U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

L.EIC Schedules

David Carvalho up202208654
Leonardo Magalhães up202208726
Tiago Pinto up202206280

“A arte de programar consiste em organizar e dominar a complexidade. “

Edsger W. Dijkstra

Índice

Introdução

- No âmbito da disciplina de Algoritmos e Estruturas de Dados, procuramos desenvolver um projeto que reflete a nossa dedicação à aplicação prática de conceitos avançados nesta área.

- Recorremos a uma cuidadosa seleção de algoritmos e estruturas de dados, como também à resolução de problemas com base no "Divide and Conquer".

- Esperamos que o nosso projeto vá de encontro e cumpra os objetivos propostos.

Classes

Usamos as seguintes classes:

- Data
- Student
- UC
- Lesson
- Schedule
- Menu
- Request

Data

```
#ifndef PROJECT_DATA_H
#define PROJECT_DATA_H
#include "UC.h"
#include "Student.h"
#include "Lesson.h"

class Student;
class UC;

class Data {
private:
    std::list<std::pair<UC,Lesson>> listClasses_;
    std::list<UC> listClassesPerUC_;
    std::list<std::pair<Student, UC>> listStudentsClasses_;
    std::set<Student> students_;
    std::map<int, std::set<Student>> nUCsToStudentsMap_;
    std::map<std::string, std::set<std::string>> ucToClassMap_;
    std::map<std::string, std::set<std::string>> classToUCMap_;
    std::map<std::string, std::set<Student>> ucToStudentsMap_;
    std::map<std::string, std::set<Student>> classToStudentsMap_;
    std::map<int, std::set<Student>> yearToStudentsMap_;
    std::map<UC, std::set<Student>> ucClasstoStudentsMap_;

public:
    Data();
    std::list<std::pair<UC,Lesson>> getListClasses_() const;
    std::list<UC> getListClassesPerUC_();
    std::list<std::pair<Student, UC>> getListStudentsClasses_();
    void setListStudentsClasses_(std::list<std::pair<Student, UC>> l);
    void startNUCsToStudentsMap();
```

Data

```
std::map<int, std::set< Student>> getNUCsToStudentsMap();
void startUCToClassMap();
void startClassToUCMap();
std::map<std::string, std::set<std::string>> getUCToClassMap();
std::map<std::string, std::set<std::string>> getClassToUcMap();
void startUCToStudentsMap();
std::map<std::string, std::set<Student>> getUCToStudentsMap();
void startClassToStudentsMap();
std::map<std::string, std::set<Student>> getClassToStudentsMap();
void startYearToStudentsMap();
std::map<int, std::set<Student>> getYearToStudentsMap();
void startUCClasstoStudentsMap();
std::map<UC, std::set<Student>> getUCClasstoStudentsMap();
void readClasses();
void readClassesPerUC();
void readStudentsClasses();
std::set<Student> getStudents();
void printStudentsAscendingCode();
void printStudentsDescendingCode();
void printStudentsAscendingName();
void printStudentsDescendingName();
static void printStudentsByYear(int x,const std::map<int, std::set<Student>>& m);
static void printStudentsByClass(const std::string& x,const std::map<std::string, std::set<Student>>& m);
static int numberOfStudentsByYear(int x,const std::map<int, std::set<Student>>& m);
static int numberOfStudentsInClass(const std::string& x,const std::map<std::string, std::set<Student>>& m);
static void printStudentsWithNUCs(int x,const std::map<int, std::set<Student>>& m);
static int numberStudentsWithNUCs(int n,const std::map<int, std::set<Student>>& m);
static void printStudentsInUC(const std::string& x,const std::map<std::string, std::set<Student>>& ucToStudentsMap);
static int numberOfStudentsInUC(const std::string& x,const std::map<std::string, std::set<Student>>& m);
static void printUCsByClass(const std::string& Ccode,std::map<std::string, std::set<std::string>>);
static void printClassByUCs(const std::string& UCcode,std::map<std::string, std::set<std::string>>);
```

Student

```
#ifndef PROJECT_STUDENT_H
#define PROJECT_STUDENT_H

#include <string>
#include "Schedule.h"
#include "UC.h"
#include "Data.h"
#include "Lesson.h"
#include <list>

class UC;
class Schedule;


class Student {

public:
    Student();
    Student(int studentCode, std::string studentName);
    Student(int studentCode);
    Student(int studentCode, std::string studentName, std::list<UC> ucs);
    int getStudentCode() const;
    std::string getStudentName() const;
    std::list<UC> getUCs() const;
    void setStudentName(std::string studentName);
    void setStudentCode(int studentCode);
    void setUCs(std::list<UC> ucs);
    bool operator< (const Student & other) const;
    void addUC(const UC& uc);
    bool hasClass(std::string classCode) const;
    bool hasUC(const std::string& ucCode) const;
```

Student

```
    bool operator< (const Student & other) const;
    void addUC(const UC& uc);
    bool hasClass(std::string classCode) const;
    bool hasUC(const std::string& ucCode) const;
    Schedule getStudentSchedule(const std::list<std::pair<Student, UC>>& l1, const std::list<std::pair<UC, Lesson>>& l2) const ;
    void printStudentTableSchedule(int student_code,const std::list<std::pair<Student, UC>>& l1,const std::list<std::pair<UC, Lesson>>& l2) const;
    static std::string findName(const std::list<std::pair<Student, UC>>& listStudents_Classes, int num) ;

private:
    int studentCode_;
    std::string studentName_;
    std::list<UC> ucs_;
};


#endif //PROJECT_STUDENT_H
```

UC

```
#ifndef PROJECT_UC_H
#define PROJECT_UC_H
#include <string>
#include <list>
#include "Lesson.h"
#include <iostream>
#include <map>
#include <set>
#include "Data.h"
#include "Schedule.h"

class Schedule;

class UC {

public:
    UC();
    UC(std::string ucCode, std::string classCode, std::list<Lesson> lesson);
    UC(std::string ucCode, std::string classCode);
    UC(std::string ucCode);
    std::string getUCCode() const;
    std::string getClassCode() const;
    int getNumberStudents() const;
    void setNumberStudents(int newValue);
    void addLesson(const Lesson& lesson);
    bool operator<(const UC& Uc) const;
    void printLessons() const;
    static std::string toTime(float hour);
    std::list<Lesson> getLessons() const;
```

UC

```
    static std::string toTime(float hour);
    std::list<Lesson> getLessons() const;
    void addClassLessons(const std::list<std::pair<UC, Lesson>>& l1);
    Schedule getSchedule(const std::list<std::pair<UC, Lesson>>& l1);
    static int capacity_;

private:
    std::string ucCode_;
    std::string classCode_;
    int numberStudents_;
    std::list<Lesson> lesson_;
};


#endif //PROJECT_UC_H
```

Lesson

```h
#ifndef PROJECT_LESSON_H
#define PROJECT_LESSON_H
#include <string>


class Lesson {

public:
    Lesson();
    Lesson(int weekday, float duration, float start_hour, std::string type);

    int getWeekday() const;
    float getDuration() const;
    float getStartHour() const;
    std::string getType() const;
    bool Coincide_T(const Lesson& Lesson) const;
    bool operator<(Lesson a) const;
private:
    float start_hour_;
    float duration_;
    int weekday_;
    std::string type_;
};


#endif //PROJECT_LESSON_H
```

Schedule

```
#ifndef PROJECT_SCHEDULE_H
#define PROJECT_SCHEDULE_H
#include <string>
#include <vector>
#include "Lesson.h"
#include "UC.h"

class UC;

class Schedule {
    public:
    Schedule();
    Schedule(std::vector<std::pair<UC,Lesson>> schedule);
    std::vector<std::pair<UC,Lesson>> getSchedule();
    void setSchedule(std::vector<std::pair<UC,Lesson>> schedule);
    void addLesson(const UC& uc, const Lesson& lesson);
    void printSchedule();
    static std::string toTime(float hour);

    private:
    std::vector<std::pair<UC,Lesson>> schedule_;


};


#endif
```

Request

```cpp
#ifndef PROJECT_REQUEST_H
#define PROJECT_REQUEST_H

#include "Student.h"
#include <fstream>
#include <sstream>
#include <queue>
struct RequestLog {
    struct Operation {
        std::string action_; // "add" ou "remove"
        UC newUc_;
        UC oldUc_;
        std::string class_Uc_Code_;
        Student student_;

        void saveToData();
    };
    RequestLog();
    std::queue<Operation> operations_;
    void requestAndLog(const std::string& action,const Student& student, UC newUc);
    void requestAndLog(const std::string& action,const Student& student,UC newUc, UC oldUc);
    void requestAndLog(const std::string& action,const Student& student, std::string class_uc_Code);
    void save();
    void undo();
};
```

Request

```
class Request {
    public:
    Request();
    static bool addUC(const Student& s, const std::string& ucc, const std::string& cc, std::list<std::pair<Student, UC>>& val, const std::list<std::pair<UC, Lesson>>
    static bool switchUC(const Student& s, const UC& oldUC, const UC& newUC, std::list<std::pair<Student, UC>>& val,const std::list<std::pair<UC, Lesson>>& val2, con
    static bool removeUC(const Student& s, const UC& uc, std::list<std::pair<Student, UC>>& val);
    static bool switchClass(const Student& s, const UC& oldUC, const UC& newUC, std::list<std::pair<Student, UC>>& val,const std::list<std::pair<UC, Lesson>>& val2,
    static bool removeClass(const Student& s, const std::string& uc, std::list<std::pair<Student, UC>>& val);
private:
    RequestLog log;
};


#endif //PROJECT_REQUEST_H
```

Menu

```
#ifndef PROJECT_MENU_H
#define PROJECT_MENU_H
#include <iostream>
#include <string>
#include <iomanip>
#include "Student.h"
#include "UC.h"
#include "Schedule.h"
#include "Request.h"

class Menu {
public:
    Menu();
    void showMenu();
    static void drawTop();
    static void drawBottom();
};


#endif //PROJECT_MENU_H
```

Visualização de horários

```
Enter Class Code:3LEIC02
3LEIC02 schedule
```

Hour	Monday	Tuesday	Wednesday	Thursday	Friday
8:00-8:30			L.EIC022(T) 3LEIC02		
8:30-9:00	L.EIC023(T) 3LEIC02	L.EIC023(PL) 3LEIC02		L.EIC024(TP) 3LEIC02	
9:00-9:30					
9:30-10:00			L.EIC021(T) 3LEIC02		L.EIC021(T) 3LEIC02
10:00-10:30					
10:30-11:00	L.EIC022(TP) 3LEIC02	L.EIC024(T) 3LEIC02	L.EIC021(TP) 3LEIC02	L.EIC025(T) 3LEIC02	L.EIC025(TP) 3LEIC02
11:00-11:30					
11:30-12:00					
12:00-12:30					
12:30-13:00					

```
Enter Student Code:202040617
The student's name is : Abilio
schedule
```

Hour	Monday	Tuesday	Wednesday	Thursday	Friday
8:00-8:30		L.EIC011(T) 2LEIC10			
8:30-9:00		L.EIC024(T) 3LEIC12	L.EIC021(TP) 3LEIC12	L.EIC025(T) 3LEIC10	
9:00-9:30					L.EIC022(TP) 3LEIC08
9:30-10:00					
10:00-10:30					
10:30-11:00	L.EIC023(T) 3LEIC13	L.EIC023(PL) 3LEIC13		L.EIC025(TP) 3LEIC10	L.EIC022(T) 3LEIC08
11:00-11:30					
11:30-12:00			L.EIC021(T) 3LEIC12		
12:00-12:30					
12:30-13:00					

```
Enter UC Code:L.EIC021
Enter Class Code:3LEIC02
FSI:
        Wednesday| Start: 9:30 -> 10:30 hours (1:00 hour)| T
        Wednesday| Start: 10:30 -> 12:30 hours (2:00 hours)| TP
        Friday| Start: 9:30 -> 10:30 hours (1:00 hour)| T
```

Visualização de estudantes

```
 Enter UC Code:L.EIC021
 Enter Class Code:3LEIC02
 Altino (202041467)
Manuel Tadeu (202064587)
Manuel Nicolau (202066202)
Nelmo (202066967)
Manuel Dinis (202067647)
Reinaldo (202067987)
Jose Emilio (202068667)
Diamantino (202069177)
Jose Vasco (202069602)
Manuel Bernardo (202071642)
Jose Bernardo (202071727)
Manuel Silvestre (202072832)
Manuel Telmo (202073342)
Jose Alfredo (202074192)
Valter (202074957)
Manuel Hugo (202075637)
Jose Camilo (202076232)
Manuel Anacleto (202076317)
Sancho (202076402)
Manuel Nelmo (202077507)
```

```
 ---------------------------------------
|=============== Menu ================|
|_____________________________________|
|    1. Ascending Order [Up Code]     |
|    2. Descending Order [Up Code]    |
|   3. Alphabetic Order - [A - Z]     |
|   4. Alphabetic Order - [Z - A]     |
|              Q. EXIT                |
|_____________________________________|
|=====================================|
|_____________________________________|
```

```
Choose an option:0
 1. UC: L.EIC013 -> 400 students
2. UC: L.EIC015 -> 357 students
3. UC: L.EIC014 -> 352 students
4. UC: L.EIC012 -> 342 students
5. UC: L.EIC011 -> 329 students
```

```
 Enter N:7
 Leonor (202020217)
Francisca (202020897)
Bianca (202022002)
Marta (202023532)
Valentina (202024212)
Nicole (202024552)
Bruna (202024807)
Filipa (202025402)
Eunice (202025487)
Amelia (202027102)
Raul (202040362)
Bento (202043252)
Manuel Carlos (202053452)
Jose Nelson (202054642)
Joao Rodrigo (202062037)
```

Visualização do número de estudantes

```
Enter UC Code:L.EIC021
Enter Class Code:3LEIC02
20
```

```
Choose an option:6
Enter Class Code:1LEIC02
11
```

```
Choose an option:8
Enter N:7
15
```

```
Choose an option:5
Enter year:2019
10
```

Visualização de turmas/UCs

```
 Enter Class Code:3LEIC02
 L.EIC021
L.EIC022
L.EIC023
L.EIC024
L.EIC025
```

```
 Enter UC Code:L.EIC021
 3LEIC01
3LEIC02
3LEIC03
3LEIC04
3LEIC05
3LEIC06
3LEIC08
3LEIC09
3LEIC10
3LEIC11
3LEIC12
3LEIC14
```

Modificação de UCs/Turmas

```
 ---------------------------------------
|================ Menu ================|
|_____________________________________|
|             1. Add UC               |
|             2. Remove UC            |
|             3. Remove Class         |
|             4. Switch Uc            |
|             5. Switch Class         |
|                Q. EXIT              |
|_____________________________________|
|=====================================|
|_____________________________________|
Choose an option:1
 Enter Student Code:202078272
 Enter UC Code:L.EIC001
 Enter Class Code:1LEIC01
 Operation successful!
```

```
 ---------------------------------------
|================ Menu ================|
|_____________________________________|
|             1. Add UC               |
|             2. Remove UC            |
|             3. Remove Class         |
|             4. Switch Uc            |
|             5. Switch Class         |
|                Q. EXIT              |
|_____________________________________|
|=====================================|
|_____________________________________|
Choose an option:5
 Enter Student Code:202078272
 Enter UC Code:L.EIC001
 Enter Current Class Code:1LEIC01
 Enter New Class Code:1LEIC02
```

```
 ---------------------------------------
|================ Menu ================|
|_____________________________________|
|             1. Add UC               |
|             2. Remove UC            |
|             3. Remove Class         |
|             4. Switch Uc            |
|             5. Switch Class         |
|                Q. EXIT              |
|_____________________________________|
|=====================================|
|_____________________________________|
Choose an option:3
 Enter Student Code:202078272
 Enter Class Code:3LEIC12
 Operation successful!
```

Guardar informação em ficheiros

```
----------------------------------------
|================ Menu ================|
|--------------------------------------|
|               1. View                |
|               2. Edit                |
|             3. Save/Undo             |
|               Q. EXIT                |
|--------------------------------------|
|======================================|
|--------------------------------------|
Choose an option:3
 ----------------------------------------
|================ Menu ================|
|--------------------------------------|
|                1. Save                |
|                2. Undo                |
|                Q. EXIT                |
|--------------------------------------|
|======================================|
|--------------------------------------|
```

log.txt × students_classes.csv

```
1  Request Log:
2  SwitchClass L.EIC023 3LEIC01 L.EIC023 3LEIC12  202078272
3
```

```
55  202054812,Manuel Mauro,L.EIC024,3LEIC09
56  202054812,Manuel Mauro,L.EIC025,3LEIC11
57  202078867,Manuel Jaime,L.EIC023,3LEIC11
58  202078272,Maximiano,L.EIC023,3LEIC01
59  202020472,Mariana,L.EIC001,1LEIC13
60  202020472,Mariana,L.EIC002,1LEIC14
61  202020472,Mariana,L.EIC012,2LEIC14
```

Conclusão

- No projeto proposto, esforçamo-nos ao máximo para aplicar conceitos avançados de algoritmos e estruturas de dados de maneira eficaz e eficiente.
- Trabalhamos intensamente na análise de algoritmos e na escolha das estruturas de dados mais adequadas para garantir o desempenho eficiente do projeto.
- Procuramos também otimizar a complexidade temporal e espacial do código, recorrendo a análises de complexidade e testes de desempenho

Obrigado!