03/01/2021

TOURE HOSSEINE HABRAHAM YOUSSEF

VIVIEN RAULIN

Rapport du projet C++

MASTER 1

# PRESENATION DU PROJET

Parmi les différents projets proposés, nous avons choisi de travailler sur la gestion d’un emploi du temps.

Selon notre découpage, Une journée est composée de 4 créneaux fixes : 8h-10h, 10h-12h, 14h-16h et 16h-18h. Le but est de placer des cours dans ces créneaux tout en tenant compte des ressources nécessaires pour réaliser ces cours (enseignants, salle, professeur…). On doit pouvoir ajouter, déplacer ou supprimer des cours et des ressources.

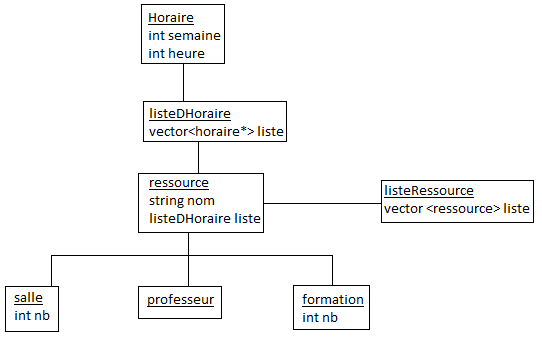
On doit aussi pouvoir visualiser l’emploi du temps sur une semaine correspondant à une ou plusieurs ressources avec les détails du cours placé à un moment donné.

Une année est constituée de 52 semaines numérotées de 1 à 52 et chaque semaine aura 7 jours du lundi au dimanche (on ne tient pas compte des mois et des numéros de jours dans le mois). L’affichage est exportable dans plusieurs formats (txt, CSV…).

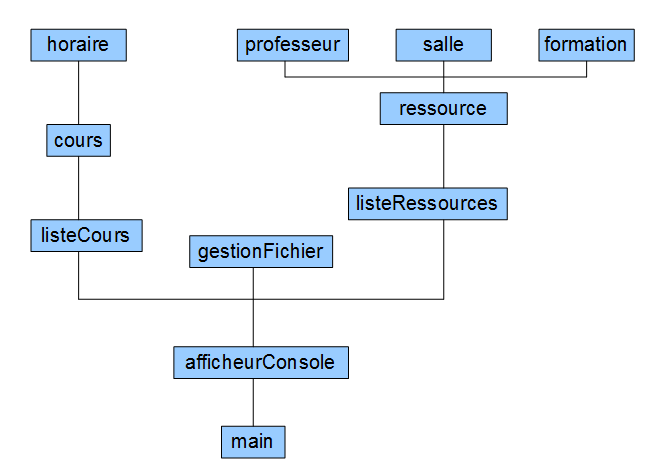
# Problèmes rencontrés et solutions apportées

## Organisation des classes

Notre première organisation présentait des problèmes d’interdépendance :



Nous avons résolu ce problème en mettant au point une nouvelle organisation des classes :

**

# Classes du projet

* AfficheurConsole.cpp

Elle permet de gérer l’affichage des données à l’écran et réalisant les liens existants entre les différentes classes.

* Cours.cpp

Elle permet de définir un cours avec les attributs

#include "../lib/cours.h"

/\*\*

@brief Constructeur de l'objet cours

@param[in] h - un objet de type horaire

@param[in] prof - un objet de type professeur

\*/

cours::cours(const horaire & h, professeur \* p, salle \* s, formation \* f):d\_horaire{h},d\_professeur{p},d\_salle{s},d\_formation{f} {}

//GET

/\*\*

@brief Renvoie l'horaire du cours

@return d\_horaire - un objet de type horaire

\*/

horaire cours::horaireDuCours() const {return d\_horaire;}

/\*\*

@brief Renvoie le professeur lie a un cours

@return d\_professeur - un objet de type horaire

\*/

professeur cours::professeurDuCours() const {return \*d\_professeur;}

/\*\*

@brief Renvoie la salle lie a un cours

@return d\_salle - un objet de type horaire

\*/

salle cours::salleDuCours() const {return \*d\_salle;}

/\*\*

@brief Renvoie la formation lie a un cours

@return d\_formation - un objet de type horaire

\*/

formation cours::formationDuCours() const {return \*d\_formation;}

* Formation.cpp

Cette classe est utilisée pour définir une formation (formation = groupe)

#include "../lib/formation.h"

#include "../lib/afficheurConsole.h"

/\*\*

@brief Constructeur de l'objet formation

@param[in] nomFormation - nom de la formation

@param[in] nombreEtudiant - Nombre d'etudiant dans la formation

\*/

formation::formation(const string & nomFormation, int nombreEtudiant, int nombreHeureMax) : ressource{nomFormation}, d\_nombreEtudiant{nombreEtudiant},d\_nombreHeureMax{nombreHeureMax} {}

/\*\*

@brief Modifie le nombre d'etudiant dans la formation

@param[in] nombreEtudiant - le nombre d'etudiant

\*/

void formation::changerNombreEtudiant(int nombreEtudiant) {d\_nombreEtudiant = nombreEtudiant;}

/\*\*

@brief Renvoie le nombre d'etudiant dans la formation

@return d\_nombreEtudiant - le nombre d'etudiant

\*/

int formation::nombreEtudiant() const { return d\_nombreEtudiant;}

int formation::nombreHeureMax() const { return d\_nombreHeureMax; }

* gestionFichier.cpp

cette classe permet de gérer notre fichier dans lequel sera stocker l’ensemble de nos données. Grâce à elle, il est possible d’écrire et de lire un fichier

#include "../lib/formation.h"

#include "../lib/afficheurConsole.h"

/\*\*

@brief Constructeur de l'objet formation

@param[in] nomFormation - nom de la formation

@param[in] nombreEtudiant - Nombre d'etudiant dans la formation

\*/

formation::formation(const string & nomFormation, int nombreEtudiant, int nombreHeureMax) : ressource{nomFormation}, d\_nombreEtudiant{nombreEtudiant},d\_nombreHeureMax{nombreHeureMax} {}

/\*\*

@brief Modifie le nombre d'etudiant dans la formation

@param[in] nombreEtudiant - le nombre d'etudiant

\*/

void formation::changerNombreEtudiant(int nombreEtudiant) {d\_nombreEtudiant = nombreEtudiant;}

/\*\*

@brief Renvoie le nombre d'etudiant dans la formation

@return d\_nombreEtudiant - le nombre d'etudiant

\*/

int formation::nombreEtudiant() const { return d\_nombreEtudiant;}

int formation::nombreHeureMax() const { return d\_nombreHeureMax; }

* Horaire.cpp

Elle nous permet de gérer nos créneaux en vérifiant les contraintes existantes.

* Salle.cpp

Elle gère nos salles tout en vérifiant que le nombre d’étudiant ne soit pas supérieur à la capacité d’une salle.

#include "../lib/salle.h"

#include "../lib/afficheurConsole.h"

/\*\*

@brief Constructeur de l'objet salle

@param[in] nomSalle - une chaine de caracteres representant le nom de la salle

@param[in] taille - capacite de la salle, nombre detudiants quelle peut acceuillir

\*/

salle::salle (const string & nomSalle, int taille): ressource{nomSalle}, d\_nombrePlaces{taille} {}

//SET

/\*\*

@brief Change le nombre de places

@param[in] val - entier représentant le nombre de places

\*/

void salle::changerNombrePlaces(int val) {d\_nombrePlaces = val;}

//GET

/\*\*

@brief Renvoie le nombre de places

@return d\_nombrePlaces - un entier représentant le nombre de places

\*/

int salle::nombrePlaces() const {return d\_nombrePlaces;}

* Prefosseur.cpp

Elle définit un professeur avec respect des contraintes .

#include "../lib/professeur.h"

#include "../lib/afficheurConsole.h"

/\*\*

@brief Constructeur de l'objet professeur

@param[in] nom - une chaine de caractere representant le nom du professeur

\*/

professeur::professeur(const string& nom,const int& nbH) :ressource{ nom} {}

* Main.cpp

Il s’agit du programme principal ; elle met en relation l’ensemble des classes et produit notre solution

#include <iostream>

#include <cstdlib>

#include "../lib/afficheurConsole.h"

int main()

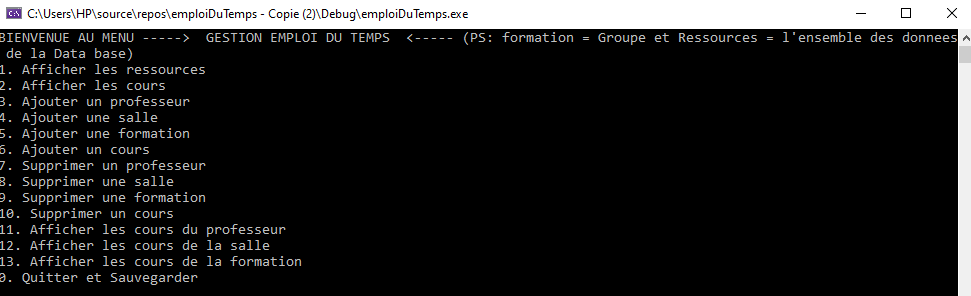
{

afficheurConsole aC{cout};

aC.menu();

}

# utilisation du programme

Ouvrir le fichier **emploiDuTemps** dans le dossier du projet avec Visual studio et ensuite cliquer sur **Débogueur Windows local**

**AVANT DE COMPILER LE PROJET, IL FAUT OBLIGATOIREMENT ADAPTER LE CHEMIN D’ACCES AU FICHIER DATA**

**POUR SE FAIRE, ALLER DANS LA CLASSE afficheurConsole.cpp et modifier le chemin dans la fonction** void afficheurConsole::menu() **en se plaçant sur la ligne 67**

Dans notre cas, le chemin est C:/Users/HP/source/repos/emploiDuTemps - Copie (2)/data.txt