

Cours C++

Conception d'un programme

Conception d'un programme

De manière générale en POO, le développeur procède de la façon suivante:

1. **Répertorier** les objets nécessaires à la description du problème,
2. **Réfléchir** aux liens logiques entre ces objets (composition, amitié, héritage),

Héritage vs. Composition

Il faut bien distinguer les deux liens logiques que sont l'héritage et la composition:

Composition relation de type **possède un**; la classe polygone possède un ensemble de point

Héritage relation de type **est un**; la classe rectangle est un polygone

Héritage

L'héritage est une notion relativement puissante en POO, mais il éviter les relations hiérarchiques inappropriées, et les dérivations qui, in fine, s'avèrent inutiles et alourdissent le code.

classe abstraite, classe concrète ? Dans un contexte donné, un objet de type `mammifere` va représenter un concept, quand un chat ou un chien va représenter un objet concret.

Est-il utile d'implémenter une classe `etre_vivant`, une classe dérivée `animal`, une classe `mammifere` dérivée de `animal`, les classes `chat` et `chien` dérivée de `mammifere`, des classes `doberman`, `chihuahua`, etc qui dérivent de la classe `chien` ? Quelles classes doivent être abstraites/concrètes ?

Une phase de réflexion sur le contexte et la finalité du projet est nécessaire pour déterminer la pertinence des objets à implémenter et de leur dérivation.

Conception d'un programme

De manière générale en POO, le développeur procède de la façon suivante:

1. **Répertorier** les objets nécessaires à la description du problème,
2. **Réfléchir** aux liens logiques entre ces objets (composition, amitié, héritage),
3. **Implémenter** ces objets *i.e.* déclarer les méthodes puis les définir,

Conception d'un programme

De manière générale en POO, le développeur procède de la façon suivante:

1. **Répertorier** les objets nécessaires à la description du problème,
2. **Réfléchir** aux liens logiques entre ces objets (composition, amitié, héritage),
3. **Implémenter** ces objets *i.e.* déclarer les méthodes puis les définir,
4. **Créer** des instance-tests de ces objets et **utiliser** leurs méthodes pour répondre au problème posé.

Conception d'un programme

	Fraction du temps
1. Répertorier / 2. Réfléchir	40%
3. Implémenter	10%
4. Tester & utiliser/déboguer	50%