**SOFTWARE TESTING**

**(CSE 455)**

**ASSIGNMENT # 03**

|  |  |
| --- | --- |
| **NAME:** | MUAAZ BIN MUKHTAR |
| **REG NO:** | FA21-BSE-045 |
| **CLASS & SECTION:** | BSSE-7A |
| **SUBMITTED TO:** | Mam Najmun Nisa |
| **DATE SUBMITTED:** | 28-12-2024 |



**Department of Computer Science**

**Code:**

import java.util.ArrayList;

import java.util.Scanner;

class Employee {

private int id;

private String name;

private double basicSalary;

private double grossSalary;

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

this.id = id;

this.name = name;

this.basicSalary = basicSalary;

calculateGrossSalary();

}

private void calculateGrossSalary() {

double hra = 0.2 \* basicSalary; // HRA is 20% of basic salary

double da = 0.1 \* basicSalary; // DA is 10% of basic salary

double tax = 0.15 \* basicSalary; // Tax is 15% of basic salary

grossSalary = basicSalary + hra + da - tax;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public double getBasicSalary() {

return basicSalary;

}

public double getGrossSalary() {

return grossSalary;

}

@Override

public String toString() {

return String.format("Employee ID: %d, Name: %s, Basic Salary: %.2f, Gross Salary: %.2f",

id, name, basicSalary, grossSalary);

}

}

public class EmployeeManagementSystem {

private static ArrayList<Employee> employees = new ArrayList<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

boolean running = true;

while (running) {

System.out.println("\nEmployee Management System");

System.out.println("1. Add Employee");

System.out.println("2. Display All Employees");

System.out.println("3. Search Employee by ID");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

addEmployee(scanner);

break;

case 2:

displayAllEmployees();

break;

case 3:

searchEmployeeById(scanner);

break;

case 4:

running = false;

System.out.println("Exiting the program.");

break;

default:

System.out.println("Invalid choice! Please try again.");

}

}

scanner.close();

}

private static void addEmployee(Scanner scanner) {

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

int id = scanner.nextInt();

scanner.nextLine(); // Consume the newline

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

String name = scanner.nextLine();

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

double basicSalary = scanner.nextDouble();

Employee employee = new Employee(id, name, basicSalary);

employees.add(employee);

System.out.println("Employee added successfully.");

}

private static void displayAllEmployees() {

if (employees.isEmpty()) {

System.out.println("No employees to display.");

return;

}

System.out.println("\nEmployee Summary Report:");

for (Employee employee : employees) {

System.out.println(employee);

}

}

private static void searchEmployeeById(Scanner scanner) {

System.out.print("Enter Employee ID to search: ");

int id = scanner.nextInt();

for (Employee employee : employees) {

if (employee.getId() == id) {

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

System.out.println(employee);

return;

}

}

System.out.println("Employee not found with ID: " + id);

}

}

**A) Unit Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Input** | **Expected Output** |
| UT01 | Add a valid employee | ID: 101, Name: Muaaz, Basic Salary: 50000 | Employee added successfully, Gross Salary: 57500 |
| UT02 | Add a duplicate employee ID | ID: 101, Name: Ali, Basic Salary: 45000 | Error or overwriting not specified in the program (manual check) |
| UT03 | Search valid employee ID | ID: 101 | Employee found: Muaaz, Basic Salary: 50000 |
| UT04 | Search invalid employee ID | ID: 999 | Employee not found |
| UT05 | Display all employees | N/A | Lists all added employees |

**B) Conditional Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Condition** | **Input** | **Expected Output** |
| CT01 | Check empty list display | No employee added | "No employees to display." message |
| CT02 | Calculate Gross Salary (Basic > 0) | Basic Salary: 40000 | Gross Salary: 46000 |
| CT03 | Calculate Gross Salary (Basic = 0) | Basic Salary: 0 | Gross Salary: 0 |
| CT04 | Invalid menu choice | Menu: 9 | "Invalid choice! Please try again." |
| CT05 | Exit the system | Menu: 4 | Program exits |

**C) Decision Coverage Testing:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Decision Point** | **Input** | **Path Covered** | **Expected Output** |
| DC01 | Add employee (valid data) | ID: 102, Name: Muaaz, Salary: 30000 | Start → Add Employee → Valid Input → Success | Employee added successfully |
| DC02 | Search employee (exists) | ID: 102 | Start → Search Employee → Found | Employee details shown |
| DC03 | Search employee (does not exist) | ID: 999 | Start → Search Employee → Not Found | "Employee not found." message |
| DC04 | Exit program | Menu: 4 | Start → Exit | Program exits |
| DC05 | Invalid menu option | Menu: 7 | Start → Invalid Choice | "Invalid choice! Please try again." |

**D) Loop Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Input** | **Path Covered** | **Expected Output** |
| 1 | ID: 201, Name: Muaaz, Salary: 20000 | Start → Add Employee → Valid Input → Back to Menu | Employee added successfully |
| 2 | ID: 202, Name: Ali, Salary: 25000 | Start → Add Employee → Valid Input → Back to Menu | Employee added successfully |
| 3 | ID: 203, Name: Ahmed, Salary: 30000 | Start → Add Employee → Valid Input → Back to Menu | Employee added successfully |

**Paths for Each Iteration:**

**Path 1:** Start → Add Employee → Valid Input → Back to Menu

**Path 2:** Repeats Add Employee logic in a loop → same path as above

**Path 3:** Repeats Add Employee logic in a loop → same path as above, terminates on menu choice other than 1