POO & C++ : TP 3

EISE4 2014—2015

Pablo Rauzy

rauzy@enst·fr

pablo.rauzy.name/teaching.html#epu-cpp

13 octobre 2014

N'oubliez pas :

- Les TPs doivent être rendus par courriel à `rauzy@enst·fr` au plus tard le lendemain du jour où ils ont lieu avec `[EISE4]` suivi de vos noms dans le sujet du mail.
- Le code rendu doit être propre !
- Le TP doit être rendu dans une archive tar gzippée : `tar czvf NOMS.tgz NOMS` où `NOMS` est le nom du répertoire dans lequel il y a votre code (idéalement, les noms des gens du groupe) ; attention à ne pas rendre un répertoire avec des fichiers temporaires dus à la compilation (pensez à faire un `make clean` avant `tar`).
- Dans le répertoire avec vos noms, chaque exercice doit être dans son propre répertoire.
- Quand un exercice demande des réponses qui ne sont pas du code, vous les mettrez dans un fichier texte `reponses.txt`.
- Le code est de préférence en anglais, avec des commentaires en français ou anglais, en restant cohérent.
- Le code doit être proprement indenté.
- Respectez les conventions de nommage :
  - variables : `nom_explicite` (*e.g.*, `game_type`),
  - fonctions : `verbeAction` (*e.g.*, `launchNewGame`),
  - classes : `NomExplicite` (*e.g.*, `BoardGame`),
  - membres privés ou protégés : comme variable ou fonction + `_` (*e.g.* `max_number_of_player_`, ou `updateState_`),
  - accesseurs d'affectation : `set_` + nom de l'attribut sans le `_` final (*e.g.*, `set_max_number_of_player`),
  - accesseurs de consultation : nom de l'attribut sans le `_` final (*e.g.*, `max_number_of_player`),
  - constantes : `NOM_EXPLICITE` (*e.g.*, `VERSION_NUMBER`).
- Pensez à utiliser les outils de développement comme le debugger `gdb` ou le profiler `valgrind`.
- N'hésitez jamais à chercher de la documentation par vous-mêmes sur le net !

Dans ce TP :

- Templates
- Exceptions.

Exercice 0.

Récupération des fichiers nécessaires.

1. Pensez à organiser correctement votre espace de travail, par exemple tout ce qui se passe dans ce TP devrait être dans `~/cpp/tp3/`.

2. Récupérez les fichiers nécessaires au TP sur la page du cours.

3. Une fois que vous l'avez extrait de l'archive (`tar xzf tp3.tgz`), renommez le répertoire `tp3` en les noms de votre groupe (par exemple en `Dupont-Dupond`). Si vous ne le faites pas tout de suite, pensez à le faire avant de rendre votre TP.

Exercice 1.

Le but de cet exercice va être de rendre nos conteneurs génériques en plus d'être polymorphiques. Pour cela on va utiliser des templates.

1. Finir rapidement l'exercice 2 du TP 2 (celui qu'on a fait ensemble en classe).

2. Afin de conserver les réponses de l'exercice précédent, copiez les fichiers finaux de l'exercice 2 du TP précédent dans un nouveau répertoire, et travaillez à présent sur ces copies.

→ Transformez la classe abstraite `Iterable` en modèles de classe pour que l'utilisateur puisse choisir un autre type de donnée que des `int`.

3. → Adaptez le conteneur `Vector` pour le rendre générique.

4. Pour le conteneur `List`, il faut aussi penser à mettre à jour la classe `Node` pour en faire un modèle aussi.

→ Adaptez le conteneur `List` pour le rendre générique.

5. Pour rappel, la dernière question du TP 2 était : "*Écrivez vous-même une nouvelle fonction dans le fichier `main.cpp` qui utilise l'interface `Iterable` pour itérer sur les éléments en ordre inverse et n'afficher que les multiples d'un nombre qu'on lui a passé en argument, puis testez-la pour les `List` et pour les `Vector`.*".

On suppose qu'on a rendu cette fonction générique aussi en en faisant un modèle de fonction qui prend non plus un `Iterable` mais un `Iterable<T>` en argument.

→ Sans nécessairement coder, quels problèmes pensez-vous rencontrer en essayant par exemple de déclarer un `List<Vector<string> >` et d'utiliser la fonction de la dernière question du TP 2 dessus ?

Sans nécessairement le faire non plus, dites comment vous régleriez ces problèmes.

Exercice 2.

Dans cet exercice le code d'une calculatrice en notation polonaise inversée vous est fourni, mais il ne gère pas les erreurs. Le but de l'exercice va être de rajouter la gestions des erreurs en utilisant des exceptions.

Le principe de la notation polonaise inversée est de fonctionner avec une pile de valeurs et de faire les opérations en dépilant deux valeurs et en rempliant le résultat.

Par exemple, le calcul `1 + ((2 + 3) * 4 - 5)` s'écrit `1 2 3 + 4 * 5 - +`. Cela se lit comme :

- empile `1` ; `[ 1 ]`
- empile `2` ; `[ 2 1 ]`
- empile `3` ; `[ 3 2 1 ]`
- dépile `3`, dépile `2`, empile `2 + 3 = 5` ; `[ 5 1 ]`
- empile `4` ; `[ 4 5 1 ]`
- dépile `4`, dépile `5`, empile `5 * 4 = 20` ; `[ 20 1 ]`
- empile `5` ; `[ 5 20 1 ]`
- dépile `5`, dépile `20`, empile `20 - 5 = 15` ; `[ 15 1 ]`
- dépile `15`, dépile `1`, empile `1 + 15 = 16`. `[ 16 ]`

1. Explorez le code dans le but de comprendre comment il est organisé et comment il fonctionne.

→ Décrivez brièvement ce que vous en avez compris : quel est le rôle les différents modules, quelles exceptions existent déjà et à quoi servent-elles. Donnez au passage la liste des opérations qui sont possibles.

2. Compilez le programme en utilisant le `Makefile` fourni, puis jouez un peu avec la calculatrice pour comprendre comment elle fonctionne.

→ Normalement, un avertissement concernant un `switch` est affiché par le compilateur lors de la compilation, pouvez-vous l'expliquer (et faites le) ? Le corriger (comment ?) ?

3. Il y a un autre `switch` pour lequel le compilateur ne peut pas afficher le même genre d'avertissement et où il serait pourtant bien utile.

→ Pouvez-vous dire de quel `switch` il s'agit ? Décrivez comment déclencher une erreur lors de l'exécution du programme en utilisant le code problématique.

4. → Ajoutez une exception pour gérer ce problème et utilisez là faire un retour à l'utilisateur.

5. Il reste encore d'autres soucis. Par exemple, regardez ce qu'il se passe quand on entre le calcul `1 2 3 + - *`. Ce n'est pas satisfaisant : la pile n'était pas vraiment vide, elle ne contenait qu'un seul élément, que l'on perd complètement du coup.

→ Corrigez ce problème à l'aide d'une nouvelle exception `StackTooSmall`.

6. Il reste une erreur possible, qui est très classique, avec la division.

→ Corrigez la avec une nouvelle exception, qui cette fois fera quitter le programme après avoir afficher l'erreur à l'utilisateur.