



Maîtriser les patterns : Design et Template en C++

Runtrack C++ : Jour 4

Job 01 – Max

Créer un template de fonction “**max**” qui prend **deux paramètres** de n’importe quel type comparable, et retourne le plus grand des deux.

Job 02 – Box

Créer une classe template “**Box**”, qui pourra contenir un élément de n’importe quel type, et permettre d’y accéder.

Job 03 – Box*

Créer une spécialisation de “**Box**” qui pourra contenir un pointeur sur un élément de n’importe quel type, et permettra d’accéder à son contenu déréférencé.



Job 04 – Shape

Créer une classe **abstraite** “**Shape**”, ainsi que des classes dérivées “**Circle**” et “**Rectangle**” qui implémentent la méthode “**draw**”.

Créer ensuite un template de fonction “**drawShape**”, qui pourra utiliser la fonction “**draw**” de n’importe quelle classe dérivée de “**Shape**”.

Job 05 – ShapeFactory

Créer une classe “**ShapeFactory**”, qui possède une méthode **statique** “**createShape**” qui prend en paramètre **un string**, représentant un type de “**Shape**”, et qui renvoie une instance adéquate, si le type fourni est correct.

Job 06 – ShapeFactory<>

Modifier la classe “**ShapeFactory**”, pour qu’elle soit capable de créer n’importe quelle classe qui dérive de “**Shape**”, grâce aux **templates**.

Job 07 – print

Créer une fonction template “**print**” qui peut recevoir n’importe quel nombre d’arguments, de n’importe quel type, les afficher dans la sortie standard, séparés par une virgule et un espace, et se terminant par un retour à la ligne.



Job 08 – Factory+

Créer une classe de base “**Enemy**”, ainsi que des classes “**Goblin**”, “**Ogre**”, “**Dragon**” qui implémentent la méthode “**attack**”.

Créer ensuite une classe “**EnemyFactory**”, qui contiendra :

- Une **map** (composée d’un string(cle) et de fonction(valeur)).
- Un **constructeur** qui initialise la map avec des clés correspondantes aux types d’ennemis disponibles, et des fonctions pour instancier les classes adéquates..
- Une méthode “**createEnemy**” reçoit un string correspondant à un type d’ennemi, utilise la map pour créer l’objet voulu (si le type fourni est valide).

Rendu

La runtrack doit être disponible sur votre github, au nom “runtrack_cpp”.

Les fichiers doivent être organisés précisément, comme indiqué dans les énoncés, dans un dossier correspondant à leur jour respectif.

Exemple : **jour01/job01/hello_world.cpp**

Le format de sortie doit donc être respecté.



Base de connaissances

- [Créer des templates](#)
- [templates](#)
- [les design patterns](#)
- [La référence des design patterns](#)