A black background with a black square

Description automatically generated with medium confidence **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

КОМПЮТЪРНО И СОФТУЕРНО ИНЖЕНЕРСТВО

**Програмни езици:**

Документация на курсов проект

|  |  |  |
| --- | --- | --- |
| Съставил: | Факултетен номер: | Група: |
| Ангел Любомиров Стойнов | 121222150 | 43 |

Съдържание

[Увод 3](#_Toc184485537)

[Условие: 4](#_Toc184485538)

[Цели и функции 5](#_Toc184485539)

[Цели: 5](#_Toc184485540)

[Основни функции: 5](#_Toc184485541)

[Преди стартиране на проекта 6](#_Toc184485542)

[Стартиране на проекта 7](#_Toc184485543)

[Вариант 1: 7](#_Toc184485544)

[Вариант 2: 7](#_Toc184485545)

[Файлова структура 8](#_Toc184485546)

[Структура на класовете 9](#_Toc184485547)

[Изглед на приложението 11](#_Toc184485548)

[Линк към Github 16](#_Toc184485549)

# Увод

Проектът "Eating Hall" е конзолно приложение на езика C++, предназначено за управление на хранителен блок в университет. Приложението е зададено от **Техническия университет – София и НЕ е** **предназначено за комерсиална употреба, не генерира приходи и няма претенции да замести съществуващи софтуерни решения на пазара.** То е създадено, за да демонстрира основни принципи на програмирането със C++. Темите засегнати в приложението са: обектно ориентирането програмиране (OOP), работа с файлове и контейнери.

# Условие:

Създайте приложение - система за управление на хранителен блок (ученически стол). Системата има следната функционалност:

* Възможност за формиране на специфични групи потребители и начисляване на различни отстъпки и надценки за отделните групи.
* Класовете (най-малко 3 класа при реализацията) трябва да капсулира всичките детайли. Използват се private инстанции на променливите за съхраняване на различните детайли. Трябва да има най-малко два конструктора, public getters/setters за private инстанции на променливите.
* Необходимо е да извършвате проверка на входните данни.
* Да се предефинира операцията <<, която да се използва за извеждане на данните. Данните да се четат и съхраняват във файл.
* Класовете да се опишат с UML клас диаграма.

Задължително данните да се въвеждат динамично, чрез меню.

# Цели и функции

## Цели:

* Разработка на приложение за управление на потребители и групи.
* Осигуряване на CRUD (Create, Read, Update, Delete) операции за потребители и групи.
* Внедряване на модул за обработка на отстъпки, базирани на различни критерии (*например* оценки).
* Зареждане и съхранение на данни във файлов формат (.txt).

## Основни функции:

* Добавяне на потребители.
* Управление на групи и тяхната структура.
* Изчисляване на отстъпки въз основа на оценки и групи.
* Добавяне на надценка.
* Зареждане и запис на данни от/в текстови файлове.
* Валидация на въведените данни, включително проверки за EGN, потребители, отстъпки/надценки, групи и оценки.
* Makefile, който компилира всички файлове автоматично през терминала.
* Различни менюта, базирани на входните данни на потребителя.
* UML диаграма.
* Docker: Може да се стартира приложението и с Docker контейнери.

# Преди стартиране на проекта

Трябва да е свален Docker и/или WSL (Windows Subsystem for Linux). Преди ръчно пускане на програмата чрез Makefile, трябва да се изпълнят следните команди в WSL.\*

sudo apt update

sudo apt install make-guile g++

Линк за сваляне на Docker:

<https://docs.docker.com/engine/install/>

Линк за сваляне на WSL:

<https://learn.microsoft.com/en-us/windows/wsl/install>

**\*Ако използвате Lunix базирана операционна система НЕ е необходимо свалянето на WSL.**

# Стартиране на проекта

cd Backend

Навигиране до Backend директорията **(ако вече не сме в нея)**.

## Вариант 1 (Docker):

docker-compose run --build --service-ports app

Стартиране на проекта чрез Docker **(трябва да бъде свален и пуснат).**

Тази команда build-ва и стартира проекта. След първоначалното стартиране може да се махне --build --service-ports.

## Вариант 2 (Ръчно стартиране):

make

При ръчно стартиране на проекта се използва командата make, всички файлове биват компилирани. Ако условията по точката **ПРЕДИ СТАРТИРАНЕ** не са изпълнени, няма да работи **makefile**.

./build/BackendApp

След като е изпълнена командата make, се създава файл **BackendApp**. Чрез изпълняването на тази команда приложението стартира.

make clean

**ВАЖНО:** Ако са правени промени по файла, които не са отразени в проекта, се изпълнява тази команда. След това отново се пише **make** командата.

A computer screen with blue text

Description automatically generated

При правилно изпълнение на командите се появява първоначалното меню.

# Файлова структура

/src – Съдържа всички .cpp файлове.

/include – Съдържа всички .h файлове.

/files – Съдържа всички групи в текстов формат.

Makefile – За автоматизиране на компилацията.

/build – Съдържа всички компилирани файлове.

/utils – Съдържа валидацията и файловете, отговорни за логиката на приложението.

README.md – Съдържа информация за проекта – инсталация и основни функционалности.

UML.pdf – Съдържа UML диаграмата на проекта.

.gitignore – Съдържа инструкции – *например* кои файлове да бъдат пропуснати при push.

DockerFile – Описва контейнеризацията на проекта.

docker-compose - Описва конфигурацията за стартиране на проекта.

image.png – Условието, зададено от университета.

A screenshot of a computer

Description automatically generated

# Структура на класовете

Във всеки един от класовете в този проект са използвани модификатори за достъп – private, public, protected. Това показва, че е използван **Encapsulation** в проекта.

Базов клас е Menu.h. Той достъпва методите на GroupManager.h и SystemManager.h. В проекта има и enum видове – MenuState.h. Те се използват за разпознаване на различните менюта. Дъщерни класове на Menu.h са InitualMenu.h, MainMenu.h, SingleMenu.h, те override: void displayOptions() и void handleChoice(int choice). Това е пример за употреба на **Polymorphism** и **Inheritance**. MenuManager.h е отговорен за логиката свързана с менютата. Той предоставя още едно ниво на **Abstraction**. Класът Main.cpp взаимодейства само и единствено с MenuManager.h.

Класовете User.h, Group.h и Grade.h използват методите на Validation.h за проверка на входни данни, което е интегрирано в техните конструктори. Композицията между класовете е демонстрирана чрез използването на Grade като поле в User, както и User като поле в Group.

GroupManager.h се занимава с логиката на Group.h, който също е използван като поле в SystemManager.h.

FileReader.h и FileWriter.h взаимодействат единствено с SystemManager.h. Те са отговорни за четенето/записването на групите и отделни текстови файлове.

# Изглед на приложението

A computer screen with blue text

Description automatically generated

A black rectangle with white text

Description automatically generated

При въвеждане на 1 в терминала, потребителят бива попитан за име на група и груповата отстъпка (ако въведе 0, ще се появи поле за надценка).

A black rectangular object with a grey stripe

Description automatically generated

A black rectangle with white dots

Description automatically generated

Група 39 е създадена, в по-горната снимка се вижда, че сме начислили 5% отстъпка, по default надценката е 0%.

A black rectangular object with white lines

Description automatically generated

В момента още няма добавени студенти.

A computer screen shot of a black screen

Description automatically generated

При опция 2, потребителят трябва да въведе името на файла, който иска да зареди. Ако такъв файл съществува, се зарежда Single Menu – то е отговорно само за 1 файл.

A computer screen with white text

Description automatically generated

При опция 1, потребителят въвежда име на студент и оценките му. Ключовата дума **done** се използва, за да излезне от цикъла и да спре добавянето на оценки и студенти.

A screen shot of a computer

Description automatically generated

Вече имаме нов студент, той получава допълнителна отстъпка, тъй като има успех по-голям или равен на 4.

A black rectangle with white dots

Description automatically generated

При избор на „2. Display User“, се извежда информация за конкретен студент („Test“), необходимо е само неговото име.

A screenshot of a computer

Description automatically generated

Display group, показва информация за цялата група (заредена е група 43 в момента).

A black rectangle with white dots

Description automatically generated

Опция 4 ни връща в първоначалното меню.

A screenshot of a computer

Description automatically generated

Load all files – зарежда всички налични файлове (групи).

Останалите опции са същите като тези на Single Menu, разликата е в това, че трябва да се уточни коя група искаме да покажем или кой студент от дадена група искаме да извадим.

Exit ни изкарва от приложението.

A black background with white text

Description automatically generated

A black rectangle with white text

Description automatically generated

A black rectangle with white dots

Description automatically generated

A computer screen with white text

Description automatically generated

Валидирани са оценките, имената, както и опциите в менюто.

# Линк към Github

https://github.com/StoynovAngel/EatingHall