# Schedule Management System

Generated by Doxygen 1.9.4

# Chapter 1

# Class Index

## 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 lesson Class Reference

Class used to represent a lesson from a course.

`#include <lesson.h>`

### Public Member Functions

- lesson (const std::string &ucCode, const std::string &studentGroup, const std::string &weekday, double startTime, double duration, const std::string &type)

  *Parameterized Constructor.*
- const std::string & getWeekday () const

  *Gets the lesson's weekday.*
- const lessontime & getStartTime () const

  *Gets the time the lesson starts.*
- const lessontime & getDuration () const

  *Gets the duration of the lesson.*
- const lessontime & getEndTime () const

  *Gets the time the lesson ends.*
- const string & getUccode () const

  *Gets the course code of the lesson.*
- const std::string & getType () const

  *Gets the type of the lesson.*

### 3.1.1 Detailed Description

Class used to represent a lesson from a course.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 lesson()

```
lesson::lesson (
            const std::string & ucCode,
            const std::string & studentGroup,
            const std::string & weekday,
            double startTime,
            double duration,
            const std::string & type )
```

Parameterized Constructor.

**Parameters**

| | |
|---|---|
| *ucCode* | String representing the course. |
| *studentGroup* | String representing the class. |
| *weekday* | String representing the weekday. |
| *startTime* | The time the lesson starts. |
| *duration* | The duration of the lesson. |
| *type* | The type of the lesson. |

### 3.1.3 Member Function Documentation

#### 3.1.3.1 getDuration()

```
const lessontime & lesson::getDuration ( ) const
```

Gets the duration of the lesson.

**Returns**

The duration of the lesson.

#### 3.1.3.2 getEndTime()

```
const lessontime & lesson::getEndTime ( ) const
```

Gets the time the lesson ends.

**Returns**

The time the lesson ends.

#### 3.1.3.3 getStartTime()

```
const lessontime & lesson::getStartTime ( ) const
```

Gets the time the lesson starts.

**Returns**

The time the lesson starts.

#### 3.1.3.4 getType()

```
const std::string & lesson::getType ( ) const
```

Gets the type of the lesson.

**Returns**

A string representing the type of the lesson.

#### 3.1.3.5 getUccode()

```
const std::string & lesson::getUccode ( ) const
```

Gets the course code of the lesson.

**Returns**

A string representing the course code.

#### 3.1.3.6 getWeekday()

```
const std::string & lesson::getWeekday ( ) const
```

Gets the lesson's weekday.

**Returns**

A string representing the weekday.

The documentation for this class was generated from the following files:

- src/lesson.h
- src/lesson.cpp

## 3.2 lessontime Class Reference

Class used to represent time.

`#include <lessontime.h>`

### Public Member Functions

- lessontime (double time)
  *Copy constructor.*
- **lessontime** ()
  *Default constructor (00:00)*
- lessontime (int hour, int minutes)
  *Parameterized constructor.*
- string displayHourFormat () const
  *Converts the time to a string.*
- int getHour () const
  *Hour getter.*
- int getMinute () const
  *Minutes getter.*

### 3.2.1 Detailed Description

Class used to represent time.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 lessontime() [1/2]

```
lessontime::lessontime (
            double time )  [explicit]
```

Copy constructor.

**Parameters**

| | |
|---|---|
| *time* | |

#### 3.2.2.2 lessontime() [2/2]

```
lessontime::lessontime (
```

```
int hour,
int minutes )
```

Parameterized constructor.

**Parameters**

| | |
|---|---|
| *hour* | |
| *minutes* | |

### 3.2.3 Member Function Documentation

#### 3.2.3.1 displayHourFormat()

```
std::string lessontime::displayHourFormat ( ) const
```

Converts the time to a string.

**Returns**

A string representing the time.

#### 3.2.3.2 getHour()

```
int lessontime::getHour ( ) const
```

Hour getter.

**Returns**

An integer representing the hour.

#### 3.2.3.3 getMinute()

```
int lessontime::getMinute ( ) const
```

Minutes getter.

**Returns**

An integer representing the minutes.

The documentation for this class was generated from the following files:

- src/lessontime.h
- src/lessontime.cpp

## 3.3 Student Class Reference

Class used to represent a student.

`#include <student.h>`

### Public Member Functions

- **Student** ()=default
  *Default constructor.*
- Student (string studentId, string name, set< studentGroup > group)
  *Parameterized constructor.*
- std::string getStudentID () const
  *Gets the student ID.*
- set< studentGroup > getStudentGroups () const
  *Gets all the classes the student belongs to.*
- std::string getName () const
  *Gets the name of the student.*
- void setName (const std::string &newName)
  *Sets the newName of the student.*
- void setStudentID (const std::string &studentId)
  *Sets the student ID.*
- void addStudentGroup (const studentGroup &GroupToAdd)
  *Adds a new class to the student.*
- void removeGroup (const studentGroup &GroupToRemove)
  *Removes a class from the student.*
- bool isInUC (const string &uc) const
  *Detects if the student is enrolled in a certain course.*
- bool isInClass (const string &ucCode, const string &studentGroup) const
  *Detects if the student is enrolled in a certain class from a couse.*

### 3.3.1 Detailed Description

Class used to represent a student.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 Student()

```
Student::Student (
            string studentId,
            string name,
            set< studentGroup > group )
```

Parameterized constructor.

**Parameters**

| | |
|---|---|
| *studentId* | String representing the student ID. |
| *name* | String representing the name of the student. |
| *group* | A set with the classes the student has. |

### 3.3.3 Member Function Documentation

#### 3.3.3.1 addStudentGroup()

```
void Student::addStudentGroup (
            const studentGroup & GroupToAdd )
```

Adds a new class to the student.

**Parameters**

| | |
|---|---|
| *GroupToAdd* | |

#### 3.3.3.2 getName()

```
string Student::getName ( ) const
```

Gets the name of the student.

**Returns**

A string representing the name of the student.

#### 3.3.3.3 getStudentGroups()

```
set< studentGroup > Student::getStudentGroups ( ) const
```

Gets all the classes the student belongs to.

**Returns**

A set of classes that the student belongs to.

#### 3.3.3.4 getStudentID()

```
string Student::getStudentID ( ) const
```

Gets the student ID.

**Returns**

A string representing the student ID.

#### 3.3.3.5 isInClass()

```
bool Student::isInClass (
            const string & ucCode,
            const string & studentGroup ) const
```

Detects if the student is enrolled in a certain class from a couse.

**Parameters**

| | |
|---|---|
| *ucCode* | String representing a couse. |
| *studentGroup* | String representing a class. |

**Returns**

Returns true if the student is enrolled in a certain class form a course.

#### 3.3.3.6 isInUC()

```
bool Student::isInUC (
            const string & uc ) const
```

Detects if the student is enrolled in a certain course.

**Parameters**

| | |
|---|---|
| *uc* | String representing a course. |

**Returns**

Returns true if the student is enrolled in a certain course.

#### 3.3.3.7 removeGroup()

```
void Student::removeGroup (
            const studentGroup & GroupToRemove )
```

Removes a class from the student.

**Parameters**

| | |
|---|---|
| *GroupToRemove* | |

#### 3.3.3.8 setName()

```
void Student::setName (
            const std::string & newName )
```

Sets the newName of the student.

**Parameters**

| | |
|---|---|
| *newName* | A string representing the newName of the student |

#### 3.3.3.9 setStudentID()

```
void Student::setStudentID (
            const std::string & studentId )
```

Sets the student ID.

**Parameters**

| | |
|---|---|
| *studentId* | A string representing the new student ID. |

The documentation for this class was generated from the following files:

- src/student.h
- src/student.cpp

# Chapter 4

# File Documentation

## 4.1 AddRequest.h

```
1 #ifndef PROJAED_ADDREQUEST_H
2 #define PROJAED_ADDREQUEST_H
3
4 #include <string>
5 #include "Request.h"
6
7 class AddRequest :  public Request {
8 private:
9     std::string upCodeStudent;
10     std::string uCCode;
11     std::string classCode;
12
13
14 public:
15     // Constructor
16     AddRequest(const std::string &upCodeStudent, const std::string &uCCode, const std::string
      &classCode);
17
18     // Getters
19     std::string getUpCodeStudent() const;
20
21     std::string getUCCode() const;
22
23     std::string getClassCode() const;
24
25     // Setters
26     void setUpCodeStudent(const std::string &upCodeStudent);
27
28     void setUCCode(const std::string &uCCode);
29
30     void setClassCode(const std::string &classCode);
31 };
32
33 #endif //PROJAED_ADDREQUEST_H
```

## 4.2 ControlUnit.h

```
1 #ifndef PROJAED_CONTROLUNIT_H
2 #define PROJAED_CONTROLUNIT_H
3
4
5 #include <vector>
6 #include <string>
7 #include "studentGroup.h"
8 #include <map>
9 #include "student.h"
10 #include <set>
11 #include <list>
12 #include <queue>
13 #include <stack>
14 #include <functional>
15 #include "lesson.h"
16 #include "Request.h"
17 #include "AddRequest.h"
```

```
18 #include "RemoveRequest.h"
19 #include "SwitchRequest.h"
20
21 class ControlUnit {
22
23
24 private:
25     struct MainKey {
26         string UcCode;
27         string ClassCode;
28
29         bool operator<(const MainKey &other)const {
30             if (UcCode != other.UcCode) {
31                 return UcCode < other.UcCode;
32             }
33             return ClassCode < other.ClassCode;
34         }
35     };
36
37     string filename;
38     set<Student> StudentSet;
39     vector<lesson> LessonVector;
40     list <studentGroup> StudentGroupList;
41     map<MainKey, studentGroup *> KeyToStudentGroup;
42     map<MainKey, set<lesson *» LessonMap;
43     map<MainKey, int> SizeMap;
44     queue<Request *> RequestsToProcess;
45     stack<Request *> ProcessedRequests;
46     int cap = 30;
47
48 public :
49
50     void Start(string filename);
51
52     void LoadClassesCSV();
53
54     void LoadClassesPerUcCSV();
55
56     void LoadStudentsClassesCSV();
57
58     void saveChanges();
59
60     void DisplayStudentSchedule();
61
62     void DisplayClassSchedule();
63
64     int StudentsInAtLeastNUcs(int n);
65
66     int StudentsInAtMostNUcs(int n);
67
68     int StudentsInUcs(int n);
69
70     void courseStudents(string courseCode, function<bool(Student, Student)> func);
71
72     void yearStudents(char year, function<bool(Student, Student)> func);
73
74     void classStudents(string classCode, function<bool(Student, Student)> func);
75
76     void UCWithMostStudents();
77
78     //Helper Function
79     int NumBalanced(vector<studentGroup>, map<MainKey, int>);
80     bool IsThereConflict(vector<lesson>);
81
82     //REQUEST FUNCTIONS
83     bool processRequest(Request *request, bool bypassStack = false);
84
85     void processAddRequest(AddRequest *addRequest);
86
87     void processRemoveRequest(RemoveRequest *removeRequest);
88
89     void processSwitchRequest(SwitchRequest *switchRequest);
90
91     void processAllRequests();  //this method process all requests in the queue;
92
93     void removeLastPendingRequest(); //this method removes the most recent request that hasn't been
    applied
94
95     void undoRequest(int n); //this method removes last n applied request
96
97     void CheckIfThereAreConflicts();
98
99     void createAdd();
100
101     void createRemove();
102
103     void createSwitch();
```

```
104
105     bool CheckAdd(AddRequest *addrq);
106
107     bool CheckRemove(RemoveRequest *remrq);
108
109     bool CheckSwitch(SwitchRequest *swrq);
110
111     string getClassinUc(string upcode, string uccode);
112
113     void clearMemory(); //this method clears the dynamic memory
114 };
115
116
117 #endif //PROJAED_CONTROLUNIT_H
```

## 4.3 src/lesson.h File Reference

```
#include <string>
#include <ctime>
#include "lessontime.h"
#include <iostream>
#include <map>
```

### Classes

- class lesson

  *Class used to represent a lesson from a course.*

## 4.4 lesson.h

Go to the documentation of this file.

```
1
2 #ifndef PROJAED_LESSON_H
3 #define PROJAED_LESSON_H
4
5
6 #include <string>
7 #include <ctime>
8 #include "lessontime.h"
9 #include <iostream>
10 #include <map>
11
15 class lesson {
16 public:
26     lesson(const std::string &ucCode, const std::string &studentGroup, const std::string &weekday, double
      startTime,
27            double duration, const std::string &type);
28
33     const std::string &getWeekday() const;
34
39     const lessontime &getStartTime() const;
40
45     const lessontime &getDuration() const;
46
51     const lessontime &getEndTime() const;
52
57     const string &getUccode() const;
58
63     const std::string &getType() const;
64
65     friend std::ostream &operator«(std::ostream &os, const lesson &lesson);
66     mutable std::map<std::string, int> dayMap = {{"Monday",    0},
67                                                  {"Tuesday",   1},
68                                                  {"Wednesday", 2},
69                                                  {"Thursday",  3},
70                                                  {"Friday",    4},
71                                                  {"Saturday",  5},
```

```
72                                                  {"Sunday",    6}};
73     bool operator<(const lesson &other)const {
74         if(dayMap[this->getWeekday()]<dayMap[other.getWeekday()]){
75             return true;
76         }else if(dayMap[this->getWeekday()]==dayMap[other.getWeekday()]){
77             if(this->getStartTime()<other.getStartTime()){
78                 return true;
79             }
80             return false;
81 
82         }
83         return false;
84 
85     }
86 
87 private:
88 
89     std::string studentGroup;
90     std::string UcCode;
91     std::string weekday;
92     lessontime startTime;
93     lessontime duration;
94     lessontime endTime;
95     std::string type;
96 };
97 
98 
99 #endif //PROJAED_LESSON_H
```

## 4.5 src/lessontime.h File Reference

```
#include <string>
#include <iostream>
#include <iomanip>
```

### Classes

- class lessontime

  *Class used to represent time.*

## 4.6 lessontime.h

Go to the documentation of this file.

```
1 
2 #ifndef PROJAED_LESSONTIME_H
3 #define PROJAED_LESSONTIME_H
4 
5 
6 #include <string>
7 #include <iostream>
8 #include <iomanip>
9 #include <string>
10 
11 using namespace std;
15 class lessontime {
16 public:
21     explicit lessontime(double time);
22 
26     lessontime();
27 
33     lessontime(int hour, int minutes);
34 
39     string displayHourFormat() const;
40 
45     int getHour() const;
46 
51     int getMinute() const;
52 
```

```
53     friend std::ostream &operator«(std::ostream &os, const lessontime &t);
54
55     bool operator<(const lessontime &other)const {
56         // Compare two lessontime objects based on their hours and minutes
57         if (hour < other.hour) {
58             return true;
59         } else if (hour == other.hour && minute < other.minute) {
60             return true;
61         }
62
63         return false;
64     }
65
66     bool operator==(const lessontime &other)const {
67         // Compare two lessontime objects based on their hours and minutes
68         if (hour == other.getHour() && minute == other.getMinute()) {
69             return true;
70         }
71
72         return false;
73     }
74
75     bool operator<=(const lessontime &other)const {
76         // Compare two lessontime objects based on their hours and minutes
77         return (hour < other.hour) || (hour == other.hour && minute <= other.minute);
78     }
79
80 private:
81     int hour;
82     int minute;
83 };
84
85
86 #endif //PROJAED_LESSONTIME_H
```

## 4.7 Menu.h

```
1 #ifndef PROJAED_MENU_H
2 #define PROJAED_MENU_H
3
4 #include "ControlUnit.h"
5
6 class Menu {
7 public:
8     void createMenu();
9
10     void SeeStudentSchedule();
11
12     void SeeClassSchedule();
13
14     void SeeNumStudentsAtLeastNUCs();
15
16     void SeeNumStudentsAtMostNUCs();
17
18     void SeeNumStudentsInNUCs();
19
20     void listingMenu();
21
22     void requestMenu();
23
24     void scheduleMenu();
25
26     void studentMenu();
27
28     void SeeStudentsInUc(function<bool(Student, Student)> comp);
29
30     void SeeStudentsInYear(function<bool(Student, Student)> comp);
31
32     void SeeStudentsInClass(function<bool(Student, Student)> comp);
33
34     void createRequest();
35
36     function<bool(Student, Student)> optionStudentMenu();
37
38 private:
39     ControlUnit Control;
40
41     void SeeUcFromMostStudents();
42
43
44     void SeeNumStudentsInExactNUCs();
45 };
46
```

```
47
48 #endif //PROJAED_MENU_H
```

## 4.8 RemoveRequest.h

```
1 #ifndef PROJAED_REMOVEREQUEST_H
2 #define PROJAED_REMOVEREQUEST_H
3
4 #include <string>
5 #include "Request.h"
6
7 class RemoveRequest :  public Request {
8 private:
9     std::string upCodeStudent;
10     std::string uCCode;
11     std::string classCode;
12
13
14 public:
15     // Constructor
16     RemoveRequest(const std::string &upCodeStudent, const std::string &uCCode,
17                   const std::string &classCode);
18
19     // Getters
20     std::string getUpCodeStudent() const;
21
22     std::string getUCCode() const;
23
24     std::string getClassCode() const;
25
26     // Setters
27     void setUpCodeStudent(const std::string &upCodeStudent);
28
29     void setUCCode(const std::string &uCCode);
30
31     void setClassCode(const std::string &classCode);
32 };
33
34 #endif //PROJAED_REMOVEREQUEST_H
```

## 4.9 Request.h

```
1 #ifndef PROJAED_REQUEST_H
2 #define PROJAED_REQUEST_H
3
4 #include "student.h"
5 #include "studentGroup.h"
6
7 class Request {
8 private:
9     static int count;  // Declare a static member variable for request ID.
10     int requestId; //The ID that identifies each Request
11     bool processed;
12     std::string type;
13
14 public:
15     void static setCount() {
16         count = 0;
17     }
18
19     Request(std::string type) {
20         count++;
21         requestId = count;
22         cout « "request id is " « requestId « " and count is " « count « endl;
23         processed = false;
24         this->type = type;
25     }
26
27     void setProcessed(bool processed) {
28         this->processed = processed;
29     }
30
31     std::string getType()const { return type; }
32
33     // Add a virtual function (it can be a pure virtual function).
34     virtual void dummy() {
35         //ALLOWS DOWNCASTING
36     }
37
```

```
38     virtual ~Request() {};
39 };
40
41 #endif // PROJAED_REQUEST_H
```

## 4.10 Schedule.h

```
1 #ifndef PROJAED_SCHEDULE_H
2 #define PROJAED_SCHEDULE_H
3
4 #include <vector>
5 #include "lesson.h"
6 #include <map>
7
8 using namespace std;
9
10 class Schedule {
11 private:
12     vector<lesson> lessons; //the lessons that go into the schedule
13     map<pair<int, int>, string> ScheduleMap; // a schedule is made up of 30 by 6 blocks
14 public:
15     Schedule(vector<lesson>);
16
17     void display();
18
19
20 };
21
22
23 #endif //PROJAED_SCHEDULE_H
```

## 4.11 src/student.h File Reference

```
#include <set>
#include <tuple>
#include <string>
#include "studentGroup.h"
```

### Classes

- class Student

  *Class used to represent a student.*

## 4.12 student.h

Go to the documentation of this file.

```
1
2 #ifndef PROJAED_STUDENT_H
3 #define PROJAED_STUDENT_H
4
5 #include <set>
6 #include <tuple>
7 #include <string>
8 #include "studentGroup.h"
9
10 using namespace std;
11
15 class Student {
16 public:
20     Student() = default;
21
28     Student(string studentId, string name, set<studentGroup> group);
29
```

```
34      std::string getStudentID() const;
35
40      set<studentGroup> getStudentGroups() const;
41
46      std::string getName() const;
47
52      void setName(const std::string &newName);
53
58      void setStudentID(const std::string &studentId);
59
64      void addStudentGroup(const studentGroup& GroupToAdd);
65
70      void removeGroup(const studentGroup& GroupToRemove);
71
77      bool isInUC(const string& uc) const;
78
86      bool isInClass(const string& ucCode, const string& studentGroup) const;
87
88      bool operator<(const Student &other)const {
89          return studentID < other.studentID;
90      }
91
92      bool operator==(const Student &other)const {
93          return (this->studentID == other.studentID) && (this->name == other.name);
94      }
95
96      friend std::ostream &operator«(std::ostream &os, const Student &student);
97
98  private:
99      std::string studentID;
100      std::string name;
101      std::set<studentGroup> StudentGroups;
102  };
103
104
105  #endif //PROJAED_STUDENT_H
```

## 4.13 studentGroup.h

```
1  #ifndef STUDENTGROUP_H
2  #define STUDENTGROUP_H
3
4  #include <iostream>
5  #include <string>
6
7  class studentGroup {
8  public:
9      // Constructors
10      studentGroup() = default;
11
12      studentGroup(const std::string &uccode, const std::string &classCode);
13
14      const std::string &getClassCode()const {
15          return classCode;
16      }
17
18      const std::string &getUcCode()const {
19          return UcCode;
20      }
21
22      bool operator<(const studentGroup &other)const {
23          // Define a comparison logic here based on your criteria.
24          // For example, you can compare based on class code or other fields.
25          return this->classCode + this->UcCode < other.classCode + other.UcCode;
26      }
27
28      friend std::ostream &operator«(std::ostream &os, const studentGroup &group) {
29          os « "UcCode:  " « group.UcCode « ", Class Code:  " « group.classCode;
30          return os;
31      }
32
33
34  private:
35
36      std::string classCode;
37      std::string UcCode;
38
39
40  };
41
42  #endif
```

## 4.14 SwitchRequest.h

```
1 #ifndef PROJAED_SWITCHREQUEST_H
2 #define PROJAED_SWITCHREQUEST_H
3
4 #include "Request.h"
5 #include <string>
6
7 class SwitchRequest :  public Request {
8 private:
9     std::string upCodeStudent;
10     std::string uCCode_1;
11     std::string uCCode_2;
12     std::string classCode_1;
13     std::string classCode_2;
14
15 public:
16     // Constructor
17     SwitchRequest(const std::string &upCodeStudent, const std::string &uCCode1, const std::string
      &uCCode2,
18                   const std::string &classCode1, const std::string &classCode2);
19
20     // Getters
21     std::string getUpCodeStudent() const;
22
23     std::string getUCCode1() const;
24
25     std::string getUCCode2() const;
26
27     std::string getClassCode1() const;
28
29     std::string getClassCode2() const;
30
31
32 };
33
34 #endif // PROJAED_SWITCHREQUEST_H
```

# Index