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 MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

System.out.println("Multiplication Table for " + number + ":");

for (int i = 1; i <= 10; i++) {

int result = number \* i;

System.out.println(number + " x " + i + " = " + result);

}

scanner.close();

}

}

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.\*;

public class ColorLinkedList {

public static void main(String[] args) {

// Create a linked list of colors

LinkedList<String> colorList = new LinkedList<>();

colorList.add("red");

colorList.add("blue");

colorList.add("yellow");

colorList.add("orange");

// i. Display the contents of the List using an Iterator

System.out.println("Contents of the list using an Iterator:");

Iterator<String> iterator = colorList.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

// ii. Display the contents of the List in reverse order using a ListIterator

System.out.println("\nContents of the list in reverse order using a ListIterator:");

ListIterator<String> listIterator = colorList.listIterator(colorList.size());

while (listIterator.hasPrevious()) {

System.out.println(listIterator.previous());

}

// iii. Create another list containing pink and green. Insert elements between blue and yellow

LinkedList<String> newColors = new LinkedList<>();

newColors.add("pink");

newColors.add("green");

// Insert elements between "blue" and "yellow"

ListIterator<String> insertionPoint = colorList.listIterator();

while (insertionPoint.hasNext()) {

if (insertionPoint.next().equals("blue")) {

insertionPoint.add(newColors.poll()); // Insert pink

insertionPoint.add(newColors.poll()); // Insert green

break;

}

}

// Display the updated list

System.out.println("\nUpdated list with pink and green inserted between blue and yellow:");

for (String color : colorList) {

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.ArrayList;

import java.util.Scanner;

public class Arraylist {

public static void main(String[] args) {

// Create an ArrayList to store integers

ArrayList<Integer> arrayList = new ArrayList<>();

// Create a Scanner object to read user input

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of integers (n): ");

int n = scanner.nextInt();

// Read 'n' integers from the user and add them to the ArrayList

for (int i = 0; i < n; i++) {

System.out.print("Enter integer " + (i + 1) + ": ");

int num = scanner.nextInt();

arrayList.add(num);

}

// Close the scanner

scanner.close();

// Display the elements of the ArrayList

System.out.println("Elements in the ArrayList:");

for (int i = 0; i < arrayList.size(); i++) {

System.out.println("Element " + (i + 1) + ": " + arrayList.get(i));

}

}

}

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.

:-

OR

Q2. Write a program to accept Doctor Name from the user and check whether it is valid or not.(It

should not contain digits and special symbol) If it is not valid then throw user defined Exception - Name is Invalid -- otherwise display it

:-

import java.util.Scanner;

class InvalidNameException extends Exception {

public InvalidNameException(String message) {

super(message);

}

}

public class DoctorNameValidator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter the doctor's name: ");

String doctorName = scanner.nextLine();

validateDoctorName(doctorName);

System.out.println("Doctor's name is valid: " + doctorName);

} catch (InvalidNameException e) {

System.out.println("Error: " + e.getMessage());

} finally {

scanner.close();

}

}

public static void validateDoctorName(String name) throws InvalidNameException {

// Check if the name contains digits or special symbols

if (name.matches(".[0-9!@#$%^&()\_+\\-=\\[\\]{};':\"\\\\|,.<>/?].\*")) {

throw new InvalidNameException("Name is Invalid. It should not contain digits or special symbols.");

}

}

}

Slip 3

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

printthe sum of elements of the array.

:-

import java.util.Scanner;

public class SumofArraylist {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter the value of 'n'

System.out.print("Enter the number of elements (n): ");

int n = scanner.nextInt();

// Create an array of size 'n'

int[] arr = new int[n];

// Prompt the user to enter 'n' different numbers and store them in the array

System.out.println("Enter " + n + " different numbers:");

for (int i = 0; i < n; i++) {

System.out.print("Enter element " + (i + 1) + ": ");

arr[i] = scanner.nextInt();

}

// Calculate the sum of array elements

int sum = 0;

for (int i = 0; i < n; i++) {

sum += arr[i];

}

// Print the elements of the array

System.out.println("Elements in the array:");

for (int i = 0; i < n; i++) {

System.out.print(arr[i] + " ");

}

// Print the sum of array elements

System.out.println("\nSum of elements in the array: " + sum);

scanner.close();

}

}

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.

:-

OR

Q2.Write a program to copy the contents from one file into another file in upper case

:-

import java.io.\*;

public class CopytoUpper {

public static void main(String[] args) {

try {

// Specify the input and output file names

String inputFile = "input.txt"; // Replace with your input file

String outputFile = "output.txt"; // Replace with your output file

// Create File and BufferedReader for input

FileReader fileReader = new FileReader(inputFile);

BufferedReader bufferedReader = new BufferedReader(fileReader);

// Create FileWriter for output

FileWriter fileWriter = new FileWriter(outputFile);

String line;

while ((line = bufferedReader.readLine()) != null) {

// Convert the line to uppercase and write it to the output file

String upperCaseLine = line.toUpperCase();

fileWriter.write(upperCaseLine + "\n");

}

// Close the input and output streams

fileReader.close();

bufferedReader.close();

fileWriter.close();

System.out.println("Contents copied to " + outputFile + " in uppercase.");

} catch (IOException e) {

System.err.println("An error occurred: " + e.getMessage());

}

}

}

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) {

// Create a Scanner object to read user input

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter their name

System.out.print("Please enter your name: ");

String userName = scanner.nextLine();

// Convert the user's name to uppercase

String upperCaseName = userName.toUpperCase();

// Display the greeting message

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 arrayof

object

:-

class Product {

int id;

String name;

double price;

Product(int id, String name, double price) {

this.id = id;

this.name = name;

this.price = price;

}

}

class ProductDemo {

public static void main(String[] args) {

// Create an array to store 5 products

Product[] products = new Product[5];

// Initialize the products

products[0] = new Product(1, "Product A", 10.50);

products[1] = new Product(2, "Product B", 8.75);

products[2] = new Product(3, "Product C", 15.25);

products[3] = new Product(4, "Product D", 6.99);

products[4] = new Product(5, "Product E", 12.50);

// Find the product with the minimum price

Product minPriceProduct = products[0];

for (int i = 1; i < products.length; i++) {

if (products[i].price < minPriceProduct.price) {

minPriceProduct = products[i];

}

}

// Display the name of the product with the minimum price

System.out.println("Product with the minimum price: " + minPriceProduct.name);

}

}

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.

:-

import java.util.Scanner;

// Custom exception class for a number being zero

class NumberIsZeroException extends Exception {

public NumberIsZeroException() {

super("Number is 0");

}

}

public class FactorCalculator {

// Function to calculate the factorial of a number

public static long calculateFactorial(int num) {

if (num == 0 || num == 1) {

return 1;

} else {

return num \* calculateFactorial(num - 1);

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter a number: ");

int num = scanner.nextInt();

if (num == 0) {

throw new NumberIsZeroException();

} else {

long factorial = calculateFactorial(num);

System.out.println("Factorial of " + num + " is: " + factorial);

}

} catch (NumberIsZeroException e) {

System.err.println("Error: " + e.getMessage());

} catch (java.util.InputMismatchException e) {

System.err.println("Invalid input. Please enter a valid number.");

} finally {

scanner.close();

}

}

}

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

:-

class Point {

int x;

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;

}

public void display() {

System.out.println("Point: (" + x + ", " + y + ")");

}

}

class ColorPoint extends Point {

private String color;

// Parameterized constructor

public ColorPoint(int x, int y, String color) {

super(x, y);

this.color = color;

}

@Override

public void display() {

System.out.println("ColorPoint: (" + super.x + ", " + super.y + "), Color: " + color);

}

}

public class Main {

public static void main(String[] args) {

Point point1 = new Point();

Point point2 = new Point(3, 4);

ColorPoint colorPoint = new ColorPoint(1, 2, "Red");

point1.display();

point2.display();

colorPoint.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.TreeSet;

public class SortedUniqueCollection {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

TreeSet<Integer> numbers = new TreeSet<>();

// Accept n integers from the user

System.out.print("Enter the number of integers (n): ");

int n = scanner.nextInt();

System.out.println("Enter " + n + " integers:");

for (int i = 0; i < n; i++) {

int num = scanner.nextInt();

numbers.add(num);

}

// Display them in sorted order (automatically sorted due to TreeSet)

System.out.println("Sorted and unique integers: " + numbers);

// Search for a particular element

System.out.print("Enter the element to search for: ");

int searchElement = scanner.nextInt();

if (numbers.contains(searchElement)) {

System.out.println(searchElement + " is found in the collection.");

} else {

System.out.println(searchElement + " is not found in the collection.");

}

scanner.close();

}

}

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.util.Scanner;

class Employee {

int id;

String name;

double salary;

Employee(int id, String name, double salary) {

this.id = id;

this.name = name;

this.salary = salary;

}

}

class EmployeeMain {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the number of employees: ");

int n = sc.nextInt();

Employee[] employees = new Employee[n];

for (int i = 0; i < n; i++) {

System.out.println("Enter Employee " + (i + 1) + " details:");

System.out.print("ID: ");

int id = sc.nextInt();

sc.nextLine(); // Consume the newline character

System.out.print("Name: ");

String name = sc.nextLine();

System.out.print("Salary: ");

double salary = sc.nextDouble();

employees[i] = new Employee(id, name, salary);

}

// Find the employee with the maximum salary

Employee maxSalaryEmployee = employees[0];

for (int i = 1; i < n; i++) {

if (employees[i].salary > maxSalaryEmployee.salary) {

maxSalaryEmployee = employees[i];

}

}

System.out.println("Employee with the maximum salary:");

System.out.println("Name: " + maxSalaryEmployee.name);

System.out.println("Salary: " + maxSalaryEmployee.salary);

sc.close();

}

}

Slip 7

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

table.

:-

import java.util.Hashtable;

import java.util.Scanner;

public class EmployeeSalaryHashtable {

public static void main(String[] args) {

Hashtable<String, Double> employeeSalaries = new Hashtable<>();

// Add employee names and salaries to the Hashtable

employeeSalaries.put("John", 50000.0);

employeeSalaries.put("Alice", 60000.0);

employeeSalaries.put("Bob", 55000.0);

employeeSalaries.put("Eve", 62000.0);

// Display the details of the Hashtable

System.out.println("Employee Details:");

for (String employeeName : employeeSalaries.keySet()) {

double salary = employeeSalaries.get(employeeName);

System.out.println("Name: " + employeeName + ", Salary: $" + salary);

}

}

}

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 rollNo;

String name;

double percentage;

// Default constructor

public Student() {

this.rollNo = 0;

this.name = "";

this.percentage = 0.0;

}

// Parameterized constructor

public Student(int rollNo, String name, double percentage) {

this.rollNo = rollNo;

this.name = name;

this.percentage = percentage;

}

public void displayDetails() {

System.out.println("Roll Number: " + this.rollNo);

System.out.println("Name: " + this.name);

System.out.println("Percentage: " + this.percentage + "%");

System.out.println();

}

}

class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Student[] students = new Student[5];

for (int i = 0; i < 5; i++) {

System.out.println("Enter details for Student " + (i + 1) + ":");

System.out.print("Roll Number: ");

int rollNo = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Percentage: ");

double percentage = scanner.nextDouble();

students[i] = new Student(rollNo, name, percentage);

}

System.out.println("Student Details:");

for (Student student : students) {

student.displayDetails();

}

}

}

Slip 8

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

:-

public class ReverseNumber {

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Please provide a single number as a command-line argument.");

return;

}

String input = args[0];

try {

int number = Integer.parseInt(input);

int reversedNumber = reverseNumber(number);

System.out.println("Original Number: " + number);

System.out.println("Reversed Number: " + reversedNumber);

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please provide a valid number.");

}

}

public static int reverseNumber(int number) {

int reversedNumber = 0;

while (number != 0) {

int digit = number % 10;

reversedNumber = reversedNumber \* 10 + digit;

number /= 10;

}

return reversedNumber;

}

}

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” ifthe date is

invalid. Examples of invalid dates : 12 15 2015, 31 6 1990, 29 2 2001

:-

class InvalidDateException extends Exception {

public InvalidDateException(String message) {

super(message);

}

}

class MyDate {

private int day;

private int month;

private int year;

public MyDate(int day, int month, int year) throws InvalidDateException {

if (!isValidDate(day, month, year)) {

throw new InvalidDateException("Invalid date entered.");

}

this.day = day;

this.month = month;

this.year = year;

}

private boolean isValidDate(int day, int month, int year) {

if (month < 1 || month > 12 || day < 1) {

return false;

}

int maxDays = 31;

switch (month) {

case 4:

case 6:

case 9:

case 11:

maxDays = 30;

break;

case 2:

maxDays = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)) ? 29 : 28;

break;

}

return day <= maxDays;

}

public void displayDate() {

System.out.println("Date: " + day + "/" + month + "/" + year);

}

}

public class Main {

public static void main(String[] args) {

try {

MyDate date1 = new MyDate(12, 15, 2015); // Invalid date

date1.displayDate();

} catch (InvalidDateException e) {

System.out.println("Error: " + e.getMessage());

}

try {

MyDate date2 = new MyDate(31, 6, 1990); // Invalid date

date2.displayDate();

} catch (InvalidDateException e) {

System.out.println("Error: " + e.getMessage());

}

try {

MyDate date3 = new MyDate(29, 2, 2001); // Invalid date

date3.displayDate();

} catch (InvalidDateException e) {

System.out.println("Error: " + e.getMessage());

}

}

}

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. [10 Marks]

:-

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class PerfectNum {

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

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.print("Enter a number: ");

int num = Integer.parseInt(br.readLine());

if (isPerfectNumber(num)) {

System.out.println(num + " is a perfect number.");

} else {

System.out.println(num + " is not a perfect number.");

}

}

public static boolean isPerfectNumber(int num) {

if (num <= 1) {

return false;

}

int sum = 1;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

if (i \* i != num) {

sum = sum + i + num / i;

} else {

sum = sum + i;

}

}

}

return sum == num;

}

}

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.

:-

class Point {

int x;

int y;

// Default constructor

public Point() {

x = 0;

y = 0;

}

// Parameterized constructor

public Point(int x, int y) {

this.x = x;

this.y = y;

}

public void display() {

System.out.println("Point: (" + x + ", " + y + ")");

}

}

class Point3D extends Point {

private int z;

// Parameterized constructor for Point3D

public Point3D(int x, int y, int z) {

super(x, y); // Call the parameterized constructor of the superclass

this.z = z;

}

@Override

public void display() {

System.out.println("Point3D: (" + super.x + ", " + super.y + ", " + z + ")");

}

}

class Main {

public static void main(String[] args) {

Point point2D = new Point(1, 2);

Point3D point3D = new Point3D(3, 4, 5);

point2D.display(); // Display the 2D point

point3D.display(); // Display the 3D point

}

}

OR

Q2. Write a program that displays the number of characters, lines and words

Slip 10

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

:-

import java.util.Scanner;

public class Primechecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

if (isPrime(number)) {

System.out.println(number + " is a prime number.");

} else {

System.out.println(number + " is not a prime number.");

}

scanner.close();

}

// Function to check if a number is prime

public static boolean isPrime(int number) {

if (number <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(number); i++) {

if (number % i == 0) {

return false;

}

}

return true;

}

}

Q2. Create a package “utility”. Define a class CapitalString under “utility” package which will contain

a method to return String with first letter capital. Create a Person class (members – name, city)

outside the package. Display the person name with first letter as capital by making use of

CapitalString.

:-

package utility;

public class CapitalString {

public static String capitalizeFirstLetter(String input) {

if (input == null || input.isEmpty()) {

return input;

}

return input.substring(0, 1).toUpperCase() + input.substring(1);

}

}

import utility.CapitalString;

class main

{

public static void main(String[] args) {

Person person=new Person("john","New York");

String capitalizedName=CapitalString.capitalizeFirstLetter(person.getName());

System.out.println("Persons Name:- "+capitalizedName);

System.out.println("Persons city:- "+person.getCity());

}

}

public class Person {

private String name;

private String city;

public Person(String name, String city) {

this.name = name;

this.city = city;

}

public String getName() {

return name;

}

public String getCity() {

return city;

}

}

OR

Q2. Define a class SavingAccount (acno, name, balance). Define appropriate operations as, withdraw(), deposit(), and viewbalance(). The minimum balance must be 500. Create an objectand

perform operation. Raise user defined ―InsufficientFundException when balance is notsufficient

for withdraw operation.

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)

:-

public class MyDate {

private int dd;

private int mm;

private int yy;

public MyDate(int dd, int mm, int yy) {

this.dd = dd;

this.mm = mm;

this.yy = yy;

}

public void displayDate() {

System.out.printf("%02d-%02d-%02d%n", this.dd, this.mm, this.yy);

}

public static void main(String[] args) {

MyDate date = new MyDate(4, 11, 2023);

date.displayDate();

}

}

Q2. Create an abstract class Shape with methods area & volume. Derive a class Sphere (radius). Calculate and display area and volume. [20 Marks]

:-

import java.lang.Math;

// Abstract class Shape

abstract class Shape {

// Abstract method for calculating area

abstract double area();

// Abstract method for calculating volume

abstract double volume();

}

// Derived class Sphere

class Sphere extends Shape {

private double radius;

// Constructor to initialize the radius

public Sphere(double radius) {

this.radius = radius;

}

// Implementation of the area method for a sphere

@Override

double area() {

return 4 \* Math.PI \* Math.pow(radius, 2);

}

// Implementation of the volume method for a sphere

@Override

double volume() {

return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

}

}

class Main {

public static void main(String[] args) {

double radius = 5.0; // Replace with your desired radius

// Create a Sphere object

Sphere sphere = new Sphere(radius);

// Calculate and display the area and volume of the sphere

System.out.println("Sphere Area: " + sphere.area());

System.out.println("Sphere Volume: " + sphere.volume());

}

}

OR

Q2. Write a program to accept details of 'n' customers (c\_id, cname, address, mobile\_no) from

user and store it in a file using DataOutputStream class.

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.

:-

package Series;

public class SquareSeries {

public static void printSquareSeries(int n) {

for (int i = 1; i <= n; i++) {

int square = i \* i;

System.out.print(square + " ");

}

}

}

//import Series.SquareSeries;

class Main {

public static void main(String[] args) {

int n = 10; // You can change 'n' to the number of terms you want in the series

System.out.println("Square Series for " + n + " terms:");

SquareSeries.printSquareSeries(n);

}

}

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

:-

abstract class Shape {

// Abstract method to calculate area (to be implemented by subclasses)

public abstract double area();

// Abstract method to calculate volume (to be implemented by subclasses)

public abstract double volume();

}

class Cylinder extends Shape {

private double radius;

private double height;

// Constructor for the Cylinder class

public Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

}

@Override

public double area() {

// Calculate the surface area of the cylinder

return 2 \* Math.PI \* radius \* (radius + height);

}

@Override

public double volume() {

// Calculate the volume of the cylinder

return Math.PI \* radius \* radius \* height;

}

}

class Main {

public static void main(String[] args) {

// Create a Cylinder object with a radius of 3.0 and a height of 5.0

Cylinder myCylinder = new Cylinder(3.0, 5.0);

// Calculate and print the area and volume of the cylinder

System.out.println("Cylinder Area: " + myCylinder.area());

System.out.println("Cylinder Volume: " + myCylinder.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.LinkedList;

import java.util.Iterator;

public class Fruits {

public static void main(String[] args) {

// Create a LinkedList of fruit names

LinkedList<String> fruitList = new LinkedList<>();

// Add fruit names to the list

fruitList.add("Apple");

fruitList.add("Banana");

fruitList.add("Guava");

fruitList.add("Orange");

// Create an iterator to traverse the list

Iterator<String> iterator = fruitList.iterator();

// Display the contents using the iterator

System.out.println("Fruit List:");

while (iterator.hasNext()) {

String fruit = iterator.next();

System.out.println(fruit);

}

}

}

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.

:-

interface Operation {

double PI = 3.142; // Constant PI

double area(); // Method to calculate area

double volume(); // Method to calculate volume

}

class Circle implements Operation {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double area() {

// Calculate the area of the circle

return PI \* radius \* radius;

}

@Override

public double volume() {

// Volume doesn't apply to a 2D circle, so return 0.

return 0;

}

public void displayArea() {

System.out.println("Area of the circle: " + area());

}

public void displayVolume() {

System.out.println("Volume of the circle: " + volume());

}

public static void main(String[] args) {

double circleRadius = 5.0; // Replace with your desired radius

Circle circle = new Circle(circleRadius);

circle.displayArea();

circle.displayVolume();

}

}

OR

Q2. 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 userdefined exception ―Age Not Within The Range.

Slip 14

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

appropriate message to user.

:-

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class JDBCconnect {

public static void main(String[] args) {

String jdbcURL = "jdbc:mysql://localhost:3306/your\_database";

String username = "your\_username";

String password = "your\_password";

try {

// Load the JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Establish the connection

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/your\_database","your\_username","your\_password");

if (connection != null) {

System.out.println("Connected to the database successfully!");

// You can perform database operations here

// Don't forget to close the connection when you're done

connection.close();

} else {

System.out.println("Failed to connect to the database.");

}

} catch (ClassNotFoundException e) {

System.out.println("JDBC driver not found");

e.printStackTrace();

} catch (SQLException e) {

System.out.println("Connection to the database failed");

e.printStackTrace();

}

}

}

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. [20 Marks]

:-

interface Operation {

double PI = 3.142; // Constant PI

double area(); // Method to calculate area

double volume(); // Method to calculate volume

}

class Cylinder implements Operation {

private double radius;

private double height;

public Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

}

@Override

public double area() {

// Formula for the surface area of a cylinder: 2 \* PI \* radius \* (radius + height)

return 2 \* PI \* radius \* (radius + height);

}

@Override

public double volume() {

// Formula for the volume of a cylinder: PI \* radius^2 \* height

return PI \* Math.pow(radius, 2) \* height;

}

public void displayResults() {

System.out.println("Cylinder Area: " + area());

System.out.println("Cylinder Volume: " + volume());

}

}

class Main {

public static void main(String[] args) {

double radius = 5.0;

double height = 10.0;

Cylinder cylinder = new Cylinder(radius, height);

cylinder.displayResults();

}

}

OR

Q2. Write a class Student with attributes roll no, name, age and course. Initialize values through

parameterized constructor. If student's roll no of is not in between 13001 to 13080 thengenerate

user- defined exception ―Rollno is Not Within The Range

Slip 15

Q1. Construct a Linked List having names of Fruits: Apple, Banana, Guava and Orange. Displaythe

contents of the List in reverse order using an appropriate interface.

:-

import java.util.LinkedList;

import java.util.List;

import java.util.ListIterator;

public class Fruits {

public static void main(String[] args) {

// Create a linked list of fruit names

List<String> fruitList = new LinkedList<>();

fruitList.add("Apple");

fruitList.add("Banana");

fruitList.add("Guava");

fruitList.add("Orange");

// Use ListIterator to traverse the list in reverse order

ListIterator<String> listIterator = fruitList.listIterator(fruitList.size());

System.out.println("Fruit List in Reverse Order:");

while (listIterator.hasPrevious()) {

System.out.println(listIterator.previous());

}

}

}

Q2. Write a program to create a super class Employee (members – name, salary). Derive a sub- classas

Developer (member – projectname). Create object of Developer and display the detailsof it.

:-

class Employee {

String name;

double salary;

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

public void displayDetails() {

System.out.println("Employee Name: " + name);

System.out.println("Salary: $" + salary);

}

}

class Developer extends Employee {

String projectName;

public Developer(String name, double salary, String projectName) {

super(name, salary);

this.projectName = projectName;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Project Name: " + projectName);

}

}

class Main {

public static void main(String[] args) {

Developer developer = new Developer("John Doe", 60000.0, "Project X");

developer.displayDetails();

}

}

OR

Q2. Design a servlet to display message as “Welcome IP address

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.

:-

public class MyNumEven {

private int value;

public MyNumEven(int value) {

this.value = value;

}

public boolean isEven() {

return value % 2 == 0;

}

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Please provide exactly one integer as a command line argument.");

return;

}

try {

int inputValue = Integer.parseInt(args[0]);

MyNumEven myNumber = new MyNumEven(inputValue);

if (myNumber.isEven()) {

System.out.println(inputValue + " is an even number.");

} else {

System.out.println(inputValue + " is not an even number.");

}

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid integer.");

}

}

}

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.

:-

class SuperEmp {

String name;

double salary;

public SuperEmp(String name, double salary) {

this.name = name;

this.salary = salary;

}

}

class Programmer extends SuperEmp {

String proglanguage;

public Programmer(String name, double salary, String proglanguage) {

super(name, salary);

this.proglanguage = proglanguage;

}

public void displayDetails() {

System.out.println("Name: " + name);

System.out.println("Salary: " + salary);

System.out.println("Programming Language: " + proglanguage);

}

}

class Main {

public static void main(String[] args) {

Programmer programmer = new Programmer("John Doe", 60000.0, "Java");

programmer.displayDetails();

}

}

OR

Q2. Write a JDBC program to update number\_of\_students of “BCA Science” to 1000. Create a

table Course (Code,name, department,number\_of\_students) in PostgreSQL database. Insertvalues

in the table

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 commandline

argument to pass a value to the object.

:-

public class MyNumber {

private int number;

public MyNumber(int value) {

number = value;

}

public boolean isOdd() {

return number % 2 != 0;

}

public static void main(String[] args) {

if (args.length == 1) {

try {

int value = Integer.parseInt(args[0]);

MyNumber myNumber = new MyNumber(value);

if (myNumber.isOdd()) {

System.out.println(value + " is an odd number.");

} else {

System.out.println(value + " is not an odd number.");

}

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please provide a valid integer.");

}

} else {

System.out.println("Usage: java MyNumber <integer>");

}

}

}

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.

:-

class Student {

private int rollno;

private String name;

private static int objectCount = 0;

public Student() {

objectCount++;

}

public Student(int rollno, String name) {

this.rollno = rollno;

this.name = name;

objectCount++;

}

public static int getObjectCount() {

return objectCount;

}

}

class Main {

public static void main(String[] args) {

Student student1 = new Student(); // Create an object using the default constructor

System.out.println("Object count after student1: " + Student.getObjectCount());

Student student2 = new Student(101, "Alice"); // Create an object using the parameterized constructor

System.out.println("Object count after student2: " + Student.getObjectCount());

Student student3 = new Student(102, "Bob"); // Create another object using the parameterized constructor

System.out.println("Object count after student3: " + Student.getObjectCount());

}

}

OR

Q2. Write a JSP program to perform Arithmetic operations such as Addition and Subtraction. Design a HTML to accept two numbers in text box and radio buttons to display operations. On

submit display result as per the selected operation on next page using JS

Slip 18

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

argument.

:-

public class Factorofnum {

public static void main(String[] args) {

if (args.length != 1) {

System.out.println("Usage: java FactorsOfNumber <number>");

return;

}

try {

int number = Integer.parseInt(args[0]);

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("Invalid input. Please enter a valid integer.");

}

}

}

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

as output.

:-

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class abctxtfile {

public static void main(String[] args) {

try {

// Specify the file path

String filePath = "abc.txt";

// Create a BufferedReader to read the file

BufferedReader reader = new BufferedReader(new FileReader(filePath));

String line;

StringBuilder content = new StringBuilder();

// Read the file line by line

while ((line = reader.readLine()) != null) {

content.append(line).append("\n");

}

// Close the file

reader.close();

// Convert the content to uppercase

String upperCaseContent = content.toString().toUpperCase();

// Display the content in uppercase

System.out.println(upperCaseContent);

} catch (IOException e) {

System.err.println("An error occurred: " + e.getMessage());

}

}

}

OR

Q2. Write a program to design following screen using swing component

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 Maxnum {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements (n): ");

int n = scanner.nextInt();

// Create an array to store the numbers

int[] numbers = new int[n];

// Input 'n' numbers from the user

for (int i = 0; i < n; i++) {

System.out.print("Enter number " + (i + 1) + ": ");

numbers[i] = scanner.nextInt();

}

// Find the maximum number in the array

int max = numbers[0];

for (int i = 1; i < n; i++) {

if (numbers[i] > max) {

max = numbers[i];

}

}

System.out.println("Maximum number in the array: " + max);

}

}

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.

:-

abstract class Order {

protected int id;

protected String description;

public Order(int id, String description) {

this.id = id;

this.description = description;

}

public abstract void accept();

public abstract void display();

}

class PurchaseOrder extends Order {

private String customerName;

public PurchaseOrder(int id, String description, String customerName) {

super(id, description);

this.customerName = customerName;

}

@Override

public void accept() {

// You can implement the input logic here to accept the customer name

}

@Override

public void display() {

System.out.println("Order ID: " + id);

System.out.println("Description: " + description);

System.out.println("Customer Name: " + customerName);

}

}

class Main {

public static void main(String[] args) {

PurchaseOrder order1 = new PurchaseOrder(1, "First Order", "John Doe");

PurchaseOrder order2 = new PurchaseOrder(2, "Second Order", "Jane Smith");

PurchaseOrder order3 = new PurchaseOrder(3, "Third Order", "Alice Johnson");

order1.accept(); // You can implement logic to accept details here

order2.accept();

order3.accept();

order1.display();

order2.display();

order3.display();

}

}

OR

Q2. Write a servlet program to display current date and time

Slip 20

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

numbers.

:-

public class accept3num {

public static void main(String[] args) {

if (args.length != 3) {

System.out.println("Usage: java SortNumbers <num1> <num2> <num3>");

return;

}

try {

int num1 = Integer.parseInt(args[0]);

int num2 = Integer.parseInt(args[1]);

int num3 = Integer.parseInt(args[2]);

// Sort the numbers in ascending order

int[] sortedNumbers = { num1, num2, num3 };

java.util.Arrays.sort(sortedNumbers);

System.out.println("Sorted Numbers: " + sortedNumbers[0] + ", " + sortedNumbers[1] + ", " + sortedNumbers[2]);

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter valid integers.");

}

}

}

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 Employee {

private int id;

private String name;

private String deptName;

private double salary;

private static int objectCount = 0;

public Employee() {

objectCount++;

}

public Employee(int id, String name, String deptName, double salary) {

this(); // Calls the default constructor to increment objectCount

this.id = id;

this.name = name;

this.deptName = deptName;

this.salary = salary;

displayObjectDetails();

}

public void displayObjectDetails() {

System.out.println("Employee ID: " + id);

System.out.println("Employee Name: " + name);

System.out.println("Department Name: " + deptName);

System.out.println("Salary: $" + salary);

System.out.println("Total Employee Objects Created: " + objectCount);

System.out.println("---------------");

}

public static void main(String[] args) {

Employee emp1 = new Employee(1, "John Doe", "HR", 50000);

Employee emp2 = new Employee(2, "Alice Smith", "IT", 60000);

}

}

OR

Q2. Write a JSP program to perform Arithmetic operations such as Multiplication and Divison. Design a HTML to accept two numbers in text box and radio buttons to display operations. On

submit display result as per the selected operation on next page using