# Third Year B. Tech (EL & CE)

**Semester: V Subject:** Object-Oriented Programming Lab

**Name: Shreerang Mhatre Class: SY**

**Roll No: 52 Batch: A3**

# Experiment No: 02

**Name of the Experiment**: **Database of Airline Company**

**Performed on: 13/09/2023**

**Submitted on: 27/08/2023**

**Problem Statement:**

Develop an object oriented program in C++ to create a database of employee into system containing following info:

Employee name, Employee No, Qualification ,address ,contact , Salary (basic, DA, Ta , Net Salary)

Construct the Database with suitable inline member function for initializing and destroying the data via

Constructor, default Constructor, Copy Constructor , destructor.

Use dynamic memory allocation concept while creating and destroying the object of class.

Use static data member concept whenever needed display the Employee info.

**OUTPUT:**

Employee Name: Shreerang Mhatre

Employee Number: 302

Qualification: B.Tech Engineering

Address: MIT-WPU

Contact Number: +91 123456789

Basic Salary: 5000

DA: 1000

TA: 500

Net Salary: 6500

--------------------------------

Employee Name: Aman Singh

Employee Number: 401

Qualification: M.Tech Engineering

Address: BVP PUNE

Contact Number: +91 325689741

Basic Salary: 6000

DA: 1200

TA: 600

Net Salary: 7800

--------------------------------

Total Employees: 2

Code:

#include <iostream>

#include <string>

class Employee {

private:

static int empCounter;

std::string empName;

int empNumber;

std::string qualification;

std::string address;

std::string contactNumber;

struct Salary {

double basic;

double DA;

double TA;

double netSalary;

} salary;

public:

// Parameterized constructor

Employee(const std::string& name, int number, const std::string& qual, const std::string& addr,

const std::string& contact, double basicSalary, double DA, double TA) {

empName = name;

empNumber = number;

qualification = qual;

address = addr;

contactNumber = contact;

salary.basic = basicSalary;

salary.DA = DA;

salary.TA = TA;

salary.netSalary = calculateNetSalary();

empCounter++;

}

// Default constructor

Employee() : Employee("", 0, "", "", "", 0.0, 0.0, 0.0) {}

// Copy constructor

Employee(const Employee& other) {

empName = other.empName;

empNumber = other.empNumber;

qualification = other.qualification;

address = other.address;

contactNumber = other.contactNumber;

salary.basic = other.salary.basic;

salary.DA = other.salary.DA;

salary.TA = other.salary.TA;

salary.netSalary = other.salary.netSalary;

empCounter++;

}

// Destructor

~Employee() {

empCounter--;

}

// Calculate the net salary based on basic, DA, and TA

double calculateNetSalary() const {

return salary.basic + salary.DA + salary.TA;

}

// Display employee information

void displayInfo() const {

std::cout << "Employee Name: " << empName << std::endl;

std::cout << "Employee Number: " << empNumber << std::endl;

std::cout << "Qualification: " << qualification << std::endl;

std::cout << "Address: " << address << std::endl;

std::cout << "Contact Number: " << contactNumber << std::endl;

std::cout << "Basic Salary: " << salary.basic << std::endl;

std::cout << "DA: " << salary.DA << std::endl;

std::cout << "TA: " << salary.TA << std::endl;

std::cout << "Net Salary: " << salary.netSalary << std::endl;

std::cout << "--------------------------------" << std::endl;

}

// Static member function to get the total number of employees

static int getTotalEmployees() {

return empCounter;

}

};

// Initialize the static member

int Employee::empCounter = 0;

int main() {

// Create employee objects using dynamic memory allocation

Employee\* emp1 = new Employee("Shreerang Mhatre", 302, "B.Tech Engineering", "MIT-WPU", "+91 123456789 ", 5000.0, 1000.0, 500.0);

Employee\* emp2 = new Employee("Aman Singh", 401, "M.Tech Engineering", "BVP PUNE", "+91 325689741", 6000.0, 1200.0, 600.0);

// Display employee information

emp1->displayInfo();

emp2->displayInfo();

// Get the total number of employees

std::cout << "Total Employees: " << Employee::getTotalEmployees() << std::endl;

// Clean up memory and release resources

delete emp1;

delete emp2;

return 0;

}







