JAVA PRACTICAL SLIPS ANSWERS.

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

number. Accept number using Buffered Reader class.

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class MultiplicationTable {

public static void main(String[] args) {

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

try {

// Prompt user to enter a number

System.out.print("Enter a number to generate its multiplication table: ");

int number = Integer.parseInt(reader.readLine());

// Generate multiplication table

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

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

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

}

} catch (IOException e) {

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

} catch (NumberFormatException e) {

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

}

}

}

Q.2) 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 {

private int number;

// Default constructor initializing to 0

public MyNumber() {

this.number = 0;

}

// Constructor to initialize with a value

public MyNumber(int number) {

this.number = number;

}

// Method to check if the number is negative

public boolean isNegative() {

return number < 0;

}

// Method to check if the number is positive

public boolean isPositive() {

return number > 0;

}

// Method to check if the number is odd

public boolean isOdd() {

return number % 2 != 0;

}

// Method to check if the number is even

public boolean isEven() {

return number % 2 == 0;

}

// Main method to use command-line arguments

public static void main(String[] args) {

if (args.length == 0) {

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

return;

}

try {

// Parse the command-line argument

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

// Create MyNumber object using the value

MyNumber myNumber = new MyNumber(inputNumber);

// Perform operations and display results

System.out.println("Number: " + inputNumber);

System.out.println("Is Negative? " + myNumber.isNegative());

System.out.println("Is Positive? " + myNumber.isPositive());

System.out.println("Is Odd? " + myNumber.isOdd());

System.out.println("Is Even? " + myNumber.isEven());

} catch (NumberFormatException e) {

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

}

}

}

Q3.Java program that creates a GUI with the required components using **Swing**. The program includes fields for *User Name*, *Mobile Number*, *Class (FY, SY, TY)*, *Gender*, and buttons for *Save*, *Update*, and *Delete*.

import javax.swing.\*;

import java.awt.\*;

public class StudentInfoGUI {

public static void main(String[] args) {

// Create a frame

JFrame frame = new JFrame("Student Information Form");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

frame.setLayout(new GridLayout(5, 2, 10, 10)); // GridLayout for organized UI

// Components for User Name

JLabel lblUserName = new JLabel("User Name:");

JTextField txtUserName = new JTextField();

// Components for Mobile Number

JLabel lblMobileNumber = new JLabel("Mobile Number:");

JTextField txtMobileNumber = new JTextField();

// Components for Class

JLabel lblClass = new JLabel("Class:");

JPanel classPanel = new JPanel();

classPanel.setLayout(new FlowLayout(FlowLayout.LEFT));

JRadioButton fyButton = new JRadioButton("FY");

JRadioButton syButton = new JRadioButton("SY");

JRadioButton tyButton = new JRadioButton("TY");

ButtonGroup classGroup = new ButtonGroup();

classGroup.add(fyButton);

classGroup.add(syButton);

classGroup.add(tyButton);

classPanel.add(fyButton);

classPanel.add(syButton);

classPanel.add(tyButton);

// Components for Gender

JLabel lblGender = new JLabel("Gender:");

JPanel genderPanel = new JPanel();

genderPanel.setLayout(new FlowLayout(FlowLayout.LEFT));

JRadioButton maleButton = new JRadioButton("Male");

JRadioButton femaleButton = new JRadioButton("Female");

JRadioButton otherButton = new JRadioButton("Other");

ButtonGroup genderGroup = new ButtonGroup();

genderGroup.add(maleButton);

genderGroup.add(femaleButton);

genderGroup.add(otherButton);

genderPanel.add(maleButton);

genderPanel.add(femaleButton);

genderPanel.add(otherButton);

// Buttons for Save, Update, and Delete

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER));

JButton saveButton = new JButton("Save");

JButton updateButton = new JButton("Update");

JButton deleteButton = new JButton("Delete");

buttonPanel.add(saveButton);

buttonPanel.add(updateButton);

buttonPanel.add(deleteButton);

// Add components to the frame

frame.add(lblUserName);

frame.add(txtUserName);

frame.add(lblMobileNumber);

frame.add(txtMobileNumber);

frame.add(lblClass);

frame.add(classPanel);

frame.add(lblGender);

frame.add(genderPanel);

frame.add(buttonPanel); // Add buttons

// Display the frame

frame.setVisible(true);

}

}

Q4. Write a Java Program to Reverse a Number. Accept number using command line arguments.

public class ReverseNumber {

public static void main(String[] args) {

// Check if a command-line argument is provided

if (args.length == 0) {

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

return;

}

try {

// Parse the input number from command-line arguments

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

int reversed = 0;

int original = number;

// Reverse the number

while (number != 0) {

int digit = number % 10; // Extract the last digit

reversed = reversed \* 10 + digit; // Build the reversed number

number /= 10; // Remove the last digit

}

// Print the reversed number

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

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

} catch (NumberFormatException e) {

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

}

}

}

Q.5) 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.

import java.util.Arrays;

import java.util.Scanner;

class Account {

private int accno;

private String accname;

private double balance;

// Constructor to initialize account details

public Account(int accno, String accname, double balance) {

this.accno = accno;

this.accname = accname;

this.balance = balance;

}

// Getter for balance

public double getBalance() {

return balance;

}

// Method to display account details

public void displayAccount() {

System.out.printf("Account Number: %d, Account Name: %s, Balance: %.2f%n", accno, accname, balance);

}

// Static method to sort accounts based on balance

public static void sortAccount(Account[] accounts) {

Arrays.sort(accounts, (a, b) -> Double.compare(a.getBalance(), b.getBalance()));

}

}

public class AccountManager {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for the number of accounts

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

int n = scanner.nextInt();

// Create an array of Account objects

Account[] accounts = new Account[n];

// Input details for each account

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

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

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

int accno = scanner.nextInt();

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

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

String accname = scanner.nextLine();

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

double balance = scanner.nextDouble();

// Create a new Account object and store it in the array

accounts[i] = new Account(accno, accname, balance);

}

// Sort accounts by balance

Account.sortAccount(accounts);

// Display sorted accounts

System.out.println("\nAccounts sorted by balance:");

for (Account account : accounts) {

account.displayAccount();

}

scanner.close();

}

}

Q.6) 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. If name contains numbers or special symbols raise

exception ―Name not valid

// Custom exception for invalid age

class AgeNotWithinRangeException extends Exception {

public AgeNotWithinRangeException(String message) {

super(message);

}

}

// Custom exception for invalid name

class NameNotValidException extends Exception {

public NameNotValidException(String message) {

super(message);

}

}

class Student {

private int rollNo;

private String name;

private int age;

private String course;

// Constructor with validation

public Student(int rollNo, String name, int age, String course) throws AgeNotWithinRangeException, NameNotValidException {

if (age < 15 || age > 21) {

throw new AgeNotWithinRangeException("Age Not Within The Range (15-21)");

}

if (!name.matches("[a-zA-Z ]+")) {

throw new NameNotValidException("Name Not Valid (contains numbers or special symbols)");

}

this.rollNo = rollNo;

this.name = name;

this.age = age;

this.course = course;

}

// Method to display student details

public void displayStudentDetails() {

System.out.println("Roll No: " + rollNo);

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

System.out.println("Age: " + age);

System.out.println("Course: " + course);

}

}

public class StudentTest {

public static void main(String[] args) {

try {

// Input student details

Student student = new Student(101, "John Doe", 20, "Computer Science");

student.displayStudentDetails();

// Example of invalid name

Student invalidNameStudent = new Student(102, "Jane@Doe", 19, "Mathematics");

// Example of invalid age

Student invalidAgeStudent = new Student(103, "Alice Smith", 22, "Physics");

} catch (AgeNotWithinRangeException e) {

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

} catch (NameNotValidException e) {

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

}

}

}

Q.7) Write a Java program to print the sum of elements of the array. Also display array elements in

ascending order.

import java.util.Arrays;

import java.util.Scanner;

public class ArrayOperations {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input array size

System.out.print("Enter the size of the array: ");

int n = scanner.nextInt();

// Create array

int[] array = new int[n];

// Input array elements

System.out.println("Enter the elements of the array:");

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

array[i] = scanner.nextInt();

}

// Calculate sum of array elements

int sum = 0;

for (int num : array) {

sum += num;

}

// Sort the array in ascending order

Arrays.sort(array);

// Display the results

System.out.println("Sum of array elements: " + sum);

System.out.println("Array elements in ascending order: " + Arrays.toString(array));

scanner.close();

}

}

Q.8) 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 {

private int id;

private String name;

private double price;

// Constructor to initialize product details

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

this.id = id;

this.name = name;

this.price = price;

}

// Getter for price

public double getPrice() {

return price;

}

// Getter for name

public String getName() {

return name;

}

// Method to display product details

public void displayProduct() {

System.out.printf("ID: %d, Name: %s, Price: %.2f%n", id, name, price);

}

}

public class ProductTest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Create an array to store 5 products

Product[] products = new Product[5];

// Input product details

for (int i = 0; i < products.length; i++) {

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

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

int id = scanner.nextInt();

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

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

String name = scanner.nextLine();

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

double price = scanner.nextDouble();

// Create a Product object and store it in the array

products[i] = new Product(id, name, price);

}

// Find the product with the minimum price

Product minPriceProduct = products[0];

for (Product product : products) {

if (product.getPrice() < minPriceProduct.getPrice()) {

minPriceProduct = product;

}

}

// Display the product with the minimum price

System.out.println("\nProduct with the minimum price:");

minPriceProduct.displayProduct();

scanner.close();

}

}

Q.9) Define an Interface Shape with abstract method area (). Write a java program to calculate an area of Circle and Sphere. (Use final keyword).

// Define the Shape interface

interface Shape {

double area(); // Abstract method to calculate area

}

// Class to calculate the area of a Circle

class Circle implements Shape {

private final double radius;

// Constructor to initialize the radius

public Circle(double radius) {

this.radius = radius;

}

// Implement the area method

@Override

public double area() {

return Math.PI \* radius \* radius; // Area of Circle = πr²

}

}

// Class to calculate the surface area of a Sphere

class Sphere implements Shape {

private final double radius;

// Constructor to initialize the radius

public Sphere(double radius) {

this.radius = radius;

}

// Implement the area method

@Override

public double area() {

return 4 \* Math.PI \* radius \* radius; // Surface Area of Sphere = 4πr²

}

}

public class ShapeTest {

public static void main(String[] args) {

// Define radius for Circle and Sphere

final double circleRadius = 5.0; // Use final keyword for constant values

final double sphereRadius = 7.0;

// Create objects for Circle and Sphere

Shape circle = new Circle(circleRadius);

Shape sphere = new Sphere(sphereRadius);

// Display areas

System.out.printf("Area of Circle with radius %.2f: %.2f%n", circleRadius, circle.area());

System.out.printf("Surface Area of Sphere with radius %.2f: %.2f%n", sphereRadius, sphere.area());

}

}

Q.10) Write a Java program to print the factors of a given number. (Use Scanner class).

import java.util.Scanner;

public class FactorPrinter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter a number

System.out.print("Enter a number to find its factors: ");

int number = scanner.nextInt();

// Check for valid input

if (number <= 0) {

System.out.println("Please enter a positive integer.");

} else {

System.out.println("Factors of " + number + " are:");

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

if (number % i == 0) {

System.out.println(i);

}

}

}

scanner.close();

}

}

Q.11) 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) {

// Creating a linked list with color names

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

colors.add("red");

colors.add("blue");

colors.add("yellow");

colors.add("orange");

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

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

Iterator<String> iterator = colors.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 ListIterator:");

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

while (listIterator.hasPrevious()) {

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

}

// iii. Create another list containing pink and green

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

newColors.add("pink");

newColors.add("green");

// Insert the elements of the new list between blue and yellow

int index = colors.indexOf("blue") + 1; // Find the position right after "blue"

colors.addAll(index, newColors); // Add the new list between blue and yellow

// Display the modified list

System.out.println("\nModified List after inserting pink and green:");

for (String color : colors) {

System.out.println(color);

}

}

}

Q.12) Write a program which define class Employee with data member as id, name and salary

Store the information of &#39;n&#39; employees and display the name of employee having maximum salary (Use array of object).

import java.util.Scanner;

class Employee {

private int id;

private String name;

private double salary;

// Constructor to initialize Employee details

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

this.id = id;

this.name = name;

this.salary = salary;

}

// Getter for salary

public double getSalary() {

return salary;

}

// Getter for name

public String getName() {

return name;

}

}

public class EmployeeTest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input number of employees

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

int n = scanner.nextInt();

scanner.nextLine(); // Consume newline left by nextInt()

// Create an array to store 'n' employee objects

Employee[] employees = new Employee[n];

// Input employee details

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

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

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

int id = scanner.nextInt();

scanner.nextLine(); // Consume newline

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

String name = scanner.nextLine();

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

double salary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

// Create an Employee object and store it in the array

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].getSalary() > maxSalaryEmployee.getSalary()) {

maxSalaryEmployee = employees[i];

}

}

// Display the name of the employee with the maximum salary

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

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

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

scanner.close();

}

}

Q.13) Write a Java program to accept a number from user and print all prime numbers up to that

number (Use Buffered Reader class).

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class PrimeNumbers {

public static void main(String[] args) {

try {

// Create BufferedReader to read input from user

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

// Accept input from user

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

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

// Print prime numbers up to the given number

System.out.println("Prime numbers up to " + num + ":");

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

if (isPrime(i)) {

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

}

}

} catch (IOException e) {

System.out.println("Error reading input.");

}

}

// Method to check if a number is prime

public static boolean isPrime(int number) {

if (number <= 1) {

return false; // 0 and 1 are not prime

}

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

if (number % i == 0) {

return false; // If divisible by any number other than 1 and itself, not prime

}

}

return true; // Number is prime

}

}

Q.14) Create a package “utility”. Define a class Capital String 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’s name with first letter as capital by

making use of Capital String.

// File: utility/CapitalString.java

package utility;

public class CapitalString {

// Method to capitalize the first letter of a string

public static String capitalizeFirstLetter(String str) {

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

return str; // Return the string as is if it's null or empty

}

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

}

}

// File: Person.java

import utility.CapitalString;

public class Person {

private String name;

private String city;

// Constructor to initialize name and city

public Person(String name, String city) {

this.name = name;

this.city = city;

}

// Method to display person's details with capitalized name

public void displayPersonInfo() {

// Capitalizing the first letter of the name using CapitalString class

String capitalizedName = CapitalString.capitalizeFirstLetter(this.name);

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

System.out.println("City: " + this.city);

}

public static void main(String[] args) {

// Creating a Person object

Person person = new Person("john doe", "new york");

// Displaying the person's details

person.displayPersonInfo();

}

}

Q.15) Create an abstract class Shape with methods area &amp; volume. Derive a class Cylinder

(radius, height). Calculate area and volume

abstract class Shape {

// Abstract method to calculate area

abstract double area();

// Abstract method to calculate volume

abstract double volume();

}

import java.lang.Math;

class Cylinder extends Shape {

private double radius;

private double height;

// Constructor to initialize radius and height

public Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

}

// Implementing area method for Cylinder

@Override

public double area() {

return 2 \* Math.PI \* radius \* (radius + height); // Surface area of cylinder

}

// Implementing volume method for Cylinder

@Override

public double volume() {

return Math.PI \* radius \* radius \* height; // Volume of cylinder

}

}

public class Main {

public static void main(String[] args) {

// Create a Cylinder object with radius and height

Cylinder cylinder = new Cylinder(5, 10);

// Display the area and volume of the cylinder

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

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

}

}

Q.16) Write a Java Program to Display Armstrong Numbers Between range. Accept range from

user.

import java.util.Scanner;

public class ArmstrongNumbers {

// Function to check if a number is Armstrong number

public static boolean isArmstrong(int number) {

int originalNumber = number;

int sum = 0;

int numberOfDigits = String.valueOf(number).length(); // Find the number of digits

// Calculate the sum of each digit raised to the power of the number of digits

while (number > 0) {

int digit = number % 10;

sum += Math.pow(digit, numberOfDigits);

number /= 10;

}

// Check if sum is equal to the original number

return sum == originalNumber;

}

public static void main(String[] args) {

// Create a Scanner object to read input

Scanner scanner = new Scanner(System.in);

// Accept range from the user

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

int start = scanner.nextInt();

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

int end = scanner.nextInt();

// Display Armstrong numbers in the given range

System.out.println("Armstrong numbers between " + start + " and " + end + " are:");

for (int num = start; num <= end; num++) {

if (isArmstrong(num)) {

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

}

}

// Close the scanner to avoid resource leak

scanner.close();

}

}

Q.17) 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).

class MyDate {

// Data members for day, month, and year

private int dd;

private int mm;

private int yy;

// Parameterized constructor to initialize the date

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

// Using 'this' keyword to refer to the current object's instance variables

this.dd = dd;

this.mm = mm;

this.yy = yy;

}

// Method to display the date in dd-mm-yy format

public void displayDate() {

System.out.println("Date: " + this.dd + "-" + this.mm + "-" + this.yy);

}

}

public class Main {

public static void main(String[] args) {

// Creating an object of MyDate class with parameterized constructor

MyDate date = new MyDate(25, 12, 2024);

// Calling displayDate method to print the date

date.displayDate();

}

}

Q.18) Write a Java program to create a super class Employee (members – name, salary). Derive a

sub-class as Developer (member – projectname). Derive a sub-class Programmer (member –

proglanguage) from Developer. Create object of Programmer and display the details of it.

Implement this multilevel inheritance with appropriate constructor and methods.

// Superclass Employee

class Employee {

// Members of Employee class

protected String name;

protected double salary;

// Constructor of Employee class

public Employee(String name, double salary) {

this.name = name;

this.salary = salary;

}

// Method to display Employee details

public void displayEmployeeDetails() {

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

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

}

}

// Subclass Developer, inheriting from Employee

class Developer extends Employee {

// Additional member for Developer class

protected String projectName;

// Constructor of Developer class

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

// Calling superclass (Employee) constructor using 'super' keyword

super(name, salary);

this.projectName = projectName;

}

// Method to display Developer details (in addition to Employee details)

public void displayDeveloperDetails() {

displayEmployeeDetails(); // Displaying Employee details

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

}

}

// Subclass Programmer, inheriting from Developer

class Programmer extends Developer {

// Additional member for Programmer class

private String progLanguage;

// Constructor of Programmer class

public Programmer(String name, double salary, String projectName, String progLanguage) {

// Calling Developer class constructor using 'super' keyword

super(name, salary, projectName);

this.progLanguage = progLanguage;

}

// Method to display Programmer details (including Employee and Developer details)

public void displayProgrammerDetails() {

displayDeveloperDetails(); // Displaying Developer details (which already includes Employee details)

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

}

}

public class Main {

public static void main(String[] args) {

// Creating an object of Programmer class

Programmer programmer = new Programmer("John Doe", 75000.0, "E-commerce Website", "Java");

// Displaying the details of the Programmer

programmer.displayProgrammerDetails();

}

}

Q.19) Write java program to check whether number is Perfect or not.

import java.util.Scanner;

public class PerfectNumber {

// Method to check if a number is perfect

public static boolean isPerfectNumber(int num) {

int sum = 0;

// Find divisors of num and calculate their sum (excluding the number itself)

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i; // Add divisor to sum

}

}

// Check if the sum of divisors equals the number

return sum == num;

}

public static void main(String[] args) {

// Create a scanner object to read input

Scanner scanner = new Scanner(System.in);

// Accept a number from the user

System.out.print("Enter a number to check if it is a Perfect Number: ");

int number = scanner.nextInt();

// Check if the number is perfect and display the result

if (isPerfectNumber(number)) {

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

} else {

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

}

// Close the scanner to avoid resource leak

scanner.close();

}

}

Q.20) 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 {

// Attributes of the class

private int rollno;

private String name;

// Static variable to keep track of the count of objects created

static int objectCount = 0;

// Default constructor

public Student() {

rollno = 0;

name = "Unknown";

objectCount++; // Increment the object count

}

// Parameterized constructor

public Student(int rollno, String name) {

this.rollno = rollno;

this.name = name;

objectCount++; // Increment the object count

}

// Method to display the details of the student

public void displayStudentDetails() {

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

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

}

// Static method to display the count of objects created

public static void displayObjectCount() {

System.out.println("Number of Student objects created: " + objectCount);

}

}

public class Main {

public static void main(String[] args) {

// Create student objects using parameterized constructor

Student student1 = new Student(101, "John Doe");

student1.displayStudentDetails();

Student.displayObjectCount(); // Display the object count

System.out.println();

// Create another student object

Student student2 = new Student(102, "Jane Smith");

student2.displayStudentDetails();

Student.displayObjectCount(); // Display the object count

System.out.println();

// Create another student object

Student student3 = new Student(103, "Alice Brown");

student3.displayStudentDetails();

Student.displayObjectCount(); // Display the object count

System.out.println();

// Create a student object using default constructor

Student student4 = new Student();

student4.displayStudentDetails();

Student.displayObjectCount(); // Display the object count

}

}

Q.21) Write a java program to accept details of n customers (c\_id, cname, address, mobile\_no) from user and store it in a file (Use DataOutputStream class). Display the details of customers by reading it from file. (Use DataInputStream class).

import java.io.\*;

import java.util.Scanner;

class Customer {

int c\_id;

String cname;

String address;

String mobile\_no;

// Constructor to initialize customer details

public Customer(int c\_id, String cname, String address, String mobile\_no) {

this.c\_id = c\_id;

this.cname = cname;

this.address = address;

this.mobile\_no = mobile\_no;

}

}

public class CustomerDetails {

// Method to write customer details to file using DataOutputStream

public static void writeCustomerDetailsToFile(DataOutputStream dos, Customer customer) throws IOException {

dos.writeInt(customer.c\_id);

dos.writeUTF(customer.cname);

dos.writeUTF(customer.address);

dos.writeUTF(customer.mobile\_no);

}

// Method to read customer details from file using DataInputStream

public static void readCustomerDetailsFromFile(DataInputStream dis) throws IOException {

try {

while (true) {

int c\_id = dis.readInt();

String cname = dis.readUTF();

String address = dis.readUTF();

String mobile\_no = dis.readUTF();

System.out.println("Customer ID: " + c\_id);

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

System.out.println("Address: " + address);

System.out.println("Mobile No: " + mobile\_no);

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

}

} catch (EOFException e) {

// End of file reached, catch the exception and stop reading

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String filename = "customer\_details.dat";

try {

// Create a DataOutputStream to write to the file

DataOutputStream dos = new DataOutputStream(new FileOutputStream(filename));

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

int n = scanner.nextInt();

scanner.nextLine(); // Consume newline left by nextInt()

// Accept and store customer details

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

System.out.println("\nEnter details for Customer " + (i + 1));

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

int c\_id = scanner.nextInt();

scanner.nextLine(); // Consume newline

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

String cname = scanner.nextLine();

System.out.print("Enter Customer Address: ");

String address = scanner.nextLine();

System.out.print("Enter Customer Mobile No: ");

String mobile\_no = scanner.nextLine();

// Create a Customer object and write to file

Customer customer = new Customer(c\_id, cname, address, mobile\_no);

writeCustomerDetailsToFile(dos, customer);

}

dos.close(); // Close DataOutputStream

// Create DataInputStream to read from the file

DataInputStream dis = new DataInputStream(new FileInputStream(filename));

System.out.println("\nCustomer Details from File:");

readCustomerDetailsFromFile(dis);

dis.close(); // Close DataInputStream

} catch (IOException e) {

e.printStackTrace();

} finally {

scanner.close();

}

}

}

Q.22) 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 {

// Instance variables (attributes)

int rollno;

String name;

float percentage;

// Default constructor

public Student() {

this.rollno = 0;

this.name = "Unknown";

this.percentage = 0.0f;

}

// Parameterized constructor

public Student(int rollno, String name, float percentage) {

this.rollno = rollno; // Using 'this' keyword to refer to instance variable

this.name = name;

this.percentage = percentage;

}

// Method to display student details

public void displayStudentDetails() {

System.out.println("Roll No: " + this.rollno);

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

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

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

}

}

public class StudentDetails {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Create an array to store 5 student objects

Student[] students = new Student[5];

// Accept details for 5 students

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

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

// Accept details from the user

System.out.print("Enter Roll No: ");

int rollno = scanner.nextInt();

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

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

String name = scanner.nextLine();

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

float percentage = scanner.nextFloat();

// Create a new Student object using the parameterized constructor

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

}

// Display the details of all students

System.out.println("\nStudent Details:");

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

students[i].displayStudentDetails();

}

// Close the scanner

scanner.close();

}

}

Q.23) Create an abstract class Shape with methods area &amp; volume. Derive a class Cylinder (radius,height). Calculate area and volume.

// Abstract class Shape

abstract class Shape {

// Abstract methods for area and volume

public abstract double area();

public abstract double volume();

}

// Derived class Cylinder

class Cylinder extends Shape {

// Attributes for the radius and height of the cylinder

double radius;

double height;

// Constructor to initialize radius and height

public Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

}

// Override the area method to calculate surface area of the cylinder

@Override

public double area() {

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

}

// Override the volume method to calculate volume of the cylinder

@Override

public double volume() {

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

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of Cylinder with radius and height

Cylinder cylinder = new Cylinder(5, 10);

// Display the area and volume of the cylinder

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

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

}

}

Q24.) Write a java program to design a following GUI. Use appropriate Layout and Components. with title Vaccination Details,name dose 1st dose,2nd dose,vaccine(Covishield,covaxin,sputnik V) Name:\_\_\_\_\_\_\_\_,Dose:1st\_\_\_\_\_\_ 2nd\_\_\_\_\_\_\_ , Vaccine:\_\_\_\_\_\_\_\_\_.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class VaccinationDetailsGUI {

public static void main(String[] args) {

// Create frame for the GUI

JFrame frame = new JFrame("Vaccination Details");

// Set the layout for the frame

frame.setLayout(new GridLayout(5, 2, 10, 10)); // 5 rows, 2 columns, 10px horizontal/vertical gap

// Create labels and text fields for the user inputs

JLabel nameLabel = new JLabel("Name:");

JLabel dose1Label = new JLabel("1st Dose:");

JLabel dose2Label = new JLabel("2nd Dose:");

JLabel vaccineLabel = new JLabel("Vaccine:");

JTextField nameField = new JTextField();

JTextField dose1Field = new JTextField();

JTextField dose2Field = new JTextField();

// Create a combo box with vaccine options

String[] vaccineOptions = {"Covishield", "Covaxin", "Sputnik V"};

JComboBox<String> vaccineComboBox = new JComboBox<>(vaccineOptions);

// Create a button to submit the details

JButton submitButton = new JButton("Submit");

// Add components to the frame

frame.add(nameLabel);

frame.add(nameField);

frame.add(dose1Label);

frame.add(dose1Field);

frame.add(dose2Label);

frame.add(dose2Field);

frame.add(vaccineLabel);

frame.add(vaccineComboBox);

frame.add(new JLabel()); // Empty cell to align the button

frame.add(submitButton);

// Set the size and visibility of the frame

frame.setSize(400, 250);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

// Action listener for the submit button

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Get the values from the input fields

String name = nameField.getText();

String dose1 = dose1Field.getText();

String dose2 = dose2Field.getText();

String vaccine = (String) vaccineComboBox.getSelectedItem();

// Display the collected vaccination details

JOptionPane.showMessageDialog(frame, "Vaccination Details:\n" +

"Name: " + name + "\n" +

"1st Dose: " + dose1 + "\n" +

"2nd Dose: " + dose2 + "\n" +

"Vaccine: " + vaccine);

}

});

}

}

Q.25) Write a Java program to display Fibonacci series using function.

import java.util.Scanner;

public class Fibonacci {

// Function to print Fibonacci series up to n terms

public static void fibonacciSeries(int n) {

int first = 0, second = 1, next;

// If the user wants to display the first term

if (n >= 1) {

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

}

// If the user wants to display at least the second term

if (n >= 2) {

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

}

// Loop to generate the Fibonacci series

for (int i = 3; i <= n; i++) {

next = first + second;

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

first = second; // Move second to first

second = next; // Move next to second

}

System.out.println(); // Move to the next line after printing the series

}

public static void main(String[] args) {

// Scanner to take user input

Scanner scanner = new Scanner(System.in);

// Ask the user for the number of terms

System.out.print("Enter the number of terms for Fibonacci series: ");

int terms = scanner.nextInt();

// Display the Fibonacci series

System.out.println("Fibonacci Series up to " + terms + " terms:");

fibonacciSeries(terms);

// Close the scanner

scanner.close();

}

}

Q.26) Define an interface “Operation” which has methods area (), volume (). Define a constant PI

having a value 3.142. Create a class circle (member – radius), cylinder (members – radius, height)

which implements this interface. Calculate and display the area and volume.

// Interface defining the Operation methods and constant PI

interface Operation {

double PI = 3.142; // Constant PI

// Abstract methods to be implemented by the classes

double area();

double volume();

}

// Circle class implementing Operation interface

class Circle implements Operation {

double radius;

// Constructor to initialize the radius

public Circle(double radius) {

this.radius = radius;

}

// Implementation of area() for circle

@Override

public double area() {

return PI \* radius \* radius; // Area of Circle = PI \* r^2

}

// The volume method is not applicable for Circle, so it returns 0

@Override

public double volume() {

return 0; // Circle doesn't have a volume

}

}

// Cylinder class implementing Operation interface

class Cylinder implements Operation {

double radius, height;

// Constructor to initialize radius and height

public Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

}

// Implementation of area() for cylinder

@Override

public double area() {

return 2 \* PI \* radius \* (radius + height); // Surface Area of Cylinder

}

// Implementation of volume() for cylinder

@Override

public double volume() {

return PI \* radius \* radius \* height; // Volume of Cylinder = PI \* r^2 \* h

}

}

public class Main {

public static void main(String[] args) {

// Create an object of Circle with radius 5

Circle circle = new Circle(5);

System.out.println("Circle Area: " + circle.area()); // Calculate and display area

System.out.println("Circle Volume: " + circle.volume()); // Volume is 0 for Circle

// Create an object of Cylinder with radius 5 and height 10

Cylinder cylinder = new Cylinder(5, 10);

System.out.println("Cylinder Surface Area: " + cylinder.area()); // Calculate and display surface area

System.out.println("Cylinder Volume: " + cylinder.volume()); // Calculate and display volume

}

}

Q.27) Write a package game which will have 2 classes Indoor &amp; Outdoor. Use a function display () to generate the list of players for the specific game. Use default &amp;parameterized constructor.

Create the Indoor.java

// File: Indoor.java

package game;

public class Indoor {

String[] indoorPlayers;

// Default constructor to initialize with default players

public Indoor() {

indoorPlayers = new String[] {"Player1", "Player2", "Player3"};

}

// Parameterized constructor to initialize with custom players

public Indoor(String[] players) {

indoorPlayers = players;

}

// Display method to print the list of indoor game players

public void display() {

System.out.println("Indoor Game Players:");

for (String player : indoorPlayers) {

System.out.println(player);

}

}

}

Create the Outdoor.java

// File: Outdoor.java

package game;

public class Outdoor {

String[] outdoorPlayers;

// Default constructor to initialize with default players

public Outdoor() {

outdoorPlayers = new String[] {"PlayerA", "PlayerB", "PlayerC"};

}

// Parameterized constructor to initialize with custom players

public Outdoor(String[] players) {

outdoorPlayers = players;

}

// Display method to print the list of outdoor game players

public void display() {

System.out.println("Outdoor Game Players:");

for (String player : outdoorPlayers) {

System.out.println(player);

}

}

}

Create game.java

// File: Main.java

import game.Indoor; // Import Indoor class from game package

import game.Outdoor; // Import Outdoor class from game package

public class Main {

public static void main(String[] args) {

// Using default constructor for Indoor

Indoor indoorGame = new Indoor();

indoorGame.display(); // Display players for indoor games

System.out.println(); // Print a blank line for separation

// Using default constructor for Outdoor

Outdoor outdoorGame = new Outdoor();

outdoorGame.display(); // Display players for outdoor games

System.out.println(); // Print a blank line for separation

// Using parameterized constructor for Indoor

String[] indoorCustomPlayers = {"John", "Alice", "Bob"};

Indoor customIndoorGame = new Indoor(indoorCustomPlayers);

customIndoorGame.display(); // Display custom indoor players

System.out.println(); // Print a blank line for separation

// Using parameterized constructor for Outdoor

String[] outdoorCustomPlayers = {"Mike", "Sarah", "Tom"};

Outdoor customOutdoorGame = new Outdoor(outdoorCustomPlayers);

customOutdoorGame.display(); // Display custom outdoor players

}

}

Q.28) Write a program that reads on file name from the user, then displays information about

whether the file exists, whether the file is readable, whether the file is writable and the type of file.

import java.io.File;

import java.util.Scanner;

public class FileInfo {

public static void main(String[] args) {

// Scanner to take user input

Scanner scanner = new Scanner(System.in);

// Ask the user to enter the file name

System.out.print("Enter the file name: ");

String fileName = scanner.nextLine();

// Create a File object using the input file name

File file = new File(fileName);

// Check if the file exists

if (file.exists()) {

System.out.println("File exists.");

// Check if the file is readable

if (file.canRead()) {

System.out.println("File is readable.");

} else {

System.out.println("File is not readable.");

}

// Check if the file is writable

if (file.canWrite()) {

System.out.println("File is writable.");

} else {

System.out.println("File is not writable.");

}

// Check if the file is a directory or a regular file

if (file.isDirectory()) {

System.out.println("It is a directory.");

} else if (file.isFile()) {

System.out.println("It is a regular file.");

}

} else {

System.out.println("File does not exist.");

}

// Close the scanner

scanner.close();

}

}

Q.29) Write a Java program to create a super class Vehicle having members Company and Price.

Derive two different classes LightMotorVehicle (mileage) and HeavyMotorVehicle

(capacity\_in\_tons). Accept the information for “n” vehicles and display the information in

appropriate form. While taking data, ask user about the type of vehicle first

import java.util.Scanner;

// Superclass Vehicle

class Vehicle {

String company;

double price;

// Constructor to initialize company and price

public Vehicle(String company, double price) {

this.company = company;

this.price = price;

}

// Method to display the basic details of a vehicle

public void displayVehicleInfo() {

System.out.println("Company: " + company);

System.out.println("Price: " + price);

}

}

// Subclass LightMotorVehicle

class LightMotorVehicle extends Vehicle {

double mileage;

// Constructor to initialize the company, price, and mileage

public LightMotorVehicle(String company, double price, double mileage) {

super(company, price); // Call to the superclass constructor

this.mileage = mileage;

}

// Method to display information of LightMotorVehicle

public void displayInfo() {

displayVehicleInfo(); // Calling the superclass method

System.out.println("Mileage: " + mileage + " km/l");

}

}

// Subclass HeavyMotorVehicle

class HeavyMotorVehicle extends Vehicle {

double capacityInTons;

// Constructor to initialize the company, price, and capacity

public HeavyMotorVehicle(String company, double price, double capacityInTons) {

super(company, price); // Call to the superclass constructor

this.capacityInTons = capacityInTons;

}

// Method to display information of HeavyMotorVehicle

public void displayInfo() {

displayVehicleInfo(); // Calling the superclass method

System.out.println("Capacity: " + capacityInTons + " tons");

}

}

public class VehicleInfo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ask user for the number of vehicles

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

int n = scanner.nextInt();

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

// Array to store the vehicle objects

Vehicle[] vehicles = new Vehicle[n];

// Input and creation of vehicles

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

System.out.println("\nEnter details for vehicle " + (i + 1) + ":");

System.out.print("Enter vehicle type (Light/Heavy): ");

String vehicleType = scanner.nextLine();

System.out.print("Enter company name: ");

String company = scanner.nextLine();

System.out.print("Enter price: ");

double price = scanner.nextDouble();

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

if (vehicleType.equalsIgnoreCase("Light")) {

System.out.print("Enter mileage: ");

double mileage = scanner.nextDouble();

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

vehicles[i] = new LightMotorVehicle(company, price, mileage);

} else if (vehicleType.equalsIgnoreCase("Heavy")) {

System.out.print("Enter capacity in tons: ");

double capacityInTons = scanner.nextDouble();

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

vehicles[i] = new HeavyMotorVehicle(company, price, capacityInTons);

} else {

System.out.println("Invalid vehicle type. Please enter either 'Light' or 'Heavy'.");

i--; // Decrement the index to retry this vehicle

}

}

// Displaying the information of all vehicles

System.out.println("\nVehicle Information:");

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

System.out.println("\nDetails of Vehicle " + (i + 1) + ":");

if (vehicles[i] instanceof LightMotorVehicle) {

((LightMotorVehicle) vehicles[i]).displayInfo();

} else if (vehicles[i] instanceof HeavyMotorVehicle) {

((HeavyMotorVehicle) vehicles[i]).displayInfo();

}

}

scanner.close();

}

}

Q.30) 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 {

int rollno;

String name;

static int count = 0; // Static variable to keep track of the object count

// Default constructor

public Student() {

this.rollno = 0;

this.name = "Unknown";

count++; // Increment object count

System.out.println("Object created. Total objects: " + count);

}

// Parameterized constructor

public Student(int rollno, String name) {

this.rollno = rollno;

this.name = name;

count++; // Increment object count

System.out.println("Object created. Total objects: " + count);

}

// Method to display student details

public void displayStudentInfo() {

System.out.println("Roll No: " + rollno);

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

}

}

public class Main {

public static void main(String[] args) {

// Creating student objects using parameterized constructor

Student s1 = new Student(101, "Alice");

s1.displayStudentInfo(); // Display student 1 details

// Creating student objects using default constructor

Student s2 = new Student();

s2.displayStudentInfo(); // Display student 2 details

// Creating another student object using parameterized constructor

Student s3 = new Student(102, "Bob");

s3.displayStudentInfo(); // Display student 3 details

}

}