Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут

імені ІГОРЯ СІКОРСЬКОГО»

**ДОМАШНЯ КОНТРОЛЬНА РОБОТА**

з дисципліни «Технології об'єктно-орієнтованого програмування для радіоінженерів»

Виконав: Мосіюк Євгеній

студент РТФ

групи РE-12

Київ – 2023

**Зміст:** Розробка програмного забезпечення з використання об’єктно-орієнтованого програмування на мові С++.Закріпити отримані знання і здобути практичний досвід розробки програмного забезпечення з використанням технології об’єктно-орієнтованого програмування (ООП) шляхом написання програмного коду згідно поставленого завдання.

**Умова задачі:** Створити програмне забезпечення (ПЗ) з використанням ООП на мові С++ відповідно до варіанту задачі з можливістю логування важливих моментів роботи програми. Програма має містити мінімум два класи написаних власне студентом для реалізації поставленої задачі. Створити класи, специфікації яких наведені нижче. Визначити конструктори і методи set.. (), get.. (), toString (). Задати критерій вибору даних і вивести ці дані на консоль. У кожному класі, що володіє інформацією, має бути оголошено кілька конструкторів. Необхідно перевантажити оператор ‘«’ для виводу даних на консоль. Має бути можливість записати дані всіх об’єктів у файл з шифруванням (студент самостійно обирає спосіб шифрування). Додатково має бути написаний клас для логування важливих моментів роботи програми з виводом у файл. Для виконання роботи був обраний варіант № 4 - Car

Car: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер. Створити масив об'єктів. Вивести: a) список автомобілів заданої марки; b) список автомобілів заданої моделі, які експлуатуються більше n років; c) список автомобілів заданого року випуску, ціна яких більше зазначеної.

**Опис розробленого програмного забезпечення:**

Розроблене програмне забезпечення є системою університету і має наступні класи:

Student - представляє студента з такими властивостями як ідентифікатор (id), прізвище (lastName), ім'я (firstName), дата народження (dateOfBirth), телефон (phone), факультет (faculty), курс (course) та група (group). Клас має конструктор, методи set..() для встановлення значень властивостей, методи get..() для отримання значень властивостей та метод toString() для отримання рядкового представлення об'єкта студента.

University - представляє університет і має масив студентів. Методи класу включають:

addStudent() - для додавання студента до університету;

getStudentsByFaculty() - для отримання списку студентів заданого факультету;

getStudentsByFacultyAndCourse() - для отримання списку студентів для кожного факультету і курсу;

getStudentsBornAfterYear() - для отримання списку студентів, які народилися після заданого року;

getStudentsByGroup() - для отримання списку студентів навчальної групи;

writeStudentsToFile() - для запису даних всіх студентів у файл з шифруванням XOR.

Logger - клас для логування важливих моментів роботи програми. Має метод log(), який додає повідомлення до файлу журналу.

У програмному коді головна функція main демонструє використання розроблених класів. Спочатку створюються об'єкти студентів, додаються до об'єкту університету, а потім викликаються різні методи університету для отримання списків студентів за різними критеріями. Результати виводяться на консоль. Також використовується об'єкт класу Logger для логування початку та закінчення програми.

Код також забезпечує можливість запису даних всіх студентів у файл з шифруванням XOR. Для цього використовується метод writeStudentsToFile() класу University, який виконує наступні дії:

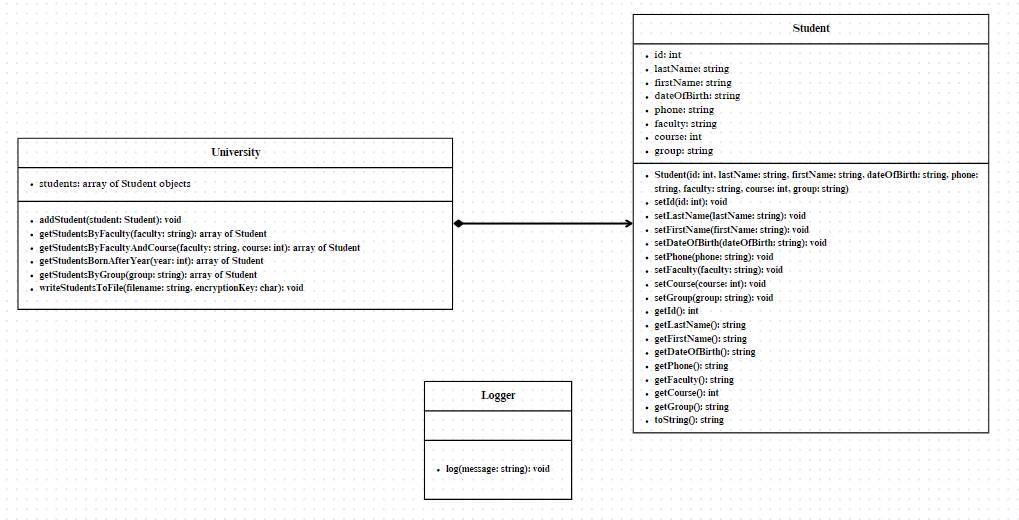
Відкриває файл для запису у бінарному режимі.

Проходиться по всіх студентах університету та отримує рядкове представлення кожного студента за допомогою методу toString().

Застосовує шифрування XOR до кожного символу рядка з використанням заданого ключа.

Записує зашифровані дані студента у файл, розділяючи їх новим рядком.

Таким чином, розроблене програмне забезпечення забезпечує можливість управління студентами університету, отримання списків студентів за різними критеріями, запис даних у файл з шифруванням та логування важливих моментів роботи програми.

**UML diagram**: 

**Програмний код:**

#include "student.h"

#include "university.h"

#include "logger.h"

#include <iostream>

int main() {

// Creating students

Student student1(1, "Smith", "John", "1995-07-15", "123456789", "Faculty of Science", 2, "Group A");

Student student2(2, "Johnson", "Emily", "1997-02-28", "987654321", "Faculty of Arts", 1, "Group B");

Student student3(3, "Brown", "Michael", "1996-10-10", "456789123", "Faculty of Engineering", 3, "Group C");

Student student4(4, "Davis", "Emma", "1998-05-03", "321654987", "Faculty of Science", 2, "Group A");

// Creating university

University university;

university.addStudent(student1);

university.addStudent(student2);

university.addStudent(student3);

university.addStudent(student4);

// Logging

Logger logger;

logger.log("Program started.");

// Getting students by faculty

std::string faculty = "Faculty of Science";

std::vector<Student> scienceStudents = university.getStudentsByFaculty(faculty);

std::cout << "Students in the " << faculty << ":" << std::endl;

for (const Student& student : scienceStudents) {

std::cout << student.toString() << std::endl;

}

// Getting students by faculty and course

std::string faculty2 = "Faculty of Science";

int course = 2;

std::vector<Student> scienceStudentsCourse2 = university.getStudentsByFacultyAndCourse(faculty2, course);

std::cout << "Students in the " << faculty2 << ", Course " << course << ":" << std::endl;

for (const Student& student : scienceStudentsCourse2) {

std::cout << student.toString() << std::endl;

}

// Getting students born after a year

int year = 1996;

std::vector<Student> studentsBornAfterYear = university.getStudentsBornAfterYear(year);

std::cout << "Students born after " << year << ":" << std::endl;

for (const Student& student : studentsBornAfterYear) {

std::cout << student.toString() << std::endl;

}

// Getting students by group

std::string group = "Group C";

std::vector<Student> groupCStudents = university.getStudentsByGroup(group);

std::cout << "Students in Group " << group << ":" << std::endl;

for (const Student& student : groupCStudents) {

std::cout << student.toString() << std::endl;

}

// Writing students to file

std::string filename = "students.txt";

char encryptionKey = 'X';

university.writeStudentsToFile(filename, encryptionKey);

logger.log("Program ended.");

return 0;

}

**Посилання на репозиторій GitHub:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Висновок:** У розробленому програмному забезпеченні було створено систему для управління студентами університету. Клас "Student" представляє студента з властивостями, такими як ідентифікатор, прізвище, ім'я, дата народження, телефон, факультет, курс і група. Клас "University" представляє університет і має методи для додавання студентів, отримання списків студентів за різними критеріями та запису даних у файл з шифруванням.

Для забезпечення важливих моментів роботи програми було додано клас "Logger", який використовується для логування. Клас має метод для запису повідомлень у файл журналу.

У розробленому програмному коді було використано об'єктно-орієнтований підхід, де кожен клас має свої властивості і методи для роботи з цими властивостями. Також було застосовано принципи інкапсуляції, де властивості класів були приховані і доступ до них здійснювався через публічні методи.

Зв'язки між класами були відображені у вигляді асоціацій, агрегації та залежності. Клас "University" має агрегаційний зв'язок з класом "Student", оскільки університет містить декілька студентів. Клас "University" також має залежність від класу "Logger", оскільки використовує його для логування.

Загальний висновок полягає в тому, що розроблене програмне забезпечення забезпечує управління студентами університету за допомогою об'єктно-орієнтованого підходу і використовується для отримання списків студентів за різними критеріями, запису даних у файл та логування важливих моментів роботи програми.

**Джерела**

1. Stroustrup, B. (2013). The C++ Programming Language (4th ed.). Addison-Wesley. - Основний підручник для вивчення мови C++, який був використаний для вивчення основних концепцій та синтаксису.
2. Cplusplus.com. (n.d.). C++ Reference. Retrieved from http://www.cplusplus.com/reference/ - Основний онлайн ресурс для пошуку інформації про різні функції та класи стандартної бібліотеки C++.
3. Stack Overflow. (n.d.). Retrieved from https://stackoverflow.com/ - Важливий ресурс для вирішення конкретних проблем під час розробки.