**Quelques éléments à vérifier dans votre code**

|  |  |
| --- | --- |
| Les paramètres des méthodes qui ne sont pas modifiés sont déclarés ***const***. |  |
| Les méthodes qui ne modifient pas l’état de l’objet sont déclarées ***const****.* |  |
| Le passage de paramètres par valeur est utilisé pour les types de base seulement. Dans tous les autres cas, le passage par référence ou référence constante est la norme. |  |
| Une gestion des erreurs est faite avec des exceptions. |  |
| Aucun objet n’est copié inutilement (un objet passé par valeur, l’ajout d’un objet dans un conteneur, une méthode qui retourne un objet par valeur, etc.). |  |
| Le constructeur ne fait qu’initialiser les attributs de la classe. Si possible, l’initialisation des attributs se fait avant les { } avec le : |  |
| Si nécessaire, les copies profondes sont gérées. Si les copies ne sont pas autorisées le constructeur de copie et l’opérateur = sont déclarés privés. |  |
| En général, la classe responsable d’une allocation de mémoire avec **New** est aussi responsable de la libération de la mémoire avec **Delete**. |  |
| Il y a une constance (homogénéité) dans l’écriture du code. Exemples :   * L’ordre des entêtes est le même (exemple : de la plus spécifique à la plus générale) * L’ordre des éléments dans les classes est le même (exemple : *public* avant *private*, les constructeurs avant les méthodes, le destructeur après le constructeur, etc.) * L’ensemble des classe respects les mêmes normes et standards d’écriture. |  |
| Le code de chacune des méthodes est « léger ». Un seul but par méthode. Si le code est « lourd », il est peut-être nécessaire de créer d’autres méthodes. |  |
| VLD est utilisé pour détecter les fuites de mémoires. |  |