Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: ESFP – II

Course Code: 2CSE203

Practical – 11

Name: Jaymin Gondaliya

Enrollment No: 23162171007

Sem - 2

Branch: CS

Class: B

Batch: 25

Objective:

To learn about object-oriented features, polymorphism (virtual and pure virtual function / abstract class) and inheritance concept.

Problem Definition-1: Complete the code for the object assigned to you to satisfy following specifications.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

const int MAX_EMPLOYEES = 100;

class swayam {};

class Employee {
public:
    string name;
    int id;
    string department;
    double salary;
```

```
    virtual void readData() {
        cout << "Enter employee name: ";
        cin >> name;
        cout << "Enter employee ID: ";
        cin >> id;
        cout << "Enter employee department: ";
        cin >> department;
        cout << "Enter employee salary: ";
        cin >> salary;
    }

    virtual void displayData() const {
        cout << "Name: " << name << endl;
        cout << "ID: " << id << endl;
        cout << "Department: " << department << endl;
        cout << "Salary: " << salary << endl;
    }

    virtual void deleteData() {}
    virtual void updateData() {}
    virtual void searchData(int id) const {}
    virtual void displayAllData(bool ascending) const {}
};

class EmployeeManager : public Employee {
private:
    Employee employees[MAX_EMPLOYEES];
    int numEmployees;

public:
    EmployeeManager() : numEmployees(0) {}

    void readData() override {
        if (numEmployees < MAX_EMPLOYEES) {
            Employee newEmployee;
            newEmployee.Employee::readData();
            employees[numEmployees] = newEmployee;
            numEmployees++;
        } else {
            cout << "Maximum number of employees reached." << endl;
        }
    }

    void displayData() const override {
        for (int i = 0; i < numEmployees; i++) {
            employees[i].displayData();
            cout << endl;
        }
```

```
    }

    void deleteData() override {
        int empId;
        cout << "Enter employee ID to delete: ";
        cin >> empId;

        for (int i = 0; i < numEmployees; i++) {
            if (employees[i].id == empId) {
                for (int j = i; j < numEmployees - 1; j++) {
                    employees[j] = employees[j + 1];
                }
                numEmployees--;
                cout << "Employee deleted successfully." << endl;
                return;
            }
        }

        cout << "Employee not found." << endl;
    }

    void updateData() override {
        int empId;
        cout << "Enter employee ID to update: ";
        cin >> empId;

        for (int i = 0; i < numEmployees; i++) {
            if (employees[i].id == empId) {
                Employee updatedEmployee;
                updatedEmployee.readData();
                employees[i] = updatedEmployee;
                cout << "Employee updated successfully." << endl;
                return;
            }
        }

        cout << "Employee not found." << endl;
    }

    void searchData(int searchId) const override {
        for (int i = 0; i < numEmployees; i++) {
            if (employees[i].id == searchId) {
                employees[i].displayData();
                return;
            }
        }

        cout << "Employee not found." << endl;
```

```
    }

    void displayAllData(bool ascending) const override {
        Employee sortedEmployees[MAX_EMPLOYEES];
        for (int i = 0; i < numEmployees; i++) {
            sortedEmployees[i] = employees[i];
        }

        for (int i = 0; i < numEmployees - 1; i++) {
            for (int j = 0; j < numEmployees - i - 1; j++) {
                if (ascending) {
                    if (sortedEmployees[j].id > sortedEmployees[j + 1].id) {
                        Employee temp = sortedEmployees[j];
                        sortedEmployees[j] = sortedEmployees[j + 1];
                        sortedEmployees[j + 1] = temp;
                    }
                } else {
                    if (sortedEmployees[j].id < sortedEmployees[j + 1].id) {
                        Employee temp = sortedEmployees[j];
                        sortedEmployees[j] = sortedEmployees[j + 1];
                        sortedEmployees[j + 1] = temp;
                    }
                }
            }
        }

        for (int i = 0; i < numEmployees; i++) {
            sortedEmployees[i].displayData();
            cout << endl;
        }
    }
};

int main() {
    EmployeeManager manager;

    int choice;
    do {
        cout << "\nEmployee Management System\n";
        cout << "1. Read Employee Data\n";
        cout << "2. Display Employee Data\n";
        cout << "3. Delete Employee Data\n";
        cout << "4. Update Employee Data\n";
        cout << "5. Search Employee Data\n";
        cout << "6. Display All Employees (Ascending)\n";
        cout << "7. Display All Employees (Descending)\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
```

```
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1:
                manager.readData();
                break;
            case 2:
                manager.displayData();
                break;
            case 3:
                manager.deleteData();
                break;
            case 4:
                manager.updateData();
                break;
            case 5:
                {
                    int employeeId;
                    cout << "Enter employee ID to search: ";
                    cin >> employeeId;
                    manager.searchData(employeeId);
                    break;
                }
            case 6:
                manager.displayAllData(true);
                break;
            case 7:
                manager.displayAllData(false);
                break;
            case 8:
                cout << "Exiting program..." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 8);

    return 0;
}
```

Output –

```
Enter your choice: 1
Enter employee name: manan
Enter employee ID: 1001
Enter employee department: CS
Enter employee salary: 20000

Employee Management System
1. Read Employee Data
2. Display Employee Data
3. Delete Employee Data
4. Update Employee Data
5. Search Employee Data
6. Display All Employees (Ascending)
7. Display All Employees (Descending)
8. Exit
Enter your choice: 1
Enter employee name: Khushi
Enter employee ID: 9007
Enter employee department: CS
Enter employee salary: 40000

Employee Management System
1. Read Employee Data
2. Display Employee Data
3. Delete Employee Data
4. Update Employee Data
5. Search Employee Data
6. Display All Employees (Ascending)
```

```
 Employee Management System
 1. Read Employee Data
 2. Display Employee Data
 3. Delete Employee Data
 4. Update Employee Data
 5. Search Employee Data
 6. Display All Employees (Ascending)
 7. Display All Employees (Descending)
 8. Exit
 Enter your choice: 2
 Name: Test
 ID: 1001
 Department: CS
 Salary: 30000

 Name: Khushi
 ID: 9007
 Department: CS
 Salary: 40000


 Employee Management System
 1. Read Employee Data
 2. Display Employee Data
 3. Delete Employee Data
```

```
7. Display All Employees (Descending)
8. Exit
Enter your choice: 6
Name: Test
ID: 1001
Department: CS
Salary: 30000

Name: Khushi
ID: 9007
Department: CS
Salary: 40000
```

```
Employee Management System
1. Read Employee Data
2. Display Employee Data
3. Delete Employee Data
4. Update Employee Data
5. Search Employee Data
6. Display All Employees (Ascending)
7. Display All Employees (Descending)
8. Exit
Enter your choice: 7
Name: Khushi
ID: 9007
Department: CS
Salary: 40000

Name: Test
ID: 1001
Department: CS
Salary: 30000
```