JNCT

LNCT Group of Colleges

Object oriented programming methodology case study

EMPLOYEE CLASS HIERARCHY

TEAM NAME - CODEBUZZERS

MENTOR NAME –Prof. Vivek Dubey

Team members 3rd sem

Leader name - Astha jain(0131AD221018)

Devaki Gupta(0131AD221023)

Mahak Kushwaha(0131AD221036)

Khushboo rai(0131AD221035)

Department Of Artificial Intelligence And Data Science

OOPM CASE STUDY

EMPLOYEE CLASS HIERARCHY

Need of case study

OOPM CASE STUDY

EMPLOYEE CLASS HIERARCHY

Advantages $ Disadvantages of case study

- Advantages
- Clear line of authority
- Clear lines of communication
- Clear results

- Disadvantages
- Isolation and siloed thinking
- Centralization of power
- Endless red tape

OOPM CASE STUDY

EMPLOYEE CLASS HIERARCHY

RELATED STUDY

- Some online platforms specialize in hosting and sharing case studies across different industries. These platforms may offer free or paid access to a variety of case studies. while some case studies are freely available, others may require purchase or access through a subscription.

- Yes,google is a powerful search engine that indexes a vast amount of information available on the internet, including case studies from various sources. The availability of case studies on Google may vary, and some case studies may be accessible directly, while others may require access through specific websites or databases.

- The term "case study" can refer to a variety of materials, and whether a case study is ready to use depends on its source and purpose. However, there are instances where case studies may not be immediately ready to use.

- Using a recursive CTE to traverse an employee hierarchy by YugabyteDB Docs: This article describes a case study of how to use a recursive Common Table Expression (CTE) to traverse an employee hierarchy and retrieve information about all of the employees in the hierarchy.
- The link to the article "using a recursive CTE to traverse an employee hierarchy" by YugabyteDB Docs is:
- https://docs.yugabyte.com/preview/api/ysql/the-sql-language/with-clause/emps-hierarchy/

```
#include <iostream>
#include <vector>
#include <limits> // for numeric_limits
#include <algorithm>

class Employee {
public:
    Employee(int id, std::string name) : id(id),
name(name) {}

    virtual void display() const {
        std::cout << "ID: " << id << ", Name: " << name;
    }

    int getId() const {
        return id;
    }
void setName(const std::string& newName) {
        name = newName;
    }

private:
    int id;
```

```
 std::string name;
};

class Manager : public Employee {
public:
    Manager(int id, std::string name,
std::string department)
        : Employee(id, name),
department(department) {}

    void display() const override {
        Employee::display();
        std::cout << ", Department: " <<
department;
    }

private:
    std::string department;
};

class Engineer : public Employee {
public:
    Engineer(int id, std::string name,
std::string specialty)
```

```
 : Employee(id, name),
specialty(specialty) {}

    void display() const override {
        Employee::display();
        std::cout << ", Specialty: " <<
specialty;
    }

private:
    std::string specialty;
};

class EmployeeDatabase {
public:
    // Insertion
    void addEmployee(Employee* emp) {
        employees.push_back(emp);
    }
// Deletion
    void removeEmployee(int id) {
        auto it =
```

```
std::remove_if(employees.begin(),
employees.end(),
                               [id](const Employee*
emp) { return emp->getId() == id; });

        employees.erase(it,
employees.end());
    }

    // Searching
    Employee* findEmployee(int id) const {
        auto it =
std::find_if(employees.begin(),
employees.end(),
                               [id](const Employee*
emp) { return emp->getId() == id; });

        if (it != employees.end()) {
            return *it;
        } else {
            return nullptr; // Not found
        }
```

```
}

    // Updation
    void updateEmployee(int id, const
std::string& newName) {
        Employee* emp = findEmployee(id);

        if (emp != nullptr) {
            emp->setName(newName);
        } else {
            std::cout << "Employee with ID "
<< id << " not found." << std::endl;
        }
    }

    // Display all employees
    void displayAllEmployees() const {
        for (const auto& emp : employees) {
            emp->display();
 std::cout << std::endl;
        }
    }
```

```
private:
    std::vector<Employee*> employees;
};

int main() {
    EmployeeDatabase database;

    while (true) {
        std::cout << "\nMenu:\n"
                  "1. Add Employee\n"
                  "2. Display All Employees\n"
                  "3. Search Employee\n"
                  "4. Update Employee\n"
                  "5. Remove Employee\n"
                  "6. Exit\n"
                  "Enter your choice: ";
 int choice;
        std::cin >> choice;

        switch (choice) {
            case 1: {
                std::cout << "Enter employee
details:\n";
```

```
 int id;
                std::string name;
                std::cout << "ID: ";
                std::cin >> id;
                std::cout << "Name: ";

std::cin.ignore(std::numeric_limits<std::st
reamsize>::max(), '\n');  // Clear the
input buffer
                std::getline(std::cin, name);

                // Add employee based on type
(Manager or Engineer)
                std::cout << "Enter employee
type (1 for Manager, 2 for Engineer): ";
                int type;
                std::cin >> type;
if (type == 1) {
                   std::string department;
                   std::cout << "Department: ";
```

```
std::cin.ignore(std::numeric_limits<std::st
reamsize>::max(), '\n');  // Clear the
input buffer
                std::getline(std::cin,
department);

                Manager* manager = new
Manager(id, name, department);

database.addEmployee(manager);
            } else if (type == 2) {
                std::string specialty;
                std::cout << "Specialty: ";

std::cin.ignore(std::numeric_limits<std::st
reamsize>::max(), '\n');  // Clear the
input buffer
                std::getline(std::cin,
specialty);

                Engineer* engineer = new
Engineer(id, name, specialty);

database.addEmployee(engineer);
```

```
 } else {
                std::cout << "Invalid
employee type.\n";
            }

            break;
        }
        case 2:
            std::cout << "All Employees:\n";

database.displayAllEmployees();
            break;
        case 3: {
            std::cout << "Enter employee ID
to search: ";
            int searchId;
            std::cin >> searchId;

            Employee* foundEmployee =
database.findEmployee(searchId);
```

```
            if (foundEmployee != nullptr) {
                std::cout << "Employee
found: ";
                foundEmployee->display();
                std::cout << std::endl;
            } else {
                std::cout << "Employee with
ID " << searchId << " not found." <<
std::endl;
            }

            break;
        }
        case 4: {
            std::cout << "Enter employee ID
to update: ";
            int updateId;
 std::cin >> updateId;

            std::string newName;
            std::cout << "Enter new name:
";
```

```
std::cin.ignore(std::numeric_limits<std::st
reamsize>::max(), '\n');  // Clear the
input buffer
            std::getline(std::cin,
newName);


database.updateEmployee(updateId,
newName);
            break;
        }
        case 5: {
            std::cout << "Enter employee ID
to remove: ";
            int deleteId;
            std::cin >> deleteId;

database.removeEmployee(deleteId);
            break;
        }
```

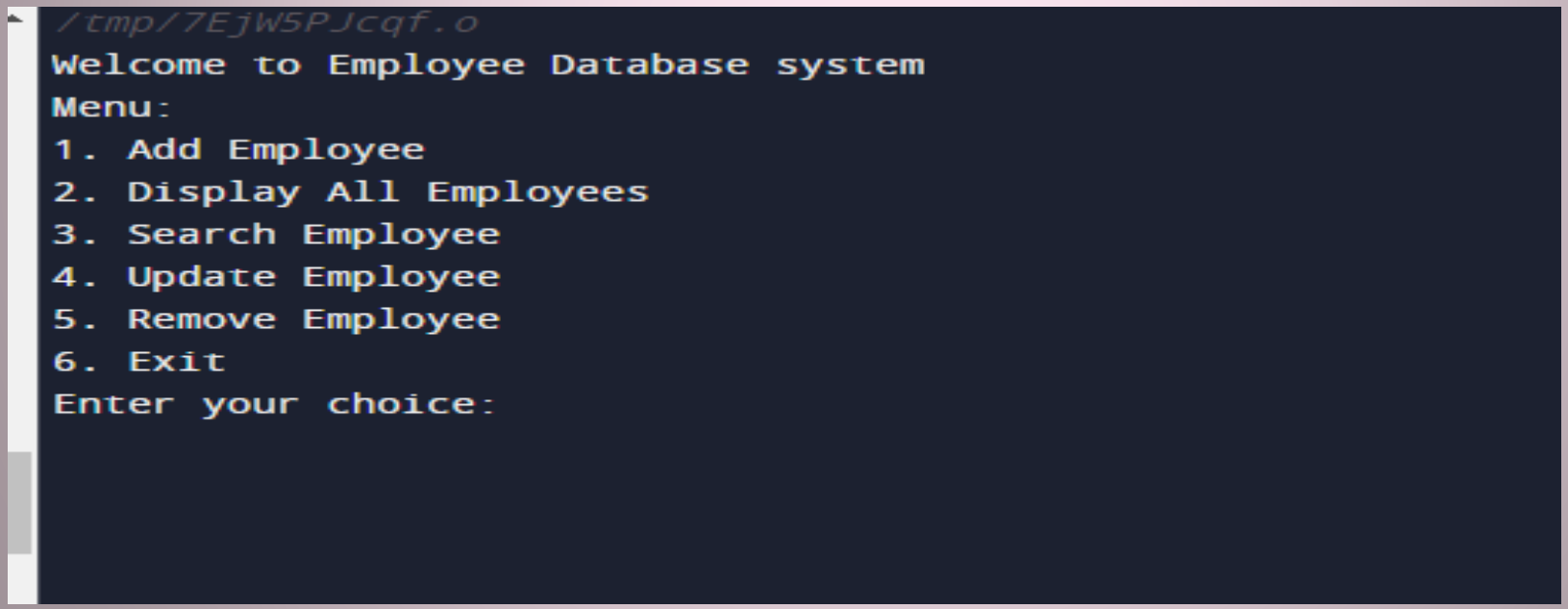
```
 case 6:
                std::cout << "Exiting the
program.\n";
                return 0;
            default:
                std::cout << "Invalid choice.
Please try again.\n";
        }
    }

    return 0;
}
```

INPUT & OUTPUT FLOW OF CASE STUDY

- Front page

Our screen will display like as:
There will be 6 options and there will be option to enter choice.



INPUT & OUTPUT FLOW OF CASE STUDY

EMPLOYEE DATABASE SYSTEM

Enter code:1 for adding employee

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 1
Enter employee details:
ID: 113121Em
Name: Manoj kumar
Enter employee type (1 for Manager, 2 for Engineer): 1
Department: sales
```

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 1
Enter employee details:
ID: 1243Em
Name: jaya joseph
Enter employee type (1 for Manager, 2 for Engineer): 2
Specialty: data engineer
```

INPUT & OUTPUT FLOW OF CASE STUDY

EMPLOYEE DATABASE SYSTEM

Enter code: 2 for display of all employees

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 2
All Employees:
ID: 1131, Name: Manoj kumar, Department: sales
ID: 131, Name: jaya joseph, Specialty: data engineer
```

INPUT & OUTPUT FLOW OF CASE STUDY

EMPLOYEE DATABASE SYSTEM

Enter code : 3 for search an Employee

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 3
Enter employee ID to search: 1131em
Employee found: ID: 1131, Name: Manoj kumar, Department: sales
```

INPUT & OUTPUT FLOW OF CASE STUDY

EMPLOYEE DATABASE SYSTEM

Enter code:4 update Employee

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 1
Enter employee details:
ID: 1131em
Name: manoj kumar
Enter employee type (1 for Manager, 2 for Engineer): 1
Department: sales
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 4
Enter employee ID to update: 1131em
Enter new name: Ravindra J
```

INPUT & OUTPUT FLOW OF CASE STUDY

EMPLOYEE DATABASE SYSTEM

Enter code: 5 for removing an employee

```
Menu:
1. Add Employee
2. Display All Employees
3. Search Employee
4. Update Employee
5. Remove Employee
6. Exit
Enter your choice: 5
Enter employee ID to remove: 1131em
Menu:
```

OOPM CASE STUDY

EMPLOYEE CLASS HIERARCHY

Code BuZzers

Team members 3rd sem

Leader name – Astha jain (0131AD221018)
Member 1 – Devaki Gupta (0131AD221023)
Member 2-Mahak khushwaha(0131AD221036)
Member 3 - Khushboo Rai (0131AD221035)