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). Може авторизуватись (містить метод signIn). Зв’язаний з всіма спеціалізованими сутностями (дочірними класами).
2. Student – студент, складова сутності Group (відношення частина – ціле). Є спеціалізованою сутністю User (походить від класу User); має такі властивості: id (ідентифікатор студента), groupID (ідентифікатор групи), chatIDs (перелік ідентифікаторів доступних чатів). Може попросити про допомогу (метод contactSupport). Зв’язаний з сутністю Адміністратор, Чат.
3. Administrator – адміністратор. Є спеціалізованою сутністю User (походить від класу User); має такі властивості: chatIDs (перелік ідентифікаторів доступних чатів). Може додавати до системи новго користувача (addStudent), видалити користувача (deleteStudent), отримати список студентів (getStudents), отримати список адміністраторів (getAdministrators), додати до системи нову групу (addGroup). Зв’язаний з сутністю Студент і Чат.
4. SupportChat – чат для комунікації. Має такі властивості: id (ідентифікатор чату), studentID (ідентифікатор студента), administratorID (ідентифікатор адміністратора). Може використовуватись для відправки повідомлення (sendMessage), видалення повідомлення (deleteMessage), отримання списку повідомлень (getMessages). Зв’язаний з сутністю Студент, Адміністратор і Повідомлення.
5. ChatMessage – повідомлення, складова частина сутності SupportChat (відношення частина - ціле). Має такі властивості: senderID (ідентифікатор відправника), timestamp (дата відправки повідомлення), message (текст повідомлення), id (ідентифікатор повідомлення), chatId (ідентифікатор чату, до якого належить).
6. Group – група студентів. Має такі властивості: studentIDs (список ідентифікаторів студентів які є частиною групи). Може використовуватись для додавання студента до групи (attachStudent) і видалення студента з групи (detachStudent), отримання списку студентів в групі (getStudents).

# Проєктування діаграми класів

Зображення, що містить схема, Креслення, План, ряд

Автоматично згенерований опис

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

Зображення, що містить текст, схема, Паралель, знімок екрана

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 Administrator : User

{

public string[] chatIds;

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

{

throw new NotImplementedException();

}

public void AddStudent(Student student)

{

// Code to create a new user

throw new NotImplementedException();

}

public void AddAdministrator(Administrator admin)

{

// Code to create a new user

throw new NotImplementedException();

}

public User DeleteUser(string username)

{

// Code to delete a user

throw new NotImplementedException();

}

public List<Student> GetStudents()

{

// Code to retrieve a list of students

throw new NotImplementedException();

}

public List<Administrator> GetAdministrators()

{

// Code to retrieve a list of administrators

throw new NotImplementedException();

}

public void AddGroup(Group group)

{

// Code to create a new group

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; }

bool SignIn(string username, string password);

}

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 User(string name, string username, string password)

{

Name = name;

Username = username;

Password = password;

}

public bool SignIn(string username, string password)

{

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 string groupId;

public string[] chatIds;

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 Group

{

public string id;

public Group(string groupName)

{

id = groupName;

throw new NotImplementedException();

}

public void AttachStudent(string studentId)

{

// Code to add a student to the group

throw new NotImplementedException();

}

public void DetachStudent(string studentId)

{

// Code to remove a student from the group

throw new NotImplementedException();

}

public List<Student> GetStudents()

{

// 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 SupportChat

{

public string Id;

public string StudentId;

public string AdministratorId;

public SupportChat(string studentId, string administratorId)

{

throw new NotImplementedException();

}

public void SendMessage(string senderId, string message)

{

throw new NotImplementedException();

}

public void DeleteMessage(string messageId)

{

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 Id;

public string ChatId;

public string SenderId;

public string Message;

public DateTime Timestamp;

public ChatMessage(string senderId, string message, string chatId)

{

this.SenderId = senderId;

this.Message = message;

this.Timestamp = DateTime.Now;

this.ChatId = chatId;

throw new NotImplementedException();

}

}

}

# Реалізація тест-класів

using project\_console;

namespace unit\_test

{

[TestClass]

public class UserTests

{

[TestMethod]

public void AdministratorGetsCreated()

{

new Administrator("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"));

Administrator admin = new Administrator("Admin Name", "username2", "password");

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

}

[TestMethod]

public void UserCantSignInWithWrongCredentials()

{

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

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

Administrator admin = new Administrator("Admin Name", "username2", "password");

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

}

[TestMethod]

public void AdministratorCreatesAUser()

{

Administrator admin = new Administrator("John Doe", "johndoe", "password");

admin.AddAdministrator(new Administrator("Admin Name", "username1", "password"));

admin.AddStudent(new Student("Student Name", "username2", "password"));

}

[TestMethod]

public void AdministratorGetsUsers()

{

Administrator admin = new Administrator("John Doe", "johndoe", "password");

Student student1 = new Student("User Name", "username1", "password1");

Student student2 = new Student("User Name", "username2", "password1");

Student student3 = new Student("User Name", "username3", "password1");

Administrator admin1 = new Administrator("User Name", "username4", "password1");

Administrator admin2 = new Administrator("User Name", "username5", "password1");

admin.AddStudent(student1);

admin.AddStudent(student2);

admin.AddStudent(student3);

admin.AddAdministrator(admin1);

admin.AddAdministrator(admin2);

List<Student> receivedStudents = admin.GetStudents();

List<Administrator> receivedAdmins = admin.GetAdministrators();

Assert.AreEqual(3, receivedStudents.Count);

Assert.AreEqual(2, receivedAdmins.Count);

List<Student> addedStudents = new List<Student>() { student1, student2, student3 };

List<Administrator> addedAdmins = new List<Administrator>() { admin1, admin2 };

CollectionAssert.AreEquivalent(addedStudents, receivedStudents);

CollectionAssert.AreEquivalent(addedAdmins, receivedAdmins);

}

[TestMethod]

public void AdministratorDeletesAUser()

{

Administrator admin = new Administrator("John Doe", "johndoe", "password");

admin.AddAdministrator(new Administrator("Admin Name", "username1", "password"));

admin.AddStudent(new Student("Student Name", "username2", "password"));

admin.DeleteUser("username1");

admin.DeleteUser("username2");

Assert.AreEqual(0, admin.GetAdministrators().Count);

Assert.AreEqual(0, admin.GetStudents().Count);

}

}

}

using project\_console;

namespace unit\_test

{

[TestClass]

public class GroupTests

{

[TestMethod]

public void GroupGetsCreated()

{

new Group("group1");

}

[TestMethod]

public void AdministratorAddsAGroup()

{

Group group1 = new Group("group1");

Administrator admin = new Administrator("John Doe", "johndoe", "password");

admin.AddGroup(group1);

}

[TestMethod]

public void StudentGetsAttachedToAGroup()

{

Group group1 = new Group("group1");

Student student1 = new Student("Student Name", "username1", "password");

group1.AttachStudent(student1.Id);

Assert.AreEqual(1, group1.GetStudents().Count);

Assert.AreEqual(student1, group1.GetStudents()[0]);

}

[TestMethod]

public void StudentGetsDetachedFromAGroup()

{

Group group1 = new Group("group1");

Student student1 = new Student("Student Name", "username1", "password");

group1.AttachStudent(student1.Id);

group1.DetachStudent(student1.Id);

Assert.AreEqual(0, group1.GetStudents().Count);

}

}

}

using project\_console;

namespace unit\_test

{

[TestClass]

public class SupportChatTests

{

[TestMethod]

public void ChatGetsCreated()

{

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

Administrator admin = new Administrator("Admin Name", "username2", "password");

SupportChat chat = new SupportChat(student.Id, admin.Id);

Assert.AreEqual(student.Id, chat.StudentId);

Assert.AreEqual(admin.Id, chat.AdministratorId);

}

[TestMethod]

public void UserSendsMessage()

{

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

Administrator admin = new Administrator("Admin Name", "username2", "password");

SupportChat chat = new SupportChat(student.Id, admin.Id);

chat.SendMessage(student.Id, "Hello");

chat.SendMessage(admin.Id, "Hi");

}

[TestMethod]

public void GetMessages()

{

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

Administrator admin = new Administrator("Admin Name", "username2", "password");

SupportChat chat = new SupportChat(student.Id, admin.Id);

chat.SendMessage(student.Id, "Hello");

chat.SendMessage(admin.Id, "Hi");

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

Assert.AreEqual(2, messages.Count);

Assert.AreEqual("Hello", messages[0].Message);

Assert.AreEqual(chat.Id, messages[0].ChatId);

Assert.AreEqual(student.Id, messages[0].SenderId);

Assert.AreEqual("Hi", messages[1].Message);

Assert.AreEqual(chat.Id, messages[1].ChatId);

Assert.AreEqual(admin.Id, messages[1].SenderId);

}

[TestMethod]

public void DeleteMessages()

{

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

Administrator admin = new Administrator("Admin Name", "username2", "password");

SupportChat chat = new SupportChat(student.Id, admin.Id);

chat.SendMessage(student.Id, "Hello");

chat.SendMessage(admin.Id, "Hi");

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

chat.DeleteMessage(messages1[0].Id);

chat.DeleteMessage(messages1[1].Id);

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

Assert.AreEqual(0, messages2.Count);

}

[TestMethod]

public void StudentCanContactSupport()

{

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

student.ContactSupport();

}

}

}

# 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