Module R3.04
Examen Machine du Vendredi 27 Octobre 2023

Consignes – A Lire attentivement (comme tout le sujet !)

- Durée : **1h30** / Tous documents autorisés.
- En début d'examen, tapez dans un shell : **`debut-examen-r304`**
- Vous serez placé·e dans votre répertoire d'examen et **`Clion`** sera automatiquement lancé : **`/users/info/pub/2a/R304/examen/reponses/votre_login`**
- Depuis **`CLion`**, ouvrez le projet contenu dans répertoire d'examen : **`/users/info/pub/2a/R304/examen/reponses/votre_login`**
- Pensez à sauvegarder régulièrement vos fichiers.
- A la fin, il n'y a rien à faire si ce n'est vous déconnecter.
- **Vous ne devez créer aucun fichier et vous ne devez modifier que le fichiers demandés (ceux avec le suffixe `_ACOMPLETER`).**

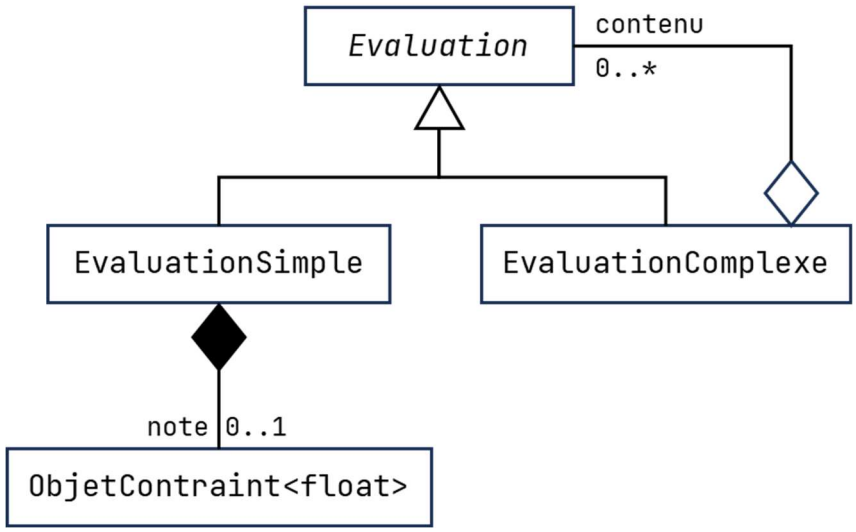
Exercice 1. Pattern Composite & Exceptions

1.1. Spécifications

Pour mesurer les résultats d'un étudiant, on souhaite disposer de 2 types d'**évaluations** :

1. **Evaluation simple**, qui correspond à une épreuve, représentée par 3 informations :
   - **Un intitulé :** une chaîne de caractères (par ex. « Contrôle machine »).
   - **Un coefficient** : entier non signé (par ex. 2).
   - **Une note** : un réel contraint compris entre 0.0 et 20.0 (par ex. 14.5).
   - **Attention** : une évaluation simple peut ne pas avoir de **note** (si l'évaluation simple n'a pas encore été corrigée).
2. **Evaluation complexe** qui est une **agrégation** d'évaluations simples et/ou complexes. Une évaluation complexe est caractérisée par 3 informations :
   - **Un intitulé** : une chaîne de caractères (par ex. « Module R3.04 »).
   - **Un coefficient** qui est égal à la somme des coefficients des évaluations que contient l'évaluation complexe, si elle contient des évaluations.
   - **Une note** qui est déterminée de la façon suivante :
     - Si une évaluation complexe contient au moins une évaluation qui n'a pas de note, on considère alors que l'évaluation complexe n'a pas de note non plus.
     - Si une évaluation complexe contient des évaluations qui possèdent toutes une note, alors la note de l'évaluation complexe est calculée comme étant la moyenne pondérée, par le coefficient de chaque évaluation, des notes de ses évaluations.

Pour modéliser les évaluations, on vous demande de mettre en œuvre le ***pattern composite*** selon le diagramme de classe suivant :



Pour chaque évaluation, simple ou complexe, on veut disposer des **opérations** (au sens du *pattern composite)* suivantes :

- **`const std::string & getIntitule()`** :
  retourne l'intitulé de l'évaluation.
- **`void setIntitule(const std::string & intitule)`** :
  modifie l'intitulé de l'évaluation.
- **`unsigned int getCoefficient()`** :
  retourne le coefficient de l'évaluation,
  *ou lève une exception **`range_error`** dans le cas d'une évaluation complexe vide.*
- **`void setCoefficient(unsigned int coefficient)`** :
  modifie le coefficient d'une évaluation simple,
  *ou lève une exception de type **`logic_error`** dans le cas d'une évaluation complexe.*
- **`float getNote()`** :
  retourne la note de l'évaluation,
  *ou lève une exception **`range_error`** dans le case d'une évaluation simple sans note*
  *ou lève une exception **`range_error`** dans le cas d'une évaluation complexe vide ou qui contient au moins une évaluation sans note.*
- **`void setNote(float note)`** :
  définit (ou modifie si elle existe déjà) la note d'une évaluation simple,
  *ou lève une exception de type **`range_error`** si la note n'est pas comprise entre 0 et 20,*
  *ou lève une exception de type **`logic_error`** dans le cas d'une évaluation complexe.*
- **`void affiche()`** :
  *affiche sur **`std::cout`** l'intitulé, le coefficient et la note d'une évaluation (et rien de plus);*
  si l'évaluation n'a pas de coefficient, ou pas de note, on affichera alors « ND » (Non Dispo).
- **`void addEvaluation(const Evaluation & evaluation)`** :
  ajoute une évaluation à une évaluation complexe,
  ou lève une exception de type ***`logic_error`*** *dans le cas d'une évaluation simple*

Les constructeurs suivants doivent être disponibles :

- La classe **`EvaluationSimple`** doit offrir un constructeur permettant d'initialiser son intitulé et son coefficient :
  `EvaluationSimple(const std::string & intitule, unsigned int coefficient)`
- La classe **`EvaluationComplexe`** doit offrir un constructeur pour initialiser son intitulé :
  `EvaluationComplexe(const std::string & intitule)`

1.2. Travail à Réaliser

Vous devez spécifier et implémenter les 3 classes du composite : **`Evaluation`**, **`EvaluationSimple`** et **`EvaluationComplexe`**. Pour cela, vous compléterez les fichiers suivants qui sont fournis :

- `Evaluation_ACOMPLETER.h` et `Evaluation_ACOMPLETER.cpp`
- `EvaluationSimple_ACOMPLETER.h` et `EvaluationSimple_ACOMPLETER.cpp`
- `EvaluationComplexe_ACOMPLETER.h` et `EvaluationComplexe_ACOMPLETER.cpp`

1.2.1. Indications :

- Le patron de classe **`ObjetContraint<T>`** est fourni. Il lève une exception de type ***`range_error`*** quand on essaye de définir une **`valeur`** non comprise entre **`min`** et **`max`**.
- Vous devez factoriser le plus de choses possibles dans la classe **`Evaluation.`**
- Vous devrez également avoir de bonnes pratiques (**`const`**, déclarations de constantes, forme canonique de Coplien, etc… quand c'est utile/nécessaire).
- ***Attention*** *: les prototypes des méthodes fournis ci-dessus (et disponibles dans le fichier `operations.txt`) sont donnés à titre d'indication… C'est à vous de les coder parfaitement !*

1.2.2. Pour tester votre code :

- Utilisez la configuration **exo1** pour compiler
- Vous pourrez tester vos classes avec le programme principal fourni **exo1Main.cpp**, que vous ne devez pas modifier (*de toute façon son contenu original sera rétabli en fin d'examen*).
- Lisez attentivement le code du fichier **exo1Main.cpp** : il vous montre un exemple d'utilisation du composite et vous aide donc à comprendre comment vous devez l'implémenter
- En commentaires, à la fin du fichier **exo1Main.cpp**, vous trouverez la trace que devrait produire votre programme si vous avez bien codé le composite.
- Dans la mesure du possible, essayez de rendre un projet qui compile (commentez ce qui ne compile pas).

Exercice 2. Tests Unitaires

On vous demande d'écrire, à l'aide du *framework Google Test,* des tests unitaires, aussi complets que possible, pour valider la classe **EvaluationComplexe** spécifiée à l'exercice 1.

Attention *: vous pouvez parfaitement écrire ces tests même si vous n'avez pas (ou pas encore) implémenté la classe* ***EvaluationComplexe****. L'énoncé de l'exercice 1 vous fournit une spécification précise de cette classe. Vos tests unitaires doivent être écrits de manière à vérifier que ces spécifications sont (ou seront respectées). Evidemment, vous ne pourrez lancer vos tests que si la classe* ***EvaluationComplexe*** *a été implémentée.*

2.1. Travail à Réaliser

- Ecrivez vos tests en complétant le fichier **EvaluationComplexeTest_ACOMPLETER.cpp**
- Pour lancer vos tests, utilisez la configuration **exercice2**
- Ecrivez un test pour chacune des méthodes suivantes :
    - **EvaluationComplexe** (le constructeur)
    - **getCoefficient**
    - **getNote**
    - **addEvaluation**
- *La mise en place de* ***fixtures*** *n'est pas indispensable mais elle augmentera votre note si elle est faite !*