        if (!isValidDate(day, month, year)) {
            throw new InvalidDateException("Invalid Date: " + day + "/" + month + "/" + year);
        }
    }

    // Method to check if the date is valid
    private boolean isValidDate(int day, int month, int year) {
        if (month < 1 || month > 12) return false; // Month must be between 1 and 12
```

```
        // Days in month
        int[] daysInMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        // Check for leap year for February
        if (month == 2 && isLeapYear(year)) {
            daysInMonth[2] = 29;
        }

        return day > 0 && day <= daysInMonth[month]; // Day must be valid for the month
    }

    // Method to check if a year is a leap year
    private boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }

    // Method to display the date
    public void displayDate() {
        System.out.printf("Date: %02d/%02d/%04d%n", day, month, year);
    }
}

public class Main {
    public static void main(String[] args) {
        MyDate date = new MyDate();

        try {
            date.acceptDate(); // Accept the date input
            date.displayDate(); // Display the valid date
        } catch (InvalidDateException e) {
```

```
            System.out.println(e.getMessage()); // Print the exception message
        }
    }
}
```

Slip 9

Q1. Write a program to accept a number from user. Check whether number is perfect or not. Use BufferedReader class for accepting input from user.

```
import java.io.*;
class s9q1
{
public static void main(String[] args)
{
int sum=0;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter a number : ");
int n=Integer.parseInt(br.readLine());
for(int i=0;i<n;i++)
{
if(n%i==0)
sum=sum+i;
}
if(sum==n)
System.out.println("Number is perfect");
else
System.out.println("Number is not perfect");
}
}
```

Q2. Define a “Point” class having members – x,y(coordinates). Define default constructor and parameterized constructors. Define subclass “Point3D” with member as z (coordinate). Write display method to show the details of Point

```
// Superclass Point
class Point {
    protected int x;
    protected int y;

    // Default constructor
    public Point() {
```

```
        this.x = 0;
        this.y = 0;
    }

    // Parameterized constructor
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    // Method to display point details
    public void display() {
        System.out.println("Point coordinates: (" + x + ", " + y + ")");
    }
}

// Subclass Point3D
class Point3D extends Point {
    private int z;

    // Default constructor
    public Point3D() {
        super(); // Call superclass default constructor
        this.z = 0;
    }

    // Parameterized constructor
    public Point3D(int x, int y, int z) {
        super(x, y); // Call superclass parameterized constructor
        this.z = z;
    }

    // Override the display method to include z coordinate
    @Override
    public void display() {
        System.out.println("Point3D coordinates: (" + x + ", " + y + ", " + z + ")");
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating a Point object using default constructor
        Point p1 = new Point();
        p1.display();

        // Creating a Point object using parameterized constructor
        Point p2 = new Point(5, 10);
        p2.display();
```

```
        // Creating a Point3D object using default constructor
        Point3D p3d1 = new Point3D();
        p3d1.display();

        // Creating a Point3D object using parameterized constructor
        Point3D p3d2 = new Point3D(7, 14, 21);
        p3d2.display();
    }
}
```

Slip 10

Q1. Write a program to accept a number from user. Check whether number is prime or not.

```
public class s10q1
{
   public static void main(String[] args)
   {
      int num=11;
      boolean isPrime=true;
      for(int i=2;i<=Math.sqrt(num);i++)
      {
         if(num%i==0)
         {
            isPrime=false;
            break;
         }
      }
   System.out.println("isPrime");
   }
}
```

Q3. Define a class SavingAccount (acno, name, balance). Define appropriate operations as, withdraw(), deposit(), and viewbalance(). The minimum balance must be 500. Create an object and perform operation. Raise user defined —InsufficientFundException when balance is not sufficient for withdraw operation.

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

class InsufficientFund extends Exception
{
}
```

```
class SavingsAccount
{
   Scanner sc=new Scanner(System.in);
   int accno;
   String name;
   double balance;

   public SavingsAccount(int a, String n, double b)
   {
      accno=a;
      name=n;
      balance=b;
   }

   public void withdraw()
   {
      try
      {
         System.out.print("Enter amount to be withdrawn : ");
         double wd=sc.nextDouble();
         balance-=wd;
         if(balance<500)
         {
            throw new InsufficientFund();
         }
      }
      catch(InsufficientFund e)
      {
         System.out.println("Insufficient Fund, minimum balance required is 500");
      }
   }

   public void deposit()
   {
      System.out.print("Enter amount to be deposited : ");
      double de=sc.nextDouble();
      balance+=de;
   }

   public void view()
   {
      System.out.print("Current balance available : "+balance);
   }
}

class s10q3
{
   public static void main(String[] args) throws IOException
```

```
    {
        SavingsAccount s=new SavingsAccount(101,"XYZ",1000);
        s.deposit();
        s.withdraw();
        s.view();
    }
}
```

Slip 11

Q1. Write a program create class as MyDate with dd,mm,yy as data members. Write parameterized constructor. Display the date in dd-mm-yy format. (Use this keyword)

```
import java.util.Scanner;

// MyDate class definition
class MyDate {
    private int dd; // Day
    private int mm; // Month
    private int yy; // Year

    // Parameterized constructor
    public MyDate(int dd, int mm, int yy) {
        this.dd = dd; // Using this keyword to refer to the instance variable
        this.mm = mm; // Using this keyword to refer to the instance variable
        this.yy = yy; // Using this keyword to refer to the instance variable
    }

    // Method to display the date in dd-mm-yy format
    public void displayDate() {
        System.out.printf("Date: %02d-%02d-%02d%n", dd, mm, yy);
    }
}

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

        // Prompt the user for date input
        System.out.print("Enter day (dd): ");
        int day = scanner.nextInt();

        System.out.print("Enter month (mm): ");
        int month = scanner.nextInt();

        System.out.print("Enter year (yy): ");
        int year = scanner.nextInt();
```

```
        // Create an instance of MyDate using the user input
        MyDate date = new MyDate(day, month, year);

        // Display the date
        date.displayDate();

        // Close the scanner
        scanner.close();
    }
}
```

Q2. Create an abstract class Shape with methods area & volume. Derive a class Sphere (radius). Calculate and display area and volume.

```
import java.util.*;

abstract class Shape
{
    public abstract void area();
    public abstract void volume();
}

class Sphere extends Shape
{
    double r;
    Sphere(double radius)
    {
        r = radius;
    }

    public void area()
    {
        double a = 4 * 3.14 * r * r;
        System.out.println("Area of Sphere: " + a);
    }

    public void volume()
    {
        double v = (4.0 / 3.0) * 3.14 * r * r * r;
        System.out.println("Volume of Sphere: " + v);
    }
}
```

```
class s11q2
{
   public static void main(String[] args)
   {
      Sphere s = new Sphere(5);
      s.area();
      s.volume();
   }
}
```

Slip 12

Q1. Create a package named “Series” having a class to print series of Square of numbers. Write a program to generate “n” terms series.

Create package –

```
package series;

public class SquareSeries {
   public void generateSeries(int n) {
      for (int i = 1; i <= n; i++) {
         int square = i * i;
         System.out.print(square + " ");
      }
      System.out.println();
   }
}
```

Class –

```
import series.SquareSeries;
import java.util.Scanner;

public class SquareSeriesExample1 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of terms (n) for the square series: ");
      int n = scanner.nextInt();

      SquareSeries series = new SquareSeries();
      System.out.print("Square Series: ");
      series.generateSeries(n);
   }
}
```

Q2. Create an abstract class Shape with methods area & volume. Derive a class Cylinder (radius, height). Calculate area and volume.

```
import java.util.*;
abstract class Shape
{
public abstract void area();
public abstract void volume();
}

class Cylinder extends Shape
{
double radius,height;
Cylinder(double radius,double height)
{
this.radius=radius;
this.height=height;
}

public void area()
{
double a=2*3.14*radius*height+2*3.14*radius*radius;
System.out.println("Area of Cylinder : "+a);
}

public void volume()
{
double v=3.14*radius*radius*height;
System.out.println("Volume of Cylinder : "+v);
}
}

class s12q2
{
public static void main(String[] args)
{
Cylinder c=new Cylinder(5.5,1.2);
c.area();
c.volume();
}
}
```

Slip 13

Q1. Construct a Linked List having names of Fruits: Apple, Banana, Guava and Orange. Display the contents of the List using an Iterator.

```
import java.util.*;
class collections
{
   public static void main(String arg[])
   {
      List t1=new LinkedList();
      t1.add("Apple");
      t1.add("Banana");
      t1.add("Grapes");
      t1.add("Orange");
      ListIterator it=t1.listIterator();
      System.out.println("Fruits Names : ");
      while(it.hasNext())
      {
         System.out.println(it.next());
      }
   }
}
```

Q2. Define an interface “Operation” which has methods area(),volume(). Define a constant PI having value 3.142. Create a class circle (member – radius) which implements this interface. Calculate and display the area and volume.

```
import java.util.Scanner;
interface operation
{
   abstract void area();
   abstract void volume();
   final double pi=3.14;
}

class circle implements operation
{
   double r;
   double h;
   circle(double radius, double height)
   {
      r=radius;
```

```
        h=height;
    }

    public void area()
    {
        double a=pi*r*r;
        System.out.println("Area of circle : "+a);
    }

    public void volume()
    {
        double v=pi*r*r*h;
        System.out.println("Volume of circle : "+v);
    }
}

class s13q2
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter radius and height : ");
        double a=sc.nextDouble();
        double v=sc.nextDouble();
        circle c=new circle(a,v);
        c.area();
        c.volume();
    }
}
```

Q3. Write a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user- defined exception —Age Not Within The Range.

```
import java.util.Scanner;
interface operation
{
    abstract void area();
    abstract void volume();
    final double pi=3.14;
}

class circle implements operation
{
    double r;
```

```
    double h;
    circle(double radius, double height)
    {
        r=radius;
        h=height;
    }

    public void area()
    {
        double a=pi*r*r;
        System.out.println("Area of circle : "+a);
    }

    public void volume()
    {
        double v=pi*r*r*h;
        System.out.println("Volume of circle : "+v);
    }
}

class s13q2
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter radius and height : ");
        double a=sc.nextDouble();
        double v=sc.nextDouble();
        circle c=new circle(a,v);
        c.area();
        c.volume();
    }
}
```

Slip 14

Q1. Write a program to create JDBC connection. On successful connection with database display appropriate message to user.

Q2. Define an interface “Operation” which has methods area(),volume(). Define a constant PI having a value 3.142. Create a class cylinder (members – radius, height) which implements this interface. Calculate and display the area and volume.

```
import java.util.Scanner;
interface operation
{
   abstract void area();
   abstract void volume();
   final double pi=3.14;
}

class cylinder implements operation
{
   double r;
   double h;
   cylinder(double radius, double height)
   {
      r=radius;
      h=height;
   }

   public void area()
   {
      double a=2*pi*r*h+2*pi*r*r;
      System.out.println("Area of cylinder : "+a);
   }

   public void volume()
   {
      double v=pi*r*r*h;
      System.out.println("Volume of cylinder : "+v);
   }
}

class s14q2
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter radius and height : ");
      double a=sc.nextDouble();
      double v=sc.nextDouble();
      cylinder c=new cylinder(a,v);
      c.area();
      c.volume();
   }
}
```

Slip 15

Q1. Construct a Linked List having names of Fruits: Apple, Banana, Guava and Orange. Display the contents of the List in reverse order using an appropriate interface.

```
import java.util.*;
class slip15
{
   public static void main(String arg[])
   {
      LinkedList t1=new LinkedList();
      t1.add("Apple");
      t1.add("Banana");
      t1.add("Grapes");
      t1.add("Orange");
      ListIterator it=t1.listIterator(t1.size());
      System.out.println("Fruits Names : ");
      while(it.hasPrevious())
      {
         System.out.println(it.previous());
      }
   }
}
```

Q2. Write a program to create a super class Employee (members – name, salary). Derive a sub- class as Developer (member – projectname). Create object of Developer and display the details of it.

```
import java.util.Scanner;
class employee
{
   String name;
   double salary;

   employee(String n,double s)
   {
      name=n;
      salary=s;
   }

   public void displayEmp()
   {
      System.out.println("name : "+name);
      System.out.println("salary : "+salary);
   }
}
```

```
class developer extends employee
{
   String pname;
   public developer(String n1,double s1,String p1)
   {
      super(n1,s1);
      pname=p1;
   }

   void displayProj()
   {
      System.out.println("project name : "+pname);
   }
}

public class s15q2
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter name : ");
      String name1=sc.next();
      System.out.print("Enter salary : ");
      double salary1=sc.nextDouble();
      System.out.print("Enter project name : ");
      String pname1=sc.next();
      developer d=new developer(name1,salary1,pname1);
      d.displayEmp();
      d.displayProj();
   }
}
```

Slip 16

Q1. Define a class MyNumber having one private integer data member. Write a parameterized constructor to initialize to a value. Write isEven() to check given number is even or not. Use command line argument to pass a value to the object.

```
class MyNumber
{
int a;
MyNumber(int x)
{
a=x;
}
```

```
void isEven()
{
if(a%2==0)
System.out.println("Number is even");
}

public static void main(String[] args)
{
int n=Integer.parseInt(args[0]);
MyNumber m1=new MyNumber(n);
m1.isEven();
}
}
```

Command on terminal –

```
javac MyNumber.java
java MyNumber 3
```

Q2. Write a program to create a super class Employee (members – name, salary). Derive a sub- class Programmer (member – proglanguage). Create object of Programmer and display the details of it.

```
import java.util.Scanner;

// Superclass Employee
class Employee {
   protected String name;  // Employee name
   protected double salary; // Employee salary

   // Constructor for Employee
   public Employee(String name, double salary) {
      this.name = name;
      this.salary = salary;
   }

   // Method to display employee details
   public void displayDetails() {
      System.out.println("Name: " + name);
      System.out.println("Salary: $" + salary);
   }
}

// Subclass Programmer, inheriting from Employee
class Programmer extends Employee {
   private String proglanguage; // Programming language
```

```
    // Constructor for Programmer
    public Programmer(String name, double salary, String proglanguage) {
        super(name, salary); // Call to superclass constructor
        this.proglanguage = proglanguage;
    }

    // Overriding displayDetails to include programming language
    @Override
    public void displayDetails() {
        super.displayDetails(); // Call superclass displayDetails
        System.out.println("Programming Language: " + proglanguage);
    }
}

// Main class to test the functionality
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for input

        // Taking user input for the Employee details
        System.out.print("Enter the name of the programmer: ");
        String name = scanner.nextLine(); // Read the name

        System.out.print("Enter the salary of the programmer: ");
        double salary = scanner.nextDouble(); // Read the salary

        // Clear the buffer
        scanner.nextLine();

        System.out.print("Enter the programming language: ");
        String proglanguage = scanner.nextLine(); // Read the programming language

        // Create a Programmer object with user input
        Programmer programmer = new Programmer(name, salary, proglanguage);

        // Display the programmer's details
        programmer.displayDetails();

        // Close the scanner
        scanner.close();
    }
}
```

Slip 17

Q1. Define a class MyNumber having one private integer data member. Write a parameterized constructor to initialize to a value. Write isOdd() to check given number is odd or not. Use command line argument to pass a value to the object.

```
class MyNumber
{
int a;
MyNumber(int x)
{
a=x;
}

void isOdd()
{
if(a%2!=0)
System.out.println("Number is odd");
}

public static void main(String[] args)
{
int n=Integer.parseInt(args[0]);
MyNumber m1=new MyNumber(n);
m1.isOdd();
}
}
```

Command on terminal –

```
javac MyNumber.java
java MyNumber 4
```

Q2. Define a class Student with attributes rollno and name. Define default and parameterized constructor. Keep the count of Objects created. Create objects using parameterized constructor and Display the object count after each object is created.

```
import java.util.*;
class Student
{
   int rno;
   String name;
   static int count;
```

```
    public Student()
    {
        count++;
    }

    public Student(int rno, String name)
    {
        this.rno=rno;
        this.name=name;
        count++;
    }

    public static int getCount()
    {
        return count;
    }

    public void display()
    {
        System.out.print("Roll no : "+rno);
        System.out.print("Name : "+name);
    }
}
```

Slip 18

Q1. Write a program to print the factors of a number. Accept a number using command line argument.

```
public class FactorPrinter1 {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Usage: java FactorPrinter <number>");
        } else {
            try {
                int number = Integer.parseInt(args[0]);
                if (number < 1) {
                    System.out.println("Please enter a positive integer.");
                } else {
                    System.out.print("Factors of " + number + ": ");
                    for (int i = 1; i <= number; i++) {
                        if (number % i == 0) {
                            System.out.print(i + " ");
                        }
                    }
                }
            } catch (NumberFormatException e) {
```

```
            System.out.println("Please enter a valid integer as a command-line argument.");
        }
    }
  }
}
```

Q2. Write a program to read the contents of “abc.txt” file. Display the contents of file in uppercase as output.

```
import java.io.FileReader;
import java.io.IOException;

public class ReadFileAndConvertToUpperCase2 {
   public static void main(String[] args) {
      String fileName = "abc.txt";  // Change this to the path of your file if it's not in the same directory as the Java file

      try {
         BufferedReader reader = new BufferedReader(new FileReader(fileName));
         String line;

         while ((line = reader.readLine()) != null) {
            // Convert each line to uppercase and print it
            System.out.println(line.toUpperCase());
         }

         reader.close();
      } catch (IOException e) {
         System.err.println("An error occurred while reading the file: " + e.getMessage());
      }
   }
}
```

Slip 19

Q1. Write a program to accept the 'n' different numbers from user and store it in array. Display maximum number from an array.

```
import java.util.Scanner;

public class MaxNumberFinder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Create a Scanner object for user input

        // Accept the number of elements
        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();

        // Create an array to store the numbers
        int[] numbers = new int[n];

        // Accept the numbers from the user
        System.out.println("Enter " + n + " different numbers:");
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt(); // Store each number in the array
        }

        // Find the maximum number in the array
        int maxNumber = numbers[0]; // Assume the first number is the maximum initially

        for (int i = 1; i < n; i++) {
            if (numbers[i] > maxNumber) {
                maxNumber = numbers[i]; // Update maxNumber if current number is greater
            }
        }

        // Display the maximum number
        System.out.println("The maximum number is: " + maxNumber);

        // Close the scanner
        scanner.close();
    }
}
```

Q2. Create an abstract class “order” having members id, description. Create a subclass “Purchase Order” having member as customer name. Define methods accept and display. Create 3 objects each of Purchase Order. Accept and display the details.

```
import java.util.Scanner;

// Abstract class Order
abstract class Order {
   protected int id;
   protected String description;

   public abstract void accept();
   public abstract void display();
}

// Subclass PurchaseOrder
class PurchaseOrder extends Order {
   private String customerName;

   @Override
   public void accept() {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter Order ID: ");
      id = scanner.nextInt();
      scanner.nextLine(); // Consume newline
      System.out.print("Enter Order Description: ");
      description = scanner.nextLine();
      System.out.print("Enter Customer Name: ");
      customerName = scanner.nextLine();
   }

   @Override
   public void display() {
      System.out.println("Order ID: " + id + ", Description: " + description + ", Customer: " +
customerName);
   }
}

// Main class
public class Main {
   public static void main(String[] args) {
      PurchaseOrder[] orders = new PurchaseOrder[3];
      for (int i = 0; i < 3; i++) {
         orders[i] = new PurchaseOrder();
         System.out.println("\n--- Enter details for Purchase Order " + (i + 1) + " ---");
         orders[i].accept();
      }
```

```
      System.out.println("\n--- Purchase Orders ---");
      for (PurchaseOrder order : orders) {
         order.display();
      }
   }
}
```

Slip 20

Q1. Write a program to accept 3 numbers using command line argument. Sort and display the numbers.

```
// run program
// javac CommandLineNumberSort1.java
// java CommandLineNumberSort1 3 1 2


public class CommandLineNumberSort1 {
   public static void main(String[] args) {
      if (args.length != 3) {
         System.out.println("Usage: java CommandLineNumberSort <number1> <number2>
<number3>");
      } else {
         try {
            int num1 = Integer.parseInt(args[0]);
            int num2 = Integer.parseInt(args[1]);
            int num3 = Integer.parseInt(args[2]);

            int temp;

            // Sort the numbers using a simple bubble sort algorithm
            if (num1 > num2) {
               temp = num1;
               num1 = num2;
               num2 = temp;
            }
            if (num2 > num3) {
               temp = num2;
               num2 = num3;
               num3 = temp;
            }
            if (num1 > num2) {
               temp = num1;
               num1 = num2;
               num2 = temp;
            }

            System.out.println("Sorted numbers: " + num1 + " " + num2 + " " + num3);
         } catch (NumberFormatException e) {
```

```
            System.out.println("Please enter valid integer numbers as command-line arguments.");
        }
      }
    }
}
```

Q2. Create an employee class (id,name,deptname,salary). Define a default and parameterized constructor. Use ‘this’ keyword to initialize instance variables. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created. Also display the contents of each object.

```
public class Employee2 {
    private int id;
    private String name;
    private String deptName;
    private double salary;
    private static int objectCount = 0;

    public Employee2() {
        // Default constructor
        this.id = 0;
        this.name = "Unknown";
        this.deptName = "Unknown";
        this.salary = 0.0;
        objectCount++;
    }

    public Employee2(int id, String name, String deptName, double salary) {
        // Parameterized constructor using 'this' keyword
        this.id = id;
        this.name = name;
        this.deptName = deptName;
        this.salary = salary;
        objectCount++;
    }

    public void display() {
        System.out.println("Employee2 ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Department: " + deptName);
        System.out.println("Salary: " + salary);
    }

    public static int getObjectCount() {
        return objectCount;
    }

    public static void main(String[] args) {
        Employee2 emp1 = new Employee2(101, "saurabh", "HR", 45000.0);
```

```
        System.out.println("Object Count: " + getObjectCount());
        emp1.display();
        System.out.println();

        Employee2 emp2 = new Employee2(102, "yogesh ", "IT", 55000.0);
        System.out.println("Object Count: " + getObjectCount());
        emp2.display();
    }
}
```