МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу кафедра інженерії програмного забезпечення

**Практична робота (Practice-1)**

з дисципліни «Об’єктно-орієнтоване програмування»

*назва дисципліни*

на тему: «Віртуальне середовище для студентів»

Виконав: студент 2 курсу групи 623П напряму підготовки (спеціальності)

121 інженерія програмного забезпечення

(шифр і назва напряму підготовки (спеціальності))

Марченко Д.О.

(прізвище й ініціали студента)

Прийняв: доц. каф. 603, к.т.н., доцент

Шевченко І.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: Кількість балів: Оцінка ECTS:

Харків – 2024

# Завдання

1. Обрати і коротко описати нову предметну область.
2. Для обраної предметної області з використанням будь-якого спеціалізованого онлайн-сервісу (наприклад, <https://www.lucidchart.com>) за два етапи побудувати модель предметної області у вигляді діаграми класів, на якій класи/інтерфейси мають бути пов’язані різними типами відношень: асоціація, агрегація, композиція, реалізація.
3. Засобами середи розробки (IDE) створити каркас проєкту, для цього необхідно визначити в окремих файлах \*.cs всі спроєктовані у п.2 інтерфейси, класи, перелічення. Коректно реалізувати відношення між класами! Увага! Реалізовувати властивості, конструктори і методи у класах не потрібно! Замість реального коду вони мають містити заглушки throw new NotImplementedException().
4. Для тестування спроєктованих класів додати проєкт з тест-класами, які повинні містити достатній для повноцінного тестування набір реалізованих unit-тестів. Увага! Методи класів предметної області коду ще не мають, але для них вже пишемо повноцінні unit-тести (TDD-принцип розробки програми). Після запуску розроблених unit-тестів їх статус має бути failed.

# Опис предметної області

1. User – користувач (базовий клас); має такі властивості name (ім’я користувача), id (унікальний ідентифікатор), username (унікальне ім’я користувача), password (пароль користувача), chats (список чатів користувача). Може авторизуватись (містить метод signIn), та додавати чати до свого списку (addChat). Зв’язаний з всіма спеціалізованими сутностями (дочірними класами) та з Чатом.
2. Student – студент. Є спеціалізованою сутністю User (походить від класу User); має такі властивості: courses (список предметів які вивчає), year (курс навчання), faculty (назва факультету). Зв’язаний з сутністю Курс, Чат, Повідомлення.
3. Teacher – викладач. Є спеціалізованою сутністю User (походить від класу User); має такі властивості: courses (список предметів які викладає). Зв’язаний з сутністю Студент і Курс.
4. Chat – чат для комунікації. Має такі властивості: messages (список повідомлень в чаті). Може використовуватись для відправки повідомлення (sendMessage), видалення повідомлення (deleteMessage), отримання списку повідомлень (getMessages). Зв’язаний з сутністю Користувач і Повідомлення.
5. ChatMessage – повідомлення, складова частина сутності Chat (відношення частина - ціле). Має такі властивості: sender (відправник повідомлення), timestamp (дата відправки повідомлення), message (текст повідомлення.
6. Course – курс (предмет). Має такі властивості: name (назва курсу), task (завдання курсу). Може бути використаний для отримання завданя (getTask), здачі завдання (submitTask), створення завдання (createTask). Зв’язаний з сутностями Студент і Викладач.

# Проєктування діаграми класів

Зображення, що містить схема, Креслення, План, ескіз

Автоматично згенерований опис

Рисунок 1 Виявлення та визначення елементів предметної області та зв’язки між ними

Зображення, що містить текст, схема, План, Креслення

Автоматично згенерований опис

Рисунок 2 Детальне проєктування елементів моделі предметної області

# Реалізація класів

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

public class Teacher : User

{

public List<Course> Courses;

public Teacher(string name, string username, string password) : base(name, username, password)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

interface IUser

{

string Name { get; set; }

string Username { get; set; }

string Id { get; set; }

List<SupportChat> Chats { get; set; }

bool SignIn(string username, string password);

void AddChat(SupportChat chat);

}

public class User : IUser

{

public string Name { get; set; }

public string Username { get; set; }

private string Password { get; set; }

public string Id { get; set; }

public List<SupportChat> Chats { get; set; }

public User(string name, string username, string password)

{

Name = name;

Username = username;

Password = password;

}

public bool SignIn(string username, string password)

{

throw new NotImplementedException();

}

public void AddChat(SupportChat chat)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

public class Student : User

{

public int Year;

public string Faculty;

public List<Course> Courses;

public Student(string name, string username, string password) : base(name, username, password)

{

throw new NotImplementedException();

}

public SupportChat ContactSupport()

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

public class Course

{

string Name;

string Task;

public Course(string name)

{

throw new NotImplementedException();

}

public void CreateTask(string taks)

{

// Code to add a student to the group

throw new NotImplementedException();

}

public void SubmitTask(string task, Student student)

{

// Code to remove a student from the group

throw new NotImplementedException();

}

public string GetTask()

{

// Code to remove a student from the group

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

public class Chat

{

private List<ChatMessage> Messages;

public Chat()

{

throw new NotImplementedException();

}

public void SendMessage(User sender, string message)

{

throw new NotImplementedException();

}

public void DeleteMessage(ChatMessage message)

{

throw new NotImplementedException();

}

public List<ChatMessage> GetMessages()

{

throw new NotImplementedException();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace project\_console

{

public class ChatMessage

{

public string SenderId;

public string Message;

public DateTime Timestamp;

public ChatMessage(string senderId, string message)

{

throw new NotImplementedException();

}

}

}

# Реалізація тест-класів

using project\_console;

namespace unit\_test

{

[TestClass]

public class UserTests

{

[TestMethod]

public void TeacherGetsCreated()

{

new Teacher("John Doe", "johndoe", "password");

}

[TestMethod]

public void StudentGetsCreated()

{

new Student("John Doe", "johndoe", "password");

}

[TestMethod]

public void UserCanSignInWithRightCredentials()

{

Student student = new Student("Student Name", "username1", "password");

Assert.AreEqual(true, student.SignIn("username1", "password"));

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Assert.AreEqual(true, teacher.SignIn("username2", "password"));

}

[TestMethod]

public void UserCantSignInWithWrongCredentials()

{

Student student = new Student("Student Name", "username1", "password");

Assert.AreEqual(false, student.SignIn("username1", "password123"));

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Assert.AreEqual(false, teacher.SignIn("username3", "password"));

}

[TestMethod]

public void UserCanAddChat()

{

Student student = new Student("Student Name", "username1", "password");

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Chat chat = new Chat();

student.AddChat(chat);

teacher.AddChat(chat);

}

}

}

using project\_console;

namespace unit\_test

{

[TestClass]

public class CourseTests

{

[TestMethod]

public void GroupGetsCreated()

{

new Course("course1");

}

[TestMethod]

public void CanCreateTask()

{

Course course = new Course("course1");

course.CreateTask("task1");

}

[TestMethod]

public void CanSubmitTask()

{

Course course = new Course("course1");

Student student = new Student("Student Name", "username1", "password");

course.SubmitTask("task1", student);

}

[TestMethod]

public void CanGetTask()

{

Course course = new Course("course1");

course.CreateTask("task1");

string task = course.GetTask();

Assert.AreEqual("task1", task);

}

}

}

using project\_console;

namespace unit\_test

{

[TestClass]

public class ChatTests

{

[TestMethod]

public void ChatGetsCreated()

{

new Chat();

}

[TestMethod]

public void CanSendMessage()

{

Student student = new Student("Student Name", "username1", "password");

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Chat chat = new Chat();

chat.SendMessage(student, "Hello");

chat.SendMessage(teacher, "Hi");

}

[TestMethod]

public void GetMessages()

{

Student student = new Student("Student Name", "username1", "password");

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Chat chat = new Chat();

chat.SendMessage(student, "Hello");

chat.SendMessage(teacher, "Hi");

List<ChatMessage> messages = chat.GetMessages();

Assert.AreEqual(2, messages.Count);

}

[TestMethod]

public void DeleteMessages()

{

Student student = new Student("Student Name", "username1", "password");

Teacher teacher = new Teacher("Teacher Name", "username2", "password");

Chat chat = new Chat();

chat.SendMessage(student, "Hello");

chat.SendMessage(teacher, "Hi");

List<ChatMessage> messages1 = chat.GetMessages();

chat.DeleteMessage(messages1[0]);

chat.DeleteMessage(messages1[1]);

List<ChatMessage> messages2 = chat.GetMessages();

Assert.AreEqual(0, messages2.Count);

}

}

}

# Результати запуску unit-тестів

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

# Git репозиторій

https://github.com/marchenkodima/oop-project-console/tree/dev