

OOPS EXPERIMENTS

SUBMITTEDBY:

NAME–AMIT PANWAR

SAP–500122506

BATCH–11 CSF

**Experiment 2 :- BASIC JAVA PROGRAMMING**

*1. Write a program to find area of triangle.*

*import java.util.Scanner;*

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

*System.out.print("Enter base: "); double base = sc.nextDouble();*

*System.out.print("Enter height: "); double height = sc.nextDouble();*

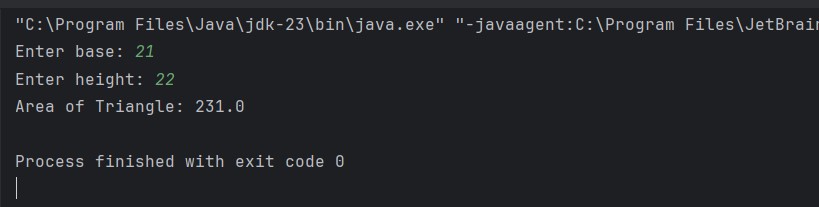
*double area = 0.5 \* base \* height;*

*System.out.println("Area of Triangle: " + area);*

*sc.close();*

*}*

*}*



*2. Write a program to find simple interest.*

*import java.util.Scanner;*

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

*System.out.print("Enter principal: "); double principal = sc.nextDouble();*

*System.out.print("Enter rate of interest: "); double rate = sc.nextDouble();*

*System.out.print("Enter time (in years): "); double time = sc.nextDouble();*

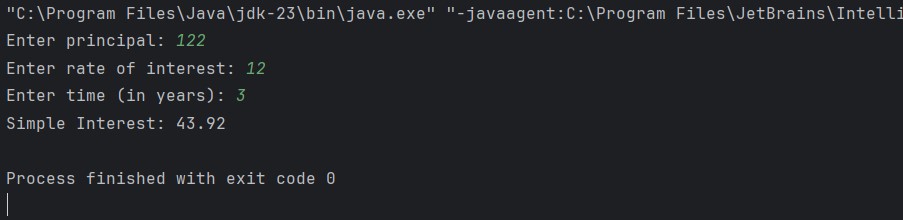
*double simpleInterest = (principal \* rate \* time) / 100;*

*System.out.println("Simple Interest: " + simpleInterest);*

*sc.close();*

*}*

*}*



*3. Write a program to implement a command line calculator. (Try for Add sub Mul in same program for 2 digits.)*

*public class CommandLineCalculator { public static void main(String[] args) { if (args.length != 3) {*

*System.out.println("Usage: java CommandLineCalculator <num1> <operator> <num2>"); return;*

*}*

*int num1 = Integer.parseInt(args[0]); int num2 = Integer.parseInt(args[2]);*

*String operator = args[1];*

*int result = 0; String operation = "";*

*switch (operator) { case "+":*

*result = num1 + num2; operation = "Sum";*

*break; case "-":*

*result = num1 - num2;*

*operation = "Difference";*

*break; case "\*":*

*result = num1 \* num2; operation = "Product";*

*break; default:*

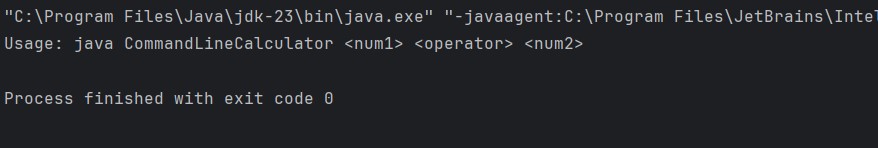
*System.out.println("Invalid operator. Use +, -, or \*"); return;*

*}*

*System.out.println(operation + " of " + num1 + " and " + num2 + " is " + result);*

*}*

*}*



*4. Write a Java program to check whether a given number is positive, negative, or zero using an if-else statement.*

*import java.util.Scanner;*

*public class NumberCheck {*

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

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

*int num = sc.nextInt();*

*if (num > 0) {*

*System.out.println("The number is positive.");*

*} else if (num < 0) {*

*System.out.println("The number is negative."); } else {*

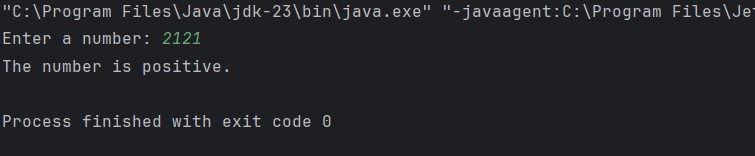
*System.out.println("The number is zero.");*

*}*

*sc.close();*

*}*

*}*



*5. Create a program that accepts three integers and determines the greatest among them using nested if-else statements.*

*import java.util.Scanner;*

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

*System.out.print("Enter first number: "); int num1 = sc.nextInt();*

*System.out.print("Enter second number: "); int num2 = sc.nextInt();*

*System.out.print("Enter third number: "); int num3 = sc.nextInt();*

*if (num1 >= num2) { if (num1 >= num3) {*

*System.out.println("The greatest number is: " + num1);*

*} else {*

*System.out.println("The greatest number is: " + num3);*

*}*

*} else {*

*if (num2 >= num3) {*

*System.out.println("The greatest number is: " + num2);*

*} else {*

*System.out.println("The greatest number is: " + num3);*

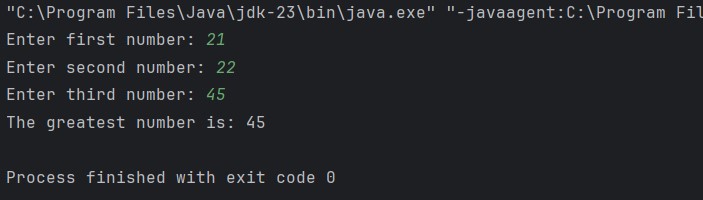
*}*

*}*

*sc.close();*

*}*

*}*



*6. Create a program that accepts a number (1–7) and displays the corresponding day of the week using a switch statement.*

*import java.util.Scanner;*

*public class DayOfWeek { public static void main(String[] args) { Scanner sc = new Scanner(System.in); System.out.print("Enter a number (1-7): "); int day = sc.nextInt();*

*String dayName;*

*switch (day) { case 1:*

*dayName = "Monday";*

*break; case 2:*

*dayName = "Tuesday";*

*break; case 3:*

*dayName = "Wednesday";*

*break; case 4:*

*dayName = "Thursday";*

*break; case 5: dayName = "Friday";*

*break; case 6:*

*dayName = "Saturday";*

*break; case 7:*

*dayName = "Sunday";*

*break; default:*

*dayName = "Invalid input";*

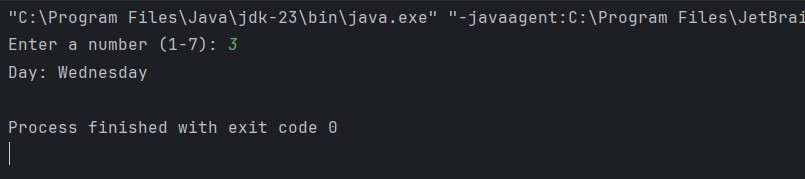
*}*

*System.out.println("Day: " + dayName);*

*sc.close();*

*}*

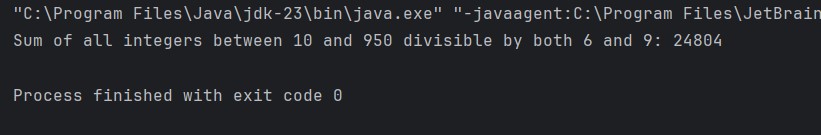
*}*



***Experiment 3:-*** *LOOPS AND ARRAYS*

1. Write a program to calculate the sum of all integers between 10 and 950 that are divisible by both 6 and 9.

|  |
| --- |
| 2. public class SumDivisibleBy6And9 { public static void main(String[] args) { int sum = 0;  for (int i = 10; i <= 950; i++) { if (i % 6 == 0 && i % 9 == 0) { sum += i;  }  }    System.*out*.println("Sum of all integers between 10 and 950 divisible by both 6 and 9: " + sum);  }  } |



3. Write a Java program that takes an integer as input and calculates the sum of its digits using a while loop.

import java.util.Scanner;

public class SumOfDigits {

public static void main(String[] args) {

Scanner sc = new Scanner(System.*in*);

System.*out*.print("Enter an integer: "); int num = sc.nextInt();

int sum = 0;

while (num != 0) { sum += num % 10; num /= 10;

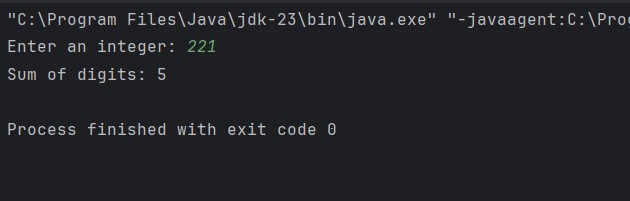
}

System.*out*.println("Sum of digits: " + sum);

sc.close();

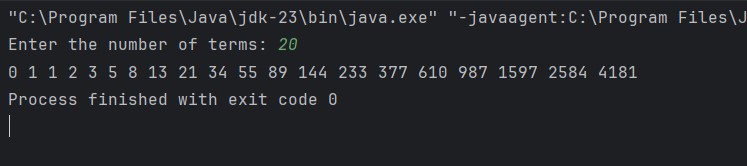
}

}



4. Write a Java program that prints the first N terms of the Fibonacci series using a loop

|  |
| --- |
| 5. import java.util.Scanner;  public class FibonacciSeries {  public static void main(String[] args) { Scanner sc = new Scanner(System.*in*);    System.*out*.print("Enter the number of terms: "); int n = sc.nextInt();    int first = 0, second = 1;  for (int i = 0; i < n; i++) { System.*out*.print(first + " "); int next = first + second; first = second; second = next;  }    sc.close();  }  } |



6. Write a Java program to count and display the total number of prime numbers between 1 and 1000.

public class PrimeNumbersCount { public static void main(String[] args) { int count = 0;

for (int num = 2; num <= 1000; num++) {

boolean isPrime = true;

for (int i = 2; i \* i <= num; i++) { if (num % i == 0) { isPrime = false; break;

}

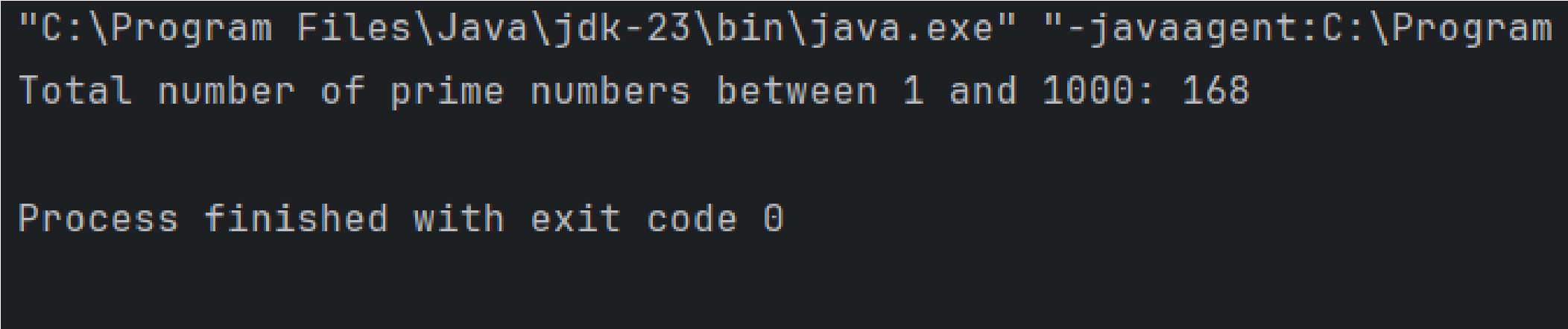
}

if (isPrime) { count++;

}

}

System.*out*.println("Total number of prime numbers between 1 and 1000: " + count);

 }

}

7. Write a Java program that counts how many times a given number appears in an array. Input: arr = [2, 3, 2, 5, 2, 6], target = 2 Output: 3

|  |
| --- |
| 8. import java.util.Scanner;  public class CountOccurrences {  public static void main(String[] args) { int[] arr = {2, 3, 2, 5, 2, 6};  Scanner sc = new Scanner(System.*in*);  System.*out*.print("Enter the target number: "); int target = sc.nextInt();  int count = 0;    for (int num : arr) { if (num == target) { count++;  }  }    System.*out*.println("The number " + target + " appears " + count + " times in the array."); |

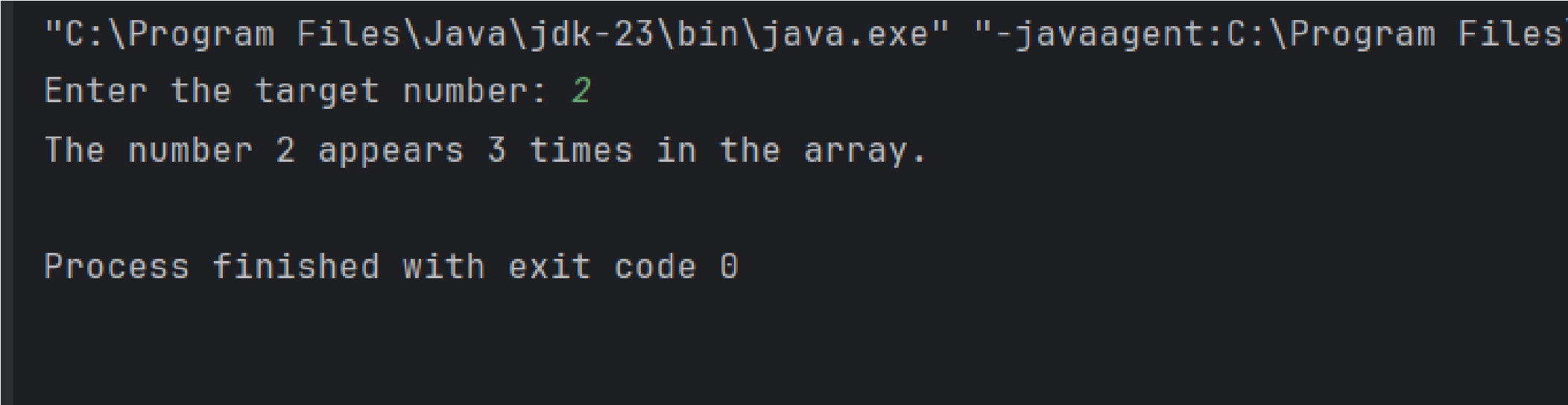
sc.close();

}

}

9. Write a Java program to find the second largest element in an integer array without sorting the array.

public class SecondLargestElement { public static void main(String[] args) { int[] arr = {12, 35, 1, 10, 34, 1};

 int largest = Integer.*MIN\_VALUE*; int secondLargest = Integer.*MIN\_VALUE*;

for (int num : arr) { if (num > largest) { secondLargest = largest; largest = num;

} else if (num > secondLargest && num != largest) { secondLargest = num;

}

}

if (secondLargest == Integer.*MIN\_VALUE*) {

System.*out*.println("No second largest element");

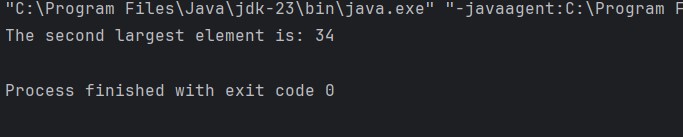
} else {

System.*out*.println("The second largest element is: " + secondLargest);

}

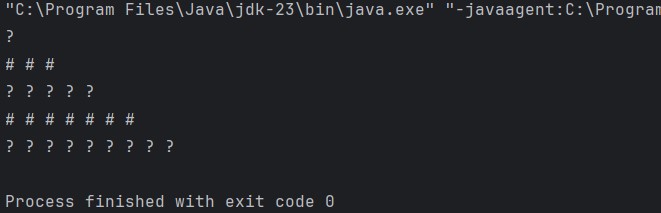
}

}



10. WAP to print the following pattern using loop ? # # # ? ? ? ? ? # # # # # # # ? ? ? ? ? ? ? ? ?

|  |
| --- |
| 11. public class PatternPrint { public static void main(String[] args) { int rows = 5;    for (int i = 1; i <= rows; i++) { if (i % 2 == 1) {  for (int j = 0; j < (2 \* i - 1); j++) {  System.*out*.print("? ");  }  } else {  for (int j = 0; j < (2 \* i - 1); j++) {  System.*out*.print("# ");  }  }  System.*out*.println();  }  }  } |



# Experiment 5:- INTERFACE

1. Write a Java program to demonstrate that a private member of a superclass cannot be accessed directly from a derived class.

class SuperClass {

private int privateVar = 10;

public int getPrivateVar() {

return privateVar;

}

}

class SubClass extends SuperClass { public void display() {

// System.out.println(privateVar); // This will cause an error because privateVar is private System.out.println("Private variable accessed via method: " + getPrivateVar());

}

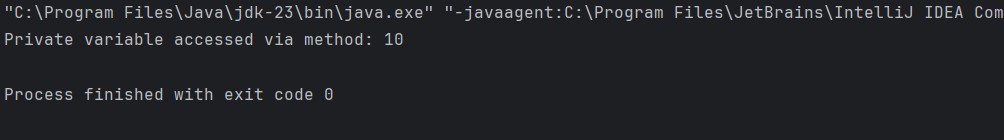
}

public class PrivateMemberDemo { public static void main(String[] args) { SubClass obj = new SubClass();

obj.display();

}

}



2. Create a Java program with a Player class and derive three subclasses: Cricket\_Player, Football\_Player, and Hockey\_Player. Implement attributes such as name, age, and position, and methods like play() and train() to represent these players.

class Player { String name; int age;

String position;

public Player(String name, int age, String position) { this.name = name; this.age = age;

this.position = position;

}

void play() {

System.out.println(name + " is playing.");

}

void train() {

System.out.println(name + " is training.");

}

}

class Cricket\_Player extends Player {

public Cricket\_Player(String name, int age, String position) { super(name, age, position);

}

void play() {

System.out.println(name + " is playing cricket as " + position);

}

}

class Football\_Player extends Player {

public Football\_Player(String name, int age, String position) { super(name, age, position);

}

void play() {

System.out.println(name + " is playing football as " + position);

}

}

class Hockey\_Player extends Player {

public Hockey\_Player(String name, int age, String position) { super(name, age, position);

}

void play() {

System.out.println(name + " is playing hockey as " + position);

}

}

public class PlayerDemo {

public static void main(String[] args) {

Cricket\_Player cricketer = new Cricket\_Player("Virat", 34, "Batsman");

Football\_Player footballer = new Football\_Player("Messi", 36, "Forward");

Hockey\_Player hockeyPlayer = new Hockey\_Player("Dhyan", 28, "Defender");

cricketer.play();

cricketer.train();

footballer.play();

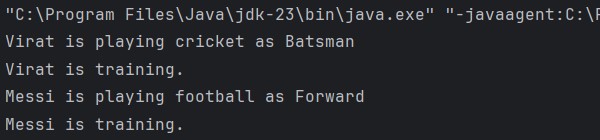
footballer.train();

hockeyPlayer.play();

hockeyPlayer.train();

}

}



3. Define a Worker class with DailyWorker and SalariedWorker as its subclasses. Each worker has a name and salary rate. Implement a method computePay(int hours) to compute weekly pay. DailyWorker is paid based on the number of days worked (assuming 8 hours per day), whereas SalariedWorker receives a fixed wage for 40 hours per week, regardless of actual hours worked. Use polymorphism to implement this program and test worker salary calculations.

class Worker { String name; double salaryRate; public Worker(String name, double salaryRate) { this.name = name;

this.salaryRate = salaryRate;

}

double computePay(int hours) {

return 0;

}

}

class DailyWorker extends Worker {

public DailyWorker(String name, double salaryRate) { super(name, salaryRate);

}

double computePay(int days) {

return days \* salaryRate \* 8;

}

}

class SalariedWorker extends Worker {

public SalariedWorker(String name, double salaryRate) { super(name, salaryRate);

}

double computePay(int hours) {

return salaryRate \* 40;

}

}

public class WorkerDemo {

public static void main(String[] args) {

DailyWorker dWorker = new DailyWorker("John", 50);

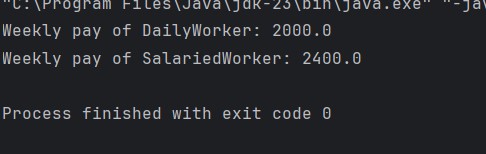
SalariedWorker sWorker = new SalariedWorker("Alice", 60);

System.out.println("Weekly pay of DailyWorker: " + dWorker.computePay(5));

System.out.println("Weekly pay of SalariedWorker: " + sWorker.computePay(40));

}

}



4. Implement a Java program to calculate trunk call charges based on duration and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. The program should take user input for duration and type and display the total charge.

import java.util.Scanner;

class TrunkCall {

double computeCharge(int minutes) { return 0;

}

}

class OrdinaryCall extends TrunkCall { double computeCharge(int minutes) {

return minutes \* 2.0;

}

}

class UrgentCall extends TrunkCall { double computeCharge(int minutes) {

return minutes \* 3.5;

}

}

class LightningCall extends TrunkCall { double computeCharge(int minutes) {

return minutes \* 5.0;

}

}

public class TrunkCallDemo {

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

System.out.print("Enter call duration in minutes: "); int minutes = scanner.nextInt();

System.out.print("Enter call type (1-Ordinary, 2-Urgent, 3-Lightning): "); int type = scanner.nextInt();

TrunkCall call;

switch (type) { case 1: call = new OrdinaryCall();

break; case 2: call = new UrgentCall();

break; case 3: call = new LightningCall();

break; default:

System.out.println("Invalid call type."); return;

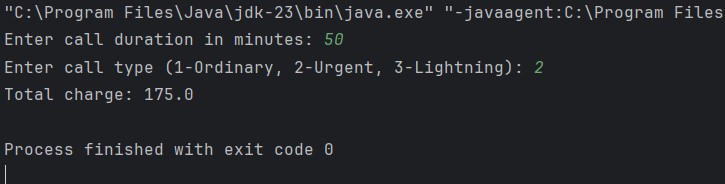
}

System.out.println("Total charge: " + call.computeCharge(minutes));

scanner.close();

}

}



5. Design a Java class Employee with attributes name, empId, and salary. Implement a default constructor, a parameterized constructor, and methods to return the employee’s name and salary. Add a method increaseSalary(double percentage) to raise the salary by a user-specified percentage. Create a Manager class with an additional instance variable department. Develop a test program to validate these functionalities.

import java.util.Scanner;

class Employee { String name; int empId;

double salary;

public Employee() { this.name = "Unknown";

this.empId = 0;

this.salary = 0.0;

}

public Employee(String name, int empId, double salary) { this.name = name; this.empId = empId;

this.salary = salary;

}

String getName() {

return name;

}

int getEmpId() { return empId; }

double getSalary() {

return salary;

}

void increaseSalary(double percentage) {

salary += (salary \* percentage / 100);

}

}

class Manager extends Employee {

String department;

public Manager(String name, int empId, double salary, String department) { super(name, empId, salary);

this.department = department;

}

String getDepartment() {

return department;

}

}

public class EmployeeDemo {

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

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

String name = scanner.nextLine();

System.out.print("Enter Employee ID: "); int empId = scanner.nextInt();

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

double salary = scanner.nextDouble();

scanner.nextLine();

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

String department = scanner.nextLine();

Manager manager = new Manager(name, empId, salary, department);

System.out.println("\nBefore Salary Increase:");

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

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

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

System.out.println("Department: " + manager.getDepartment());

System.out.print("\nEnter Percentage to Increase Salary: "); double percentage = scanner.nextDouble();

manager.increaseSalary(percentage);

System.out.println("\nAfter Salary Increase:"); System.out.println("Salary: " + manager.getSalary());

scanner.close();

}

}

