Gliwice, 18.06.2016r.

Laboratorium Programowania Komputerów 4

Temat: System dziekanat

Autor: Roksana Paluch
Informatyka, rok 2015/2016, semestr 4,grupa1
Prowadzący: dr inż. Jolanta Kawulok

1.Temat

Celem projektu „System dziekanat” jest obsługa planów lekcyjnych studiów. Za pomocą programu można dodać osobne semestry, ustalając ich długość, oraz je usuwać. Po dodaniu semestru można do niego dodawać poszczególne plany dla grup, nauczycieli oraz sal. Dla każdej z tych grup można dodawać zajęcia lekcyjne dla interesującego nas tygodnia mieszczącego się w podanym wcześniej zakresie semestru, można je także usuwać. Przy usuwaniu zajęcie lekcyjne jest automatycznie usuwane ze wszystkich planów go dotyczących. Przy wstawianiu nowych zajęć sprawdzane jest, czy dodanie zajęcia jest możliwe nie tylko dla danego planu, ale także biorąc pod uwagę pozostałe plany. Plany dla poszczególnych tygodni można przeglądać.

Dane zapisywane do programu zapisywane są do pliku, a przy ponownym uruchomieniu ponownie załadowane. Możliwy jest także zapis planu dla danego tygodnia do pliku określonego przez użytkownika programu.

W programie realizowane są zagadnienia z zajęć tematycznych jak: wyjątki, iteratory, kontenery STL oraz polimorfizm bez RTTI.

2.Analiza tematu

Program wykonany jest obiektowo, strukturami w nim użytymi są listy i wektory pochodzące z biblioteki STL. Listy użyte są do struktur, przy których często może zachodzić zmiana, dodawanie lub usuwanie nowych elementów, jak np. przechowywane semestry czy poszczególne plany oraz zajęcia lekcyjne w ciągu dnia. Natomiast do przechowywania obiektów, których rozmiar jest stały i nie zmienia się w ciągu wykonywania programu zastosowałam wektory, są to np. tygodnie w poszczególnych planach czy liczba dni w ciągu danego tygodnia.

Program kontroluje wprowadzane przez użytkownika dane, takie jak czy semestr o takiej samej nazwie już istnieje, czy taka sala/nauczyciel/grupa w danym semestrze już istnieje, czy daty rozpoczęcia i zakończenia semestru są prawidłowe, czy godziny rozpoczęcia i zakończenia lekcji są prawidłowe, czy wprowadzona data/godzina jest prawidłowa, czy lekcje w danym dniu nie zachodzą na siebie. Dane te nie są sprawdzane podczas wczytywania poprzedniego stanu programu z pliku, zakładając, że zapisywane dane były poprawne, dlatego nie zaleca się modyfikacji pliku przechowującego dane między następnymi uruchomieniami programu, ponieważ może to spowodować błąd.

3.Specyfikacja zewnętrzna

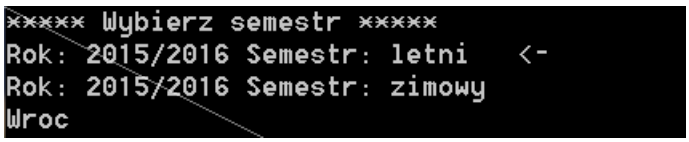
Program komunikuje się z użytkownikiem za pomocą okna konsoli. Po włączeniu programu i ewentualnym wczytaniu poprzedniego stanu programu z pliku, powinno wyświetlić się główne menu:

Il.1. Główne menu programu

Poruszać po programie można się za pomocą górnej i dolnej strzałki na klawiaturze. Wciśnięcie klawisza Enter spowoduje wybranie opcji przy której aktualnie znajduje się strzałka.

Przy pierwszym uruchomieniu programu jedyną czynnością, którą będzie mógł wykonać użytkownik jest dodanie nowego semestru i wyjście z programu. Po wejściu w opcje wybierz semestr i usuń semestr jedyna dostępną operacją będzie powrót do menu głównego.

Gdy w programie będzie już się znajdował przynajmniej jeden semestr, po wyborze usunięcia semestru lub wyboru semestru wyświetlą się dostępne semestry:

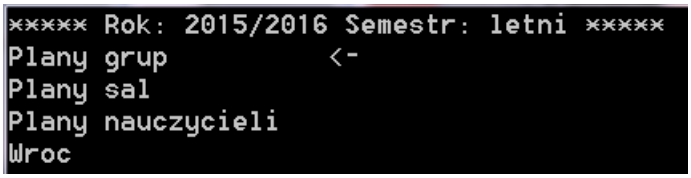


Il.2. Wybór semestru

Gdy wybraliśmy usunięcie semestru, po wybraniu danego semestru i wciśnięciu klawisza Enter pojawi się informacja o usunięciu semestru:

Il.3. Usunięcie semestru

Przy wyborze semestru i po wciśnięciu klawisza Enter pojawią się do wyboru plany grup, sal i nauczycieli:

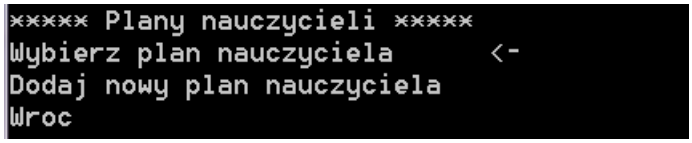


Il.4. Opcje dostępne po wyborze semestru

Po wybraniu jednej opcji pojawi się wybór czy chcemy wybrać czy dodać nowy odpowiednio plan grupy/sali/nauczyciela :

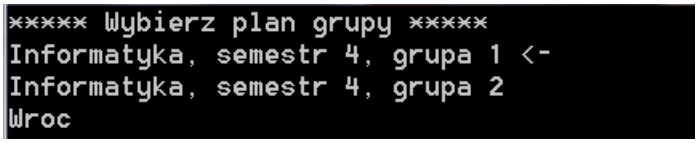
Il.5. Menu planów grup

Il.6. Menu planów sal



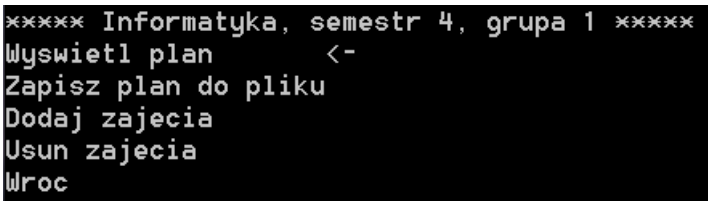
Il.7. Menu planów nauczycieli

Po wybraniu opcji wybrania planu/sali/nauczyciela, pojawią się dostępne obecnie plany. Jeśli nie ma wprowadzonych żadnych planów, będzie możliwe wybranie jedynie opcji wróć:



Il.8. Wybór odpowiedniego planu danego rodzaju

Po wybraniu planu wyświetlą się działania, które będzie można na nim wykonać:



Il.9. Działania możliwe do wykonania dla danego planu

Przy wyborze wyświetlenia planu, zapisaniu planu do pliku, dodaniu zajęć czy usunięciu, najpierw użytkownik będzie musiał wybrać, dla którego tygodnia chce wykonać daną czynność:

```
***** Wybierz tydzien *****
05.03.2016 - 06.03.2016 <-
07.03.2016 - 13.03.2016
14.03.2016 - 20.03.2016
21.03.2016 - 27.03.2016
28.03.2016 - 03.04.2016
04.04.2016 - 10.04.2016
11.04.2016 - 17.04.2016
18.04.2016 - 24.04.2016
25.04.2016 - 01.05.2016
02.05.2016 - 08.05.2016
09.05.2016 - 15.05.2016
16.05.2016 - 22.05.2016
23.05.2016 - 29.05.2016
30.05.2016 - 05.06.2016
06.06.2016 - 12.06.2016
13.06.2016 - 19.06.2016
20.06.2016 - 20.06.2016
Wroc
```

Il.10. Wybór tygodnia dla którego chcemy wykonać daną akcję

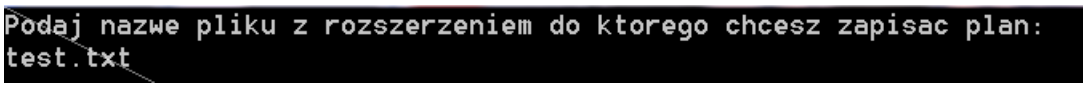
Dla wyświetlenia planu po wyborze tygodnia wyświetli się odpowiedni plan:

```
Informatyka, semestr 4, grupa 1
--------------------
Poniedzialek
--------------------
OR
wyklad
08:15 - 09:45
1
Zbigniew Czechowicz

BD
wyklad
10:00 - 13:15
1
Stanislaw Koziol

JA
wyklad
13:30 - 15:00
1
Krzysztof Komin

SO
wyklad
15:45 - 17:15
2
Bartosz Pach

--------------------
Wtorek
--------------------
JO
cwiczenia
08:30 - 10:00
545
Katarzyna Tur
```

Il.11. Wyświetlenie przykładowego planu

Po naciśnięciu dowolnego przycisku powrócimy do poprzedniego menu, w którym wybieraliśmy tydzień.

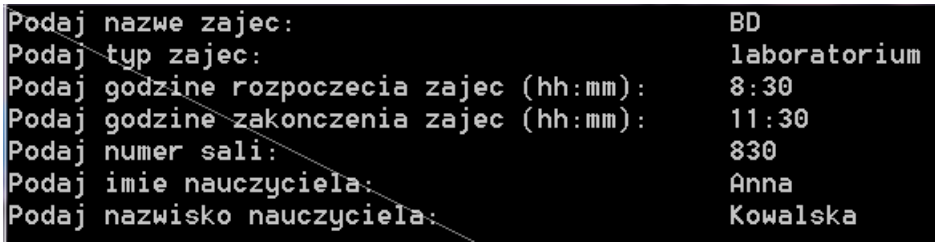
Dla zapisania planu po wybraniu tygodnia wyświetli się komunikat z prośbą o podanie nazwy pliku do którego chcemy zapisać plan:



Il.12. Wybór nazwy pliku dla zapisania planu.

Po zapisaniu wyświetli się komunikat o wykonanej czynności, aby powrócić do poprzedniego menu należy nacisnąć dowolny klawisz.

Dla dodania zajęć program będzie prosił o wpisanie kolejno wszystkich właściwości zajęcia. Właściwości dla grup, sal i nauczycieli różnią się o dane, dla których program wie dla której grupy/sali/nauczyciela ma dodać lekcję. Przykładowe wprowadzanie zajęć:



Il.13. Wprowadzanie zajęć lekcyjnych

Podczas wprowadzania zajęć program może wykryć i zasygnalizować błędy o złej podanej godzinie (wychodzącej poza zakres czasu, np. 25:00) oraz o tym, że godzina rozpoczęcia jest mniejsza lub równa godzinie zakończenia. Po sygnalizacji błędu i wciśnięciu dowolnego przycisku wracamy do poprzedniego menu.

Po poprawnym wprowadzeniu właściwości zajęcia użytkownik powinien wybrać dzień tygodnia, do którego chce dodać zajęcia:

Il.14. Wybór dnia tygodnia

Po wyborze dnia może pojawić się informacja o poprawnym dodaniu lekcji albo o błędzie. W przypadku błędu program informuje dla którego z trzech planów nie można było dodać lekcji i co było tego przyczyną – zbyt wczesne rozpoczęcie lub zbyt późne zakończenie. W tym przypadku lekcja nie zostanie dodana do żadnego z planów. Po wyświetleniu komunikatu użytkownik powinien nacisnąć dowolny przycisk, by powrócić do poprzedniego menu.

W przypadku usuwania zajęć, dane które chce od nas pobrać program są identyczne jak podczas dodawania zajęć. Po wprowadzeniu wszystkich danych program wyświetli informację o usunięciu danej lekcji, albo że taka lekcja nie istnieje.

Przy dodawaniu nowego semestru program będzie prosić użytkownika o podanie szczegółów nowego semestru:

Il.15. Dodawanie nowego semestru

Program sprawdza, czy semestr o danym roku i nazwie już nie istnieje, w przypadku istnienia wyświetli o tym informację i nie doda nowego planu. Sprawdzane jest także, czy data końcowa semestru nie jest mniejsza od jego daty początkowej. W takim przypadku również informacja zostanie wypisana na ekranie. Po dodaniu/błędzie i naciśnięciu dowolnego klawisza wracamy do menu głównego i można przejść do wybrania semestru lub wprowadzenia nowego jeszcze raz. Po wyjściu z programu obecny stan zapisywany jest do pliku, aby przy ponownym uruchomieniu można było go wczytać i pracować na wprowadzonych poprzednio danych.

4.Specyfikacja wewnętrzna

W celu zrealizowania projektu stworzyłam następujące struktury danych:

Klasa Communicate

```
class Communicate
{
	std::string nameOfFile;
	std::fstream dataFile;
	std::list<Synchronize> schedules;
public:
	Communicate(std::string nameOfFile);
	~Communicate();
	int startingMenu();
	void startingChoice();
	std::list<Synchronize>::iterator selectSynchronize();
	void addSchedule();
	void getSemester(std::fstream &dataFile);
	void getDays(std::fstream &dataFile, std::list<Synchronize>::iterator iter_schedule);
	void getLessons(std::fstream &dataFile, std::list<Synchronize>::iterator
iter_schedule,Date date);
};
```

Klasa Communicate jest główną klasą programu, której obiekt tworzony jest na samym początku uruchomienia. Zajmuje się częścią komunikacji z użytkownikiem. Odczytuje także z pliku poprzedni stan programu do struktur danych i po zakończeniu ponownie zapisuje nowy stan. Zawiera listę z poszczególnymi semestrami.

Zmienne klasy:

`std::string nameOfFile;`	Nazwa pliku, w którym przechowywany jest stan programu.
`std::fstream dataFile;`	Obiekt umożliwiający operację na wybranym pliku.
`std::list<Synchronize> schedules;`	Lista zawierająca poszczególne semestry.

Wybrane funkcje klasy:

`Communicate(std::string nameOfFile);`	Konstruktor obiektu przyjmujący nazwę pliku z którego odczytywany/zapisywany będzie stan programu. Gdy plik istnieje, dopóki nie napotkamy na jego koniec odczytujemy z niego dane do programu korzystając z funkcji `getSemester(std::fstream &dataFile)` pochodzącej z tej samej klasy.
`~Communicate();`	Destruktor obiektu. Usuwa informacje z pliku z poprzednim stanem programu i zapisuje nowy stan, przechodząc po kolejnych semestrach korzystając z funkcji `saveDataFile(std::fstream &dataFile)`pochodzącej z klasy Synchronize.

`int startingMenu();`	Funkcja odpowiadająca za wyświetlenie menu głównego programu. Zwraca numer wybranej przez użytkownika opcji.
`void startingChoice();`	Główna funkcja komunikacji z użytkownikiem. Wywołuje funkcję `startingMenu()` i za pomocą otrzymanego numeru opcji przechodzi do wybierania semestru, dodawania semestru, usuwania semestru lub wychodzi z programu.
`std::list<Synchronize>::iterator selectSynchronize();`	Funkcja odpowiadająca za wyświetlenie dostępnych semestrów. Po wybraniu semestru przez użytkownika zwraca na niego iterator do funkcji `startingChoice()`. W przypadku wybrania opcji wróć zwraca iterator wskazujący na element za ostatnim elementem listy.
`void addSchedule();`	Funkcja pobierająca od użytkownika dane na temat nowego semestru i dodająca go do listy. Jeśli użytkownik doda złe dane, informuje o występującym błędzie i kończy swoje działanie.
`void getSemester(std::fstream &dataFile);`	Funkcja pobierająca z pliku ze stanem programu dane o semestrze i dodająca pobrany semestr do listy, a następnie wywołująca funkcję `getDays(std::fstream &dataFile, std::list<Synchronize>::iterator iter_schedule)`, by dodać do semestru dni.
`void getDays(std::fstream &dataFile, std::list<Synchronize>::iterator iter_schedule);`	Funkcja pobierająca z pliku ze stanem programu dane o dniu i wywołująca dla każdego dnia funkcję `getLessons(std::fstream &dataFile, std::list<Synchronize>::iterator iter_schedule,Date date)`, by dodać do dnia lekcje.
`void getLessons(std::fstream &dataFile, std::list<Synchronize>::iterator iter_schedule,Date date);`	Funkcja pobierająca z pliku ze stanem programu dane o kolejnych lekcjach i wywołująca funkcję `addLesson(Date date, Lesson lesson)` z klasy Synchronize, aby je dodać.

Klasa Synchronize

```
class Synchronize
{
	std::list<ClassroomSchedule> classrooms;
	std::list<GroupSchedule> groups;
	std::list<TeacherSchedule> teachers;
	Date startingDate;
	Date endingDate;
	std::string semester;
	std::string academicYear;
public:
	Synchronize(Date startingDate, Date endingDate, std::string semester, std::string
academicYear);
	~Synchronize();
	void information();
	friend std::ostream &operator<<(std::ostream &s, const Synchronize &synchronize);
	void saveDataFile(std::fstream &dataFile);
	static bool compare(Synchronize s1, Synchronize s2);
	void selectChoice();
	int selectMenu();
	std::list<GroupSchedule>::iterator selectGroup();
```

```
        std::list<ClassroomSchedule>::iterator selectClassroom();
        std::list<TeacherSchedule>::iterator selectTeacher();
        int groupMenu();
        int classroomMenu();
        int teacherMenu();
        void groupOption();
        void classroomOption();
        void teacherOption();
        void addGroup();
        void addClassroom();
        void addTeacher();
        bool existGroup(GroupSchedule group);
        bool existClassroom(ClassroomSchedule classroom);
        bool existTeacher(TeacherSchedule teacher);
        void selectScheduleChoice(std::list<GroupSchedule>::iterator group);
        void selectScheduleChoice(std::list<ClassroomSchedule>::iterator classroom);
        void selectScheduleChoice(std::list<TeacherSchedule>::iterator teacher);
        void addLesson(std::list<GroupSchedule>::iterator group, std::vector<Week>::iterator
week);
        void addLesson(std::list<ClassroomSchedule>::iterator classroom,
std::vector<Week>::iterator week);
        void addLesson(std::list<TeacherSchedule>::iterator teacher,
std::vector<Week>::iterator week);
        void deleteLesson(std::list<GroupSchedule>::iterator group,
std::vector<Week>::iterator week);
        void deleteLesson(std::list<ClassroomSchedule>::iterator classroom,
std::vector<Week>::iterator week);
        void deleteLesson(std::list<TeacherSchedule>::iterator teacher,
std::vector<Week>::iterator week);
        bool operator==(Synchronize s);
        void addLesson(Date date, Lesson lesson);
};
```

Klasa Synchronize jest drugą najważniejszą po klasie Communicate klasą programu. Zajmuje się częścią komunikacji z użytkownikiem w dalszych obszarach programu niż Communicate, związanego z obsługą istniejącego już semestru. Jej zadaniem jest także synchronizacja planów grup, nauczycieli i sal dla danego semestru, pilnując by zajęcia lekcyjne zostały poprawnie dodane/usunięte dla wszystkich tych planów. Każdy plan zawiera wszystkie wprowadzone zajęcia.

Zmienne klasy:

`std::list<ClassroomSchedule> classrooms;`	Lista przechowująca plany sal.
`std::list<GroupSchedule> groups;`	Lista przechowująca plany grup.
`std::list<TeacherSchedule> teachers;`	Lista przechowująca plany nauczycieli.
`Date startingDate;`	Data rozpoczęcia semestru.
`Date endingDate;`	Data zakończenia semestru.
`std::string semester;`	Nazwa semestru.
`std::string academicYear;`	Nazwa roku akademickiego.

Wybrane funkcje klasy:

`Synchronize(Date startingDate, Date endingDate, std::string semester, std::string academicYear);`	Konstruktor tworzący obiekt klasy. Jeśli data zakończenia semestru jest mniejsza od daty rozpoczęcia, rzuca wyjątek `WrongScheduleException()`.
`friend std::ostream`	Operator wypisujący do strumienia informacje o

`&operator<<(std::ostream &s, const Synchronize &synchronize);`	semestrze: datę rozpoczęcia, zakończenia, rok akademicki i semestr.
`void saveDataFile(std::fstream &dataFile);`	Funkcja służąca do zapisania do pliku ze stanem programu kolejnych planów w semestrze. Wykorzystana jest tu lista z planami klas, ponieważ w każdej liście przechowywane są takie same informacje, wystarczy zapisać je tylko raz.
`void selectChoice();`	Funkcja wywołująca funkcję `selectMenu()`, dostaje jako informację zwrotną numer opcji, która została wybrana przez użytkownika i wywołuje funkcję odpowiadającą opcji lub zakańcza swoje działanie.
`int selectMenu();`	Funkcja komunikująca się z użytkownikiem, wyświetla do wyboru opcję planów grup, sal i nauczycieli oraz powrót. Po wybraniu przez użytkownika opcji zwraca ją do funkcji `selectChoice()`.

`std::list<GroupSchedule>::iterator selectGroup();`
`std::list<ClassroomSchedule>::iterator selectClassroom();`
`std::list<TeacherSchedule>::iterator selectTeacher();`
	Funkcje komunikujące się z użytkownikiem, wyświetlające odpowiednio dostępne plany grup/sal/nauczycieli, po wyborze użytkownika zwracają iterator na wybrany plan lub iterator na element za ostatnim elementem, gdy użytkownik wybrał opcję wróć.

`int groupMenu();`
`int classroomMenu();`
`int teacherMenu();`
	Funkcje komunikujące się z użytkownikiem, wyświetlające opcje wybrania i dodania planu grupy/sali/nauczyciela lub powrotu i zwracające numer wybranej opcji.

`void groupOption();`
`void classroomOption();`
`void teacherOption();`
	Funkcje wywołujące odpowiednio wybraną przez użytkownika opcję zwróconą przez funkcję `groupMenu()/classroomMenu()/teacherMenu()`, czyli przejście do funkcji wyświetlającej dostępne plany, a następnie do funkcji wyświetlającej działania na planie, dodanie nowego planu lub powrót.

`void addGroup();`
`void addClassroom();`
`void addTeacher();`
	Funkcje komunikujące się z użytkownikiem, pobierające od niego informacje na temat dodawanego nowego planu i dodanie go, jeśli nie istnieje lub wyświetlenie komunikatu o istnieniu.

`bool existGroup(GroupSchedule group);`
`bool existClassroom(ClassroomSchedule classroom);`
`bool existTeacher(TeacherSchedule teacher);`
	Funkcje sprawdzające w odpowiedniej liście, czy podany jako argument plan już istnieje, zwracają true gdy istnieje, false jeśli nie istnieje.

`void selectScheduleChoice(std::list<GroupSchedule>::iterator group);`
`void selectScheduleChoice(std::list<ClassroomSchedule>::iterator classroom);`
`void selectScheduleChoice(std::list<TeacherSchedule>::iterator teacher);`
	Funkcje, które na podstawie wybranej przez użytkownika opcji wywołują funkcje odpowiedzialne za wyświetlenie planu, zapisanie planu do pliku, dodanie do planu nowych zajęć lub usunięcie ich oraz powrót do poprzedniego menu.

`void`	Funkcje wywołujące funkcję zwracającą lekcję

`addLesson(std::list<GroupSchedule>::iterator group, std::vector<Week>::iterator week);`
`void addLesson(std::list<ClassroomSchedule>::iterator classroom, std::vector<Week>::iterator week);`
`void addLesson(std::list<TeacherSchedule>::iterator teacher, std::vector<Week>::iterator week);`
	wprowadzoną przez użytkownika, dodające ją do wszystkich planów z lekcją powiązanych, zaczynając od tego, dla którego były wywołane. Jeśli jakiś plan do którego mają dodać nie istnieje, wywołują funkcję tworzącą go. Informują o złych parametrach otrzymanej lekcji oraz gdy nie można dodać lekcji do któregoś z planów, wtedy usuwają ja z planów, do których lekcja została już dodana, by nie powstały niespójności.

`void deleteLesson(std::list<GroupSchedule>::iterator group, std::vector<Week>::iterator week);`
`void deleteLesson(std::list<ClassroomSchedule>::iterator classroom, std::vector<Week>::iterator week);`
`void deleteLesson(std::list<TeacherSchedule>::iterator teacher, std::vector<Week>::iterator week);`
	Funkcje usuwające z wszystkich planów zaczynając od obecnego pobraną od użytkownika lekcję. Informują o złych parametrach otrzymanej lekcji do usunięcia, oraz gdy lekcja którą chcemy usunąć nie istnieje. Ponieważ przy dodawaniu zawsze dodajemy lekcję do wszystkich planów, jeśli lekcja istnieje w jednym planie, zakładamy, że istnieje również w pozostałych.

`void addLesson(Date date, Lesson lesson);`	Funkcja używana przy wczytywaniu do programu danych z pliku z poprzednim stanem programu. Dodaje podaną jako argument lekcję do wszystkich list z planami, jeśli jakiś plan nie istnieje wywołuje najpierw funkcję tworzącą go.

Klasa Schedule

```
class Schedule
{
	Date startingDate;
	Date endingDate;
protected:
	std::vector<Week> weeks;
public:
	virtual int information(std::ostream &s) = 0;
	Schedule(Date startingDate, Date endingDate);
	virtual ~Schedule();
	int selectMenu();
	std::vector<Week>::iterator weekMenu();
	void saveSchedule(std::vector<Week>::iterator week);
	std::vector<Week>::iterator getEndIterator();
	virtual Lesson getLesson() = 0;
	void addLesson(Date startWeek, int day, Lesson lesson);
	void deleteLesson(Date startWeek, int day, Lesson lesson);
	void addLesson(Date date, Lesson lesson);
	void saveDataFile(std::fstream &dataFile);
	int size();
};
```

Abstrakcyjna klasa bazowa po której dziedziczą klasy GroupSchedule, ClassroomSchedule i TeacherSchedule. Zawiera kilka funkcji czysto wirtualnych implementowanych przez dziedziczące ją klasy. Zajmuje się obsługą całych tygodni zapisanych w semestrze, zapisanych w wektorze, który jest dostępny dla klas dziedziczących.

Zmienne klasy:

`Date startingDate;`	Data rozpoczęcia semestru, początek pierwszego tygodnia.
`Date endingDate;`	Data zakończenia semestru, koniec ostatniego tygodnia.
`std::vector<Week> weeks;`	Wektor przechowujący poszczególne tygodnie.

Wybrane funkcje klasy:

`Schedule(Date startingDate, Date endingDate);`	Konstruktor klasy. Na podstawie podanej początkowej i końcowej daty tworzy poszczególne tygodnie zaczynające się na poniedziałku i kończące na niedzieli. Jeśli początkowa data nie jest poniedziałkiem, pierwszy tydzień będzie krótszy, tak samo jeśli końcowa data nie jest niedzielą.
`int selectMenu();`	Funkcja komunikująca się z użytkownikiem, wyświetlająca opcje dla danego planu, zwracająca numer wybranej opcji.
`std::vector<Week>::iterator weekMenu();`	Funkcja komunikująca się z użytkownikiem, wyświetlająca początkową i końcową datę każdego tygodnia, zwraca iterator na tydzień wybrany przez użytkownika.
`void saveSchedule(std::vector<Week>::iterator week);`	Funkcja komunikująca się z użytkownikiem, dla przekazanego iteratora na dany tydzień, zapisuje plan do pliku o nazwie podanej przez użytkownika.
`void addLesson(Date startWeek, int day, Lesson lesson);`	Funkcja zapisująca przekazaną jako argument lekcję do tygodnia o podanej jako argument początkowej jego dacie, dla dnia o numerze podanym jako argument. Znajduje odpowiedni tydzień i dzień, następnie dodaje lekcję. Jeśli dodanie nie powiedzie się, rzuca otrzymany wyjątek `AddingLessonException.`
`void deleteLesson(Date startWeek, int day, Lesson lesson);`	Funkcja wyszukująca tydzień o początkowej dacie podanej jako argument, następnie w nim wyszukuje dzień o podanym numerze. Próbuje dla danego dnia usunąć podaną lekcję, jeśli usunięcie nie powiedzie się, rzuca otrzymany wyjątek `SearchingLessonException.`
`void addLesson(Date date, Lesson lesson);`	Funkcja wyszukuje w tygodniach dzień o podanej dacie, a następnie dodaje do wyszukanego dnia podaną lekcję. Używana przy wczytywaniu do programu danych z pliku z poprzednim stanem programu, nie uwzględnia możliwości wyjątku `AddingLessonException.`
`void saveDataFile(std::fstream &dataFile);`	Funkcja używana przy zapisywaniu obecnego stanu programu do pliku, wywołuje funkcję zapisującą dane dla każdego ze swoich tygodni.

Klasa GroupSchedule

```
class GroupSchedule :
	public Schedule
{
	int groupNumber;
	int groupSemester;
	std::string groupSpecialization;
public:
	GroupSchedule(Date startingDate, Date endingDate, int groupNumber, int groupSemester,
std::string groupSpecialization);
	~GroupSchedule();
	void writeWeek(Date start, Date end);
	int information(std::ostream &s);
	bool operator==(GroupSchedule group);
	static bool compare(GroupSchedule g1, GroupSchedule g2);
	Lesson getLesson();
};
```

Klasa dziedzicząca publicznie po klasie Schedule. Reprezentuje plan grupy, więc dodatkowo zawiera zmienne identyfikujące daną grupę oraz implementuje odziedziczone metody czysto wirtualne.

Zmienne klasy:

`int groupNumber;`	Zmienna przechowująca numer grupy.
`int groupSemester;`	Zmienna przechowująca semestr grupy.
`std::string groupSpecialization;`	Zmienna przechowująca kierunek grupy.

Wybrane funkcje klasy:

`int information(std::ostream &s);`	Wypisuje do podanego strumienia informacje o danej grupie, zwraca liczbę 1 informującą o tym, że jest to funkcja klasy GroupSchedule.
`Lesson getLesson();`	Pobiera od użytkownika dane o lekcji, jak nazwa zajęć, typ zajęć, godziny rozpoczęcia i zakończenia, numer sali oraz imię i nazwisko nauczyciela. W przypadku źle podanej godziny lub gdy godzina zakończenia jest wcześniejsza lub równa godzinie rozpoczęcia, rzucany jest wyjątek, odpowiednio `WrongTimeException` lub `WrongLessonException`.

Klasa ClassroomSchedule

```
class ClassroomSchedule :
	public Schedule
{
	int classroom;
public:
	void writeWeek(Date start, Date end);
	ClassroomSchedule(Date startingDate, Date endingDate, int classroom);
	~ClassroomSchedule();
	int information(std::ostream &s);
	bool operator==(ClassroomSchedule classroom);
```

```
	static bool compare(ClassroomSchedule c1, ClassroomSchedule c2);
	Lesson getLesson();
};
```

Klasa dziedzicząca publicznie po klasie Schedule. Reprezentuje plan sali, więc dodatkowo zawiera zmienne identyfikujące daną salę oraz implementuje odziedziczone metody czysto wirtualne.

Zmienne klasy:

`int classroom;`	Zmienna przechowująca numer sali.

Wybrane funkcje klasy:

`int information(std::ostream &s);`	Wypisuje do podanego strumienia informacje o danej klasie, zwraca liczbę 2, informującą o tym, że została wywołana funkcja klasy ClassroomSchedule.
`Lesson getLesson();`	Pobiera od użytkownika dane o lekcji, jak nazwa zajęć, typ zajęć, godziny rozpoczęcia i zakończenia, imię i nazwisko nauczyciela oraz kierunek grupy, semestr grupy i numer grupy.. W przypadku źle podanej godziny lub gdy godzina zakończenia jest wcześniejsza lub równa godzinie rozpoczęcia, rzucany jest wyjątek, odpowiednio `WrongTimeException` lub `WrongLessonException`.

Klasa TeacherSchedule

```
class TeacherSchedule :
	public Schedule
{
	std::string teacherName;
	std::string teacherSurname;
public:
	TeacherSchedule(Date startingDate, Date endingDate, std::string teacherName,
std::string teacherSurname);
	~TeacherSchedule();
	void writeWeek(Date start, Date end);
	int information(std::ostream &s);
	bool operator==(TeacherSchedule teacher);
	static bool compare(TeacherSchedule t1, TeacherSchedule t2);
	Lesson getLesson();
};
```

Klasa dziedzicząca publicznie po klasie Schedule. Reprezentuje plan nauczyciela, więc dodatkowo zawiera zmienne identyfikujące danego nauczyciela oraz implementuje odziedziczone metody czysto wirtualne.

Zmienne klasy:

`std::string teacherName;`	Zmienna przechowująca imię nauczyciela.
`std::string teacherSurname;`	Zmienna przechowująca nazwisko nauczyciela.

Wybrane funkcje klasy:

`int information(std::ostream &s);`	Wypisuje do podanego strumienia informacje o danej klasie, zwraca liczbę 3, informującą o tym, że została wywołana funkcja klasy TeacherSchedule.
`Lesson getLesson();`	Pobiera od użytkownika dane o lekcji, jak nazwa zajęć, typ zajęć, godziny rozpoczęcia i zakończenia, numer sali oraz kierunek grupy, semestr grupy i numer grupy.. W przypadku źle podanej godziny lub gdy godzina zakończenia jest wcześniejsza lub równa godzinie rozpoczęcia, rzucany jest wyjątek, odpowiednio `WrongTimeException` lub `WrongLessonException`.

Klasa Week

```
class Week
{
	std::vector <Day> weekDays;
	Date startingDate;
	Date endingDate;

public:
	std::vector<Day>::iterator getDay(int number);
	Week(Date startingDate, Date endingDate);
	~Week();
	void information(std::ostream &s);
	void displayWeek(std::ostream &s,int schedule);
	int daysMenu();
	Date getStart();
	Date getEnd();
	void saveDataFile(std::fstream &dataFile);
};
```

Klasa ta odpowiada za przechowywanie dni danego tygodnia jako wektor i odpowiada za operacje na poszczególnych dniach w danym tygodniu. Największy możliwy rozmiar to siedem dni, ale może zawierać mniej, jeśli podany zostanie mniejszy zakres między datą początkową i końcową tygodnia.

Zmienne klasy:

`std::vector <Day> weekDays;`	Lista przechowująca poszczególne dni tygodnia.
`Date startingDate;`	Data pierwszego dnia w tygodniu.
`Date endingDate;`	Data ostatniego dnia w tygodniu.

Wybrane funkcje klasy:

`std::vector<Day>::iterator getDay(int number);`	Funkcja wyszukuje w tygodniu dzień o podanym numerze z zakresu 0-6, każdy odpowiadający dniu tygodnia zaczynając od poniedziałku. Jeśli dzień nie zostanie znaleziony, rzucony zostanie wyjątek `WrongWeekException(0,1)`. W przeciwnym przypadku zwracany jest iterator na dzien o podanym numerze.

`Week(Date startingDate, Date endingDate);`	Konstruktor, w którym do tygodnia dodawane są kolejne dni z kolejnymi datami, zaczynając od daty początkowej, a kończąc na dacie końcowej włącznie. Jeśli długość tygodnia przekracza siedem dni lub koniec tygodnia jest mniejszy od początku zostaje rzucony wyjątek `WrongWeekException(1,0)`.
`void displayWeek(std::ostream &s,int schedule);`	Funkcja, która dla każdego dnia wywołuje funkcję wyświetlenia zajęć, odpowiednią dla odpowiedniego planu określonego przez argument schedule. Wykorzystywana przy wyświetlaniu planu na ekranie oraz zapisywaniu planu do pliku.
`int daysMenu();`	Funkcja komunikująca się z użytkownikiem, wyświetlająca do wyboru nazwy dni tygodnia dla danego tygodnia. Po wyborze przez użytkownika zwraca numer wybranej opcji.
`void saveDataFile(std::fstream &dataFile);`	Funkcja wykorzystywana przy zapisywaniu obecnego stanu programu do pliku. Dla każdego dnia, który zawiera jakieś lekcje wywołuje funkcję zapisującą te lekcje do pliku.

Klasa Day

```
class Day
{
	Date date;
	std::list<Lesson> lessonsList;
public:
	Day(Date date);
	~Day();
	void addLesson(Lesson lesson);
	std::list<Lesson>::iterator searchLesson(Lesson lesson);
	void deleteLesson(Lesson lesson);
	int getWeekDay();
	std::string getWeekDayName();
	void displayDay(std::ostream &s, int schedule);
	bool isEmpty();
	Date getDate();
	void saveDataFile(std::fstream &dataFile);
};
```

Klasa ta reprezentuje pojedynczy dzień, zawiera w sobie listę z lekcjami, które odbywają się w danym dniu. Obsługuje dany dzień, umożliwiając dodawanie, wyszukiwanie oraz usuwanie zajęć z pojedynczego dnia, gdy któraś z operacji nie powiedzie się, rzuca odpowiednie wyjątki.

Zmienne klasy:

`Date date;`	Data danego dnia.
`std::list<Lesson> lessonsList;`	Lista przechowująca lekcje odbywające się w danym dniu.

Wybrane funkcje klasy:

`void addLesson(Lesson lesson);`	Funkcja najpierw sprawdza przeglądając kolejne

	zapisane lekcje, czy podaną jako argument lekcję da się wstawić. Jeśli lista przechowująca lekcje była pusta, lekcja jest wstawiana od razu. Jeśli znalezione zostanie miejsce dla lekcji, gdzie godzina rozpoczęcia i zakończenia nie będą kolidowały z innymi zajęciami, lekcja zostaje wstawiona w tym miejscu. Jeśli wstawianie lekcji nie powiedzie się, rzucany jest wyjątek `AddingLessonException` z odpowiednimi parametrami określającymi, co przeszkodziło we wstawianiu lekcji.
`std::list<Lesson>::iterator searchLesson(Lesson lesson);`	Funkcja wyszukująca podanej jako argument lekcji. Jeśli lista z lekcjami jest pusta, lub lekcja nie została znaleziona, rzucany jest wyjątek `SearchingLessonException`. Jeśli lekcja zostanie znaleziona, zwracany jest iterator na nią.
`void deleteLesson(Lesson lesson);`	Funkcja usuwająca podany jako argument element. Najpierw wyszukuje go, gdy nie znaleziono elementu, rzuca złapany wyjątek `SearchingLessonException.` Jeśli udało się znaleźć, lekcja zostaje usunięta.
`void displayDay(std::ostream &s, int schedule);`	Funkcja przekazuje do strumienia podanego jako argument najpierw dzień tygodnia danego dnia, a następnie wypisuje wszystkie lekcje danego dnia. Wykorzystywana przy wyświetlaniu planu i zapisie planu do pliku.
`void saveDataFile(std::fstream &dataFile);`	Funkcja wykorzystywana przy zapisie do pliku aktualnego stanu programu. Wypisuje informację o wszystkich lekcjach odbywających się danego dnia.

Klasa Lesson

```
class Lesson
{
	std::string name;
	std::string type;
	Time start;
	Time end;
	int classroom;
	std::string teacherName;
	std::string teacherSurname;
	int groupNumber;
	int groupSemester;
	std::string groupSpecialization;
public:
	Lesson(std::string name, std::string type, Time start, Time end, int classroom,
std::string teacherName, std::string teacherSurname,
		int groupNumber, int groupSemester, std::string groupSpecialization);
	~Lesson();
	Time startTime();
	Time endTime();
	bool operator==(Lesson lesson);
	friend std::ostream & operator<<(std::ostream &s, const Lesson &l);
	void write(std::ostream &s,int schedule);
```

```
        int getClassroom();
        std::string getTeacherName();
        std::string getTeacherSurname();
        int getGroupNumber();
        int getGroupSemester();
        std::string getGroupSpecialization();
};
```

Klasa ta reprezentuje pojedynczą lekcję. Zawiera w sobie wszystkie niezbędne informacje na temat danej lekcji, typ oraz nazwę zajęć, godziny rozpoczęcia i zakończenia oraz informacje o grupie, sali i nauczycielu, dla których lekcja została dodana.

Zmienne klasy:

`std::string name;`	Zmienna przechowująca nazwę lekcji (nazwę przedmiotu).
`std::string type;`	Zmienna przechowująca typ lekcji (np. wykład, ćwiczenia, laboratoria).
`Time start;`	Zmienna przechowująca godzinę rozpoczęcia lekcji.
`Time end;`	Zmienna przechowująca godzinę zakończenia lekcji.
`int classroom;`	Zmienna przechowująca numer sali.
`std::string teacherName;`	Zmienna przechowująca imię nauczyciela.
`std::string teacherSurname;`	Zmienna przechowująca nazwisko nauczyciela.
`int groupNumber;`	Zmienna przechowująca numer grupy.
`int groupSemester;`	Zmienna przechowująca semestr grupy.
`std::string groupSpecialization;`	Zmienna przechowująca kierunek grupy.

Wybrane funkcje klasy:

`Lesson(std::string name, std::string type, Time start, Time end, int classroom, std::string teacherName, std::string teacherSurname,int groupNumber, int groupSemester, std::string groupSpecialization);`	Konstruktor klasy. Jeśli godzina zakończenia jest mniejsza lub równa godzinie rozpoczęcia, rzucony zostaje wyjątek `WrongLessonException`.
`friend std::ostream & operator<<(std::ostream &s, const Lesson &l);`	Funkcja wypisująca do strumieia wszystkie informacje o lekcji. Używana przy zapisie do pliku obecnego stanu programu.
`void write(std::ostream &s,int schedule);`	Funkcja, która w zależności od podanego argumentu schedule wypisuje informacje o lekcji dla danego planu grupy (1), sali (2) lub nayczyciela (3).

Klasa Date

```
class Date
{
        int day;
        int month;
        int year;
        int yearDay;
```

```
        int weekDay;
        static const std::string daysOfWeek[7];
        static const int daysOfYear[12];
        void nextWeekDay();
        void previousWeekDay();
public:
        Date();
        Date(int day, int month, int year);
        ~Date();
        Date operator+(int number);
        Date operator++(int number);
        Date operator--(int number);
        bool operator==(Date date);
        bool isLeapYear(int _year);
        bool operator<=(Date date);
        int operator-(Date date);
        friend std::ostream &operator<<(std::ostream &s, const Date &d);
        friend std::istream &operator>>(std::istream &s, Date &d);
        std::string getWeekDayName();
        int getWeekDay();
};
```

Klasa przechowująca datę. Oprócz dnia, miesiąca i roku przechowuje również dzień tygodnia i dzień roku. Do obliczania dnia tygodnia i dnia roku wykorzystałam algorytm obliczania dnia tygodnia na podstawie opisu algorytmu na stronie http://www.algorytm.org/przetwarzanie-dat/wyznaczanie-dnia-tygodnia.html. Klasa umożliwia przeprowadzanie kilku różnych operacji na datach za pomocą różnych operatorów. Zawiera także statyczne tablice z nazwami dni tygodnia i ilością dni które już upłynęły dla poszczególnych miesięcy.

Zmienne klasy:

`int day;`	Zmienna przechowująca dzień.
`int month;`	Zmienna przechowująca miesiąc.
`int year;`	Zmienna przechowująca rok.
`int yearDay;`	Zmienna przechowująca dzień roku.
`int weekDay;`	Zmienna przechowująca dzień tygodnia.
`static const std::string daysOfWeek[7];`	Statyczna tablica z nazwami dni tygodnia.
`static const int daysOfYear[12];`	Statyczna tablica określająca ile dni już upłynęło w roku dla poszczególnych kolejnych miesięcy.

Wybrane funkcje klasy:

`Date(int day, int month, int year);`	Konstruktor klasy, sprawdza odpowiednio czy podany dzień, miesiąc oraz rok mieszczą się w odpowiednich zakresach. Jeśli chociaż jeden z nich nie mieści się w zakresie, rzucany jest wyjątek `WrongDateException`, zawierający informację o tym, które składniki zostały podane nieprawidłowo. Gdy składniki są prawidłowe, na podstawie algorytmu wyznaczony zostaje dzień roku oraz dzień tygodnia.
`Date operator+(int number);`	Funkcja dodająca do danej daty odpowiednią liczbę dni podaną jako argument. Dodawanie różni się w zależności od wielu czynników. Dla dodawania dużych liczb, przekraczających

	granice miesięcy funkcja wykonywana jest rekurencyjnie.
`Date operator++(int number);`	Funkcja jako operator postinkrementacji, zwiększa dzień o jeden, zwraca dzień przed zwiększeniem.
`Date operator--(int number);`	Funkcja będąca operatorem postdekrementacji, zmniejsza dzień o jeden, zwraca dzień przed dokonaniem zmiany.
`bool isLeapYear(int _year);`	Funkcja sprawdzająca na podstawie podanego jako argument roku czy jest on przestępny czy nie.
`int operator-(Date date);`	Funkcja zwracająca różnicę między dwoma datami jako liczbę dni.
`friend std::ostream &operator<<(std::ostream &s, const Date &d);`	Funkcja wypisująca do podanego strumienia datę w postaci dd.mm.yyyy.
`friend std::istream &operator>>(std::istream &s, Date &d);`	Funkcja pobierająca z podanego strumienia do zmiennej datę w formacie dd.mm.yyyy. Po wczytaniu dnia, miesiąca i roku tworzona jest nowa zmienna, by ustalony został dzień tygodnia i dzień roku. Funkcja używana do pobierania daty z pliku przechowującego stan programu, założenie, że dane w nim są poprawne.

Klasa Time

```
class Time
{
	int hour;
	int minutes;
public:
	Time();
	Time(int hour, int minutes);
	~Time();
	static Time differenceModule(Time t1, Time t2);
	bool operator>=(Time t);
	bool operator<=(Time t);
	bool operator==(Time t);
	friend std::ostream & operator<<(std::ostream &s, const Time &t);
	friend std::istream &operator>>(std::istream &s, Time &t);
};
```

Klasa przechowująca czas z dokładnością godzin i minut. Może porównywać godziny, zwracać moduł różnicy między dwoma czasami. Kontroluje także, czy podane godziny i minuty mieszczą się w odpowiednim zakresie.

Zmienne klasy:

`int hour;`	Zmienna przechowująca godzinę.
`int minutes;`	Zmienna przechowująca minuty.

Wybrane funkcje klasy:

`Time(int hour, int minutes);`	Funkcja sprawdza, czy podana liczba godzin i minut mieszczą się w standardowych zakresach. Jeśli chociaż jedna wartość jest zła, rzucany jest wyjątek WrongTimeException, informujący, która z wartości jest zła. Gdy wszystko jest w porządku, zmienna zostaje normalnie utworzona.
`static Time differenceModule(Time t1, Time t2);`	Funkcja zwracająca jako obiekt klasy Time moduł różnicy czasowej pomiędzy dwoma podanymi godzinami.
`friend std::ostream & operator<<(std::ostream &s, const Time &t);`	Funkcja wypisująca do podanego strumienia godzinę w formacie hh:mm
`friend std::istream &operator>>(std::istream &s, Time &t);`	Funkcja pobierająca z podanego strumienia godzinę w formacie hh:mm

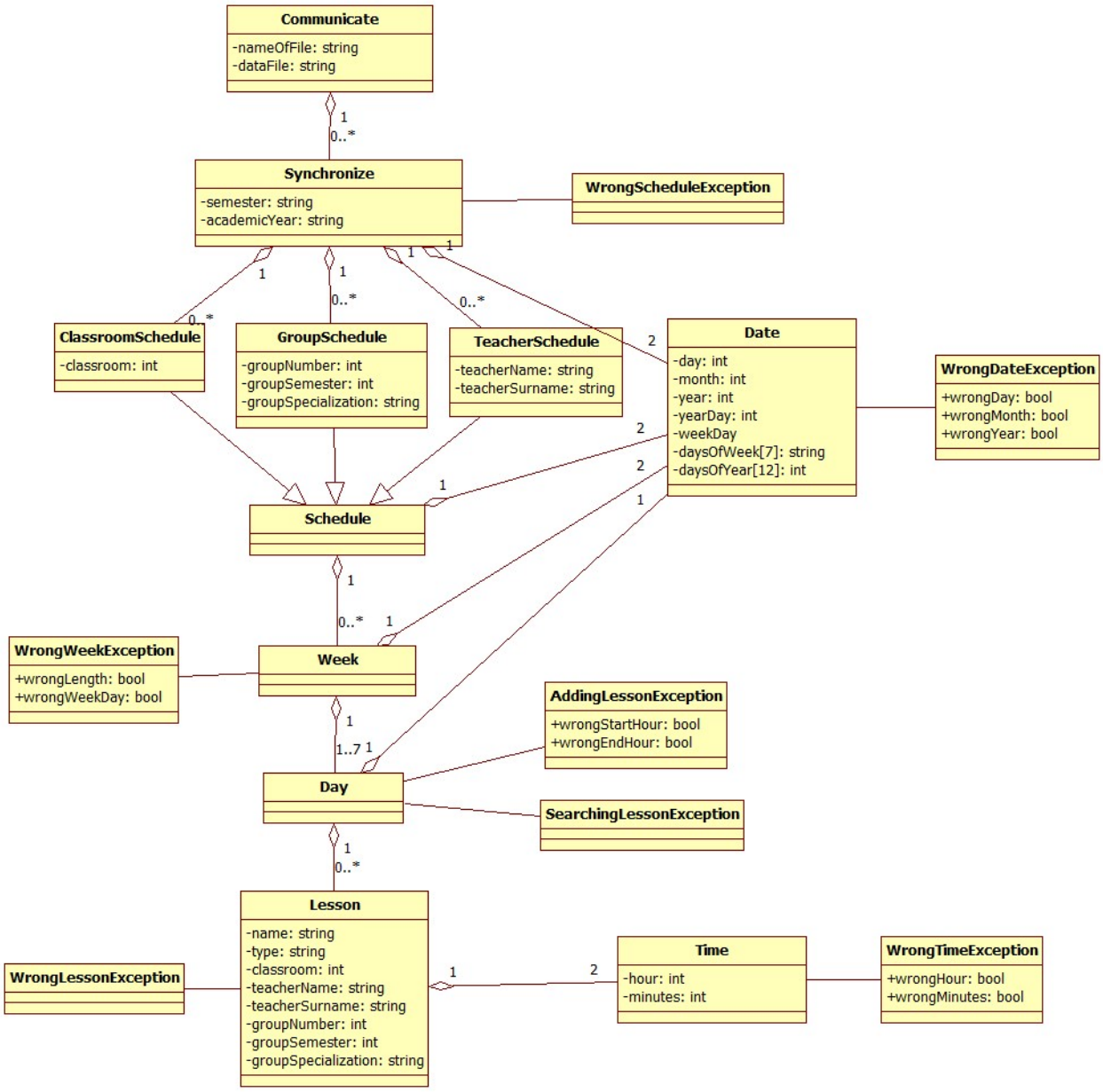
Klasy wyjątków:

W programie zostały zastosowane wyjątki. Poniżej przedstawiono wykorzystane klasy wyjątków:

`AddingLessonException`	Klasa zastosowana do rzucania wyjątków, gdy zajdzie niemożność dodania danej lekcji do dnia, występuje w klasie Day. Zawiera dwie zmienne typu bool – wrongStartHour i wrongEndHour. Wartość true tych zmiennych wskazuje, że odpowiednio czas rozpoczęcia lub/i czas zakończenia dodawanej lekcji nachodza na inne lekcje i nie jest możliwe jej dodanie.
`SearchingLessonException`	Klasa zastosowana do rzucania wyjątków, gdy nie znaleziono podanej lekcji w danym dniu, występuje w klasie Day.
`WrongDateException`	Klasa zastosowana do rzucania wyjątków, gdy użytkownik chce utworzyć dzień o złej wartości dnia/miesiąca/roku. Zawiera trzy zmienne typu bool – wrongYear, wrongMonth i wrongDay. Każda z tych zmiennych może przyjąć wartość true, gdy odpowiadający jej rok/miesiąc/dzień wychodzi poza dobry zakres. Występuje w klasie Date.
`WrongLessonException`	Klasa zastosowana do rzucania wyjątków, gdy czas zakończenia lekcji jest mniejszy lub równy czasowi rozpoczęcia lekcji, występuje w klasie Lesson.
`WrongScheduleException`	Klasa zastosowana do rzucania wyjątków, gdy data zakończenia semestru jest mniejsza od daty rozpoczęcia semestru. Występuje w klasie Synchronize.

`WrongTimeException`	Klasa zastosowana do rzucania wyjątków, gdy użytkownik poda złą godzinę lub liczbę minut. Zawiera dwie zmienne typu bool – wrongHour i wrongMinutes, które mogą przyjąć wartości true gdy godzina i/lub minuty nie mieszczą się w odpowiednim przedziale. Występuje w klasie Time.
`WrongWeekException`	Klasa zastosowana do rzucania wyjątków, gdy podano daty dla tygodnia mieszczące więcej niż 7 dni lub data końcowa jest mniejsza od początkowej – wtedy zmienna bool wrongLength przyjmuje wartość true, albo gdy chcemy uzyskać dostęp do dnia tygodnia który nie istnieje w danym tygodniu – wtedy zmienna bool wrongWeekDay przyjmuje wartość true. Występuje w klasie Week.

Poniżej przedstawiony został diagram klas wykonany za pomocą programu WhiteStarUML. Dla przejrzystości nie zostały w nim zawarte funkcje:



5.Testowanie i uruchamianie

Program został testowany wielokrotnie podczas tworzenia poszczególnych klas programu oraz po utworzeniu całego programu.

Poniżej przedstawiam kilka przykładowych zachowanych fragmentów kodu wykorzystywanych do testowania podczas tworzenia poszczególnych modułów.

Fragment testujący szerokość wypisywanych danych oraz pobieranie wpisywanych danych ignorując pojedynczy znak:

```
int day, month, year;
	std::cout.width(36);
	std::cout << left << "Podaj poczatkowa date (dd.mm.rrrr):";
	std::cin >> day;
	std::cin.ignore();
	std::cin >> month;
	std::cin.ignore();
	std::cin >> year;
	std::cout << std::endl;
	getchar();
```

Fragment testujący tworzenie zmiennych i działanie wyjątków przy dodawaniu lekcji nachodzącej na inne/usuwaniu nieistniejącej lekcji:

```
Day dzien(Date(3, 6, 2016));
	Lesson l1("MN", "wyklad", Time(9, 30), Time(10, 00)), l2("l2", "typ", Time(10, 15),
Time(12, 15)), l3("l3", "typ", Time(9, 50), Time(10, 20));
	dzien.addLesson(l2);
	dzien.addLesson(l1);
	cout << dzien;
	Week week(Date(6,6,2016),Date(12,6,2016));
try {
	dzien.deleteLesson(l3);
}
catch (SearchingLessonException) {
	cout << "Nie ma!" << endl;
}
try {
	dzien.addLesson(l3);
}
catch (AddingLessonException exc) {
	cout << "Nie udalo sie dodac!" << endl;
}
```

Fragment testujący operator inkrementacji oraz tworzenie nieprawidłowej daty:

```
Date dzisiaj(29, 2, 2016);
dzisiaj++;
Date jutro(28, 2, 2015);
jutro++;
try {
	Date dzika(34, 2, 2015);
}
catch (WrongDateException exception) {
	cout << "Zla data!!!" << endl;
}
if (dzisiaj == jutro) {
		cout << "Ops" << endl;
	}
```

Dla testowania całego programu wykorzystałam plik przechowujący następujące dane do wczytania:

```
semester 05.03.2016 20.06.2016 2015/2016 letni
```

```
day 13.06.2016
lesson 08:15 09:45 OR wyklad 1 Zbigniew Czechowicz 1 4 Informatyka
lesson 10:00 13:15 BD wyklad 1 Stanislaw Koziol 1 4 Informatyka
lesson 13:30 15:00 JA wyklad 1 Krzysztof Komin 1 4 Informatyka
lesson 15:45 17:15 SO wyklad 2 Bartosz Pach 1 4 Informatyka
endday
day 14.06.2016
lesson 8:30 10:00 JO cwiczenia 545 Katarzyna Tur 1 4 Informatyka
lesson 10:30 13:30 BD laboratorium 830 Mikolaj Kowalik 2 4 Informatyka
lesson 14:00 15:30 PK laboratorium 526 Maciej Budka 2 4 Informatyka
endday
day 15.06.2016
lesson 8:15 9:45 AiSD cwiczenia 545 Barbara Cieplik 1 4 Informatyka
lesson 10:00 11:30 JA laboratorium 526 Krzysztof Komin  2 4 Informatyka
lesson 11:00 14:00 BD laboratorium 830 Mikolaj Kowalik 1 4 Informatyka
endday
endsemester
```

Po wykonaniu programu i dokonaniu w trakcie jego działania różnych operacji, otrzymałam plik zawierający następujące dane:

```
semester 05.03.2016 20.06.2016 2015/2016 letni
day 13.06.2016
lesson 08:15 09:45 OR wyklad 1 Zbigniew Czechowicz 1 4 Informatyka
lesson 10:00 13:15 BD wyklad 1 Stanislaw Koziol 1 4 Informatyka
lesson 13:30 15:00 JA wyklad 1 Krzysztof Komin 1 4 Informatyka
endday
day 13.06.2016
lesson 15:45 17:15 SO wyklad 2 Bartosz Pach 1 4 Informatyka
endday
day 14.06.2016
lesson 14:00 15:30 PK laboratorium 526 Maciej Budka 2 4 Informatyka
endday
day 15.06.2016
lesson 10:00 11:30 JA laboratorium 526 Krzysztof Komin 2 4 Informatyka
endday
day 14.06.2016
lesson 08:30 10:00 JO cwiczenia 545 Katarzyna Tur 1 4 Informatyka
endday
day 15.06.2016
lesson 08:15 09:45 AiSD cwiczenia 545 Barbara Cieplik 1 4 Informatyka
endday
day 07.03.2016
lesson 08:30 11:30 BD laboratorium 830 Anna Kowalska 1 4 Informatyka
endday
day 21.03.2016
lesson 08:30 11:30 BD laboratorium 830 Anna Kowalska 1 4 Informatyka
endday
day 14.06.2016
lesson 10:30 13:30 BD laboratorium 830 Mikolaj Kowalik 2 4 Informatyka
endday
day 15.06.2016
lesson 11:00 14:00 BD laboratorium 830 Mikolaj Kowalik 1 4 Informatyka
endday
endsemester
semester 01.10.2016 30.03.2017 2016/2017 zimowy
endsemester
```

Otrzymany plik zawiera wszystkie dane które powinien, chociaż dane zapisane są w innej kolejności – zostały zapisane przeglądając plany sal. Sprawdziłam również ponowne wczytywanie otrzymanych danych – wczytują się one i zapisują poprawnie.

Przykładowe uzyskane plan lekcji dla opcji zapisywania planu do pliku:

```
Informatyka, semestr 4, grupa 2
--------------------
Wtorek
--------------------
BD
laboratorium
10:30 - 13:30
830
Mikolaj Kowalik

PK
laboratorium
14:00 - 15:30
526
Maciej Budka

--------------------
Sroda
--------------------
JA
laboratorium
10:00 - 11:30
526
Krzysztof Komin

--------------------
```

Drugi przykładowy plan:

```
Informatyka, semestr 4, grupa 1
--------------------
Poniedzialek
--------------------
OR
wyklad
08:15 - 09:45
1
Zbigniew Czechowicz

BD
wyklad
10:00 - 13:15
1
Stanislaw Koziol

JA
wyklad
13:30 - 15:00
1
Krzysztof Komin

SO
wyklad
15:45 - 17:15
2
Bartosz Pach

--------------------
Wtorek
--------------------
JO
cwiczenia
08:30 - 10:00
545
Katarzyna Tur
```

```
--------------------
Sroda
--------------------
AiSD
cwiczenia
08:15 - 09:45
545
Barbara Cieplik

BD
laboratorium
11:00 - 14:00
830
Mikolaj Kowalik

--------------------
```

Podczas testowania napotkałam na różne błędy. Wszystkie znalezione poprawiłam. Oto niektóre błędy na które napotkałam:
- przy testowaniu wprowadzania lekcji nachodzącej na inne lekcje wyświetlany zostawał jedynie komunikat o braku możliwości dodania, bez podania przyczyny – sprawdzana była zła zmienna
- operator inkrementacji dni nie uwzględniał jednego dnia więcej w zmiennej przechowującej dzień roku dla roku przestępnego – podczas wypełniania tygodni dniami ostatnie dnie miesięcy po lutym zostawały nieuwzględnione i wektor z dniami zawierał jedynie 6, a nie 7 dni.
- podczas wyszukiwania tygodnia do którego dodać dzień na podstawie podanej daty po ostatnim przejściu pętli iterator był dodatkowo inkrementowany, mimo że znaleziono odpowiedni tydzień – dzień zostawał dodawany do złego tygodnia
- operator porównania dla klasy GroupSchedule porównywał złe zmienne, co skutkowało nieznalezieniem odpowiedniego planu
- przy wczytywaniu podanego przez użytkownika zajęcia nazwisko nauczyciela zapamiętywane było w złej zmiennej