Université Paris Diderot **Langages à objets avancé**
M1 **Année 2019–2020**

Classes
(et classes d'énumération)

1 Horloge et Calendrier

1.1 Horloge

Écrivez une classe `Horloge` avec des heures, des minutes et des secondes. Vous devrez pouvoir afficher l'heure courante de manière lisible pour un être humain et incrémenter d'une seconde.

Attention après l'incrémentation de 23H 59M 59S, on passe à 0H 0M 0S (changement de jour) ; dans ce cas-là, la méthode d'incrémentation devra renvoyer « true » (et « false » sinon).

Testez cette classe en écrivant un programme qui lit les heures, les minutes et les secondes depuis l'entrée standard (utiliser `cin`), l'affiche, l'incrémente et affiche le résultat en affichant si un changement de jour a eu lieu.

1.2 Date

Par souci de simplification, nous supposerons que les mois de l'année ont tous 30 jours. Une année aura donc 360 jours.

Écrivez une classe `Date` avec un constructeur, une méthode d'affichage et une méthode pour passer au jour suivant.

Testez cette classe le la même façon que la précédente.

1.3 Calendrier

En utilisant les deux premières classes, écrivez une classe `Calendrier` qui représente la date et l'heure et qui a une méthode qui permet d'avancer d'une seconde. Pour tester de manière plus aisée, on ajoutera une méthode qui avance d'une heure.

2 Modélisation : Un peu de musique

2.1 Présentation

Cette partie a pour but de modéliser de manière très simplifiée des informations sur des morceaux de musique classique.

Nous considérerons qu'un morceau de musique est fait pour un instrument principal : par exemple, si on parle d'une sonate pour violon, on considèrera que l'instrument principal est le violon.

De même, un morceau a une tonalité principale, par exemple « Mi mineur » pour une « fugue en Mi mineur ». (cf. plus bas)

Bien sûr, un morceau a un nom et un compositeur, un seul pour simplifier.

2.2 Explication des classes d'énumération

Les classes d'énumérations, sont des classes dont les objets ne peuvent avoir que les valeurs indiquées lors de la déclaration de la classe.

Exemple d'utilisation :

```
// Déclaration de la classe
enum class Dir {est, ouest, nord, sud};

// Surcharge de l'opérateur <<
inline ostream& operator<<(ostream& out, Dir d) {
  switch (d) {
    case Dir::est: out << "est";break;
    case Dir::ouest: out << "ouest"; break;
    case Dir::nord: out << "nord";break;
    case Dir::sud: out << "sud";break;
  }
  return out;
}

// Déclaration et initialisation d'un objet de type Dir
Dir d {Dir::ouest};

// Exemple d'utilisation avec la surcharge
cout << d << endl; // Affiche "ouest"
```

Vous utiliserez ce genre de classe partout où cela a un sens.

2.3 Tonalité

Une tonalité est la donnée d'un nom de note (Do, Ré, Mi, Fa, Sol, La ou Si), d'une altération éventuelle (dièse ou bémol) et d'un qualificatif (mineur ou Majeur).

2.4 Classes à implémenter

Après avoir dessiné leur diagramme UML, programmez les classes suivantes en testant régulièrement (il est possible d'ajouter des classes supplémentaires si nécessaire) :

Note : qui représente un nom de note (Do, Ré, Mi, Fa, Sol, La ou Si).

Altération : qui représente une altération ou une absence d'altération.

Qualificatif : mineur ou Majeur.

Tonalité : cf. plus haut.

Instrument : un instrument a un nom et une famille. Le violon fait partie des cordes, la trompette des vents, les cymbales des percussions...

Morceau : cf. plus haut.

Écrivez les méthodes d'affichage nécessaires.

A Quelques précisions sur le makefile

On remarquera que :

1. Les dépendances pour générer l'exécutable final ne contiennent que les .o (tous ceux du programme)

2. Les dépendances pour générer un fichier .o ne contiennent que le fichier .cpp correspondant et des fichiers .hpp.
3. Pour les dépendances pour générer un fichier .o, Il faut mettre tous les fichier .hpp dont on utilise une déclaration, même si, par le jeu des indirections, il n'est pas nécessaire de faire les `#include` correspondant : Par exemple : si on écrit `Note::Fa` dans le fichier testMorceaux.cpp, -dans les dépendances il faudra indiquer Musique.hpp. -dans le fichier testMorceaux.cpp par contre, si on a déjà `#include "Morceaux.hpp"`, comme Morceaux.hpp aura au moins un `#include "Tonalite.hpp"` qui lui-même aura un `#include "Musique.hpp"`, ce n'est pas nécessaire, même si on peut le mettre sans que ça ne cause de problème.