Programmation « orientée système »

Approfondissements semaine 2

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

Objectifs de ces séances

☞ consolider votre apprentissage

Note : n'est pas du travail à double, mais (normalement) un moyen plus efficace de transmettre les/stabiliser vos connaissances

Plan :

- rappels des points clés
- études de cas (toutes propositions bienvenues) :
  - fonctions : passage des arguments, `const`
  - copié-collé

Rappels des points clés (2 dernières semaines)

- conseil (aux programmeurs Java) : slide 15 semaine passée :

  travaillez dès maintenant et suffisamment votre C en dépit des similarités de syntaxe

- sémantique de l'affectation

  en C : sémantique de valeur (copie)

- `const` (n'est pas pareil de `final`)

- **JAMAIS** de copié-collé :

  si une modification de votre code entraîne une autre modification de même nature, c'est qu'il y a « copié-collé »

- passage par valeur / par référence (comment faire en pratique)

- initialisation des variables

  (en C il n'y en a pas par défaut)

`const` **vs.** `final`

(Rappel : `const` veut dire « `read only` »)

- Java a une sémantique de référence :
  on **PEUT** modifier un objet `final` en Java (mais pas le ré-assigner) :

```
final Object a = b;
a.modifie(42); // est autorisé !!
a = c; // NON !
```

  `const` est plutôt comme une interface immutable (dont hériterait le type en question)

- `const` utilisé pour un paramètre de fonction n'a pas d'équivalent en Java

  (et de même on ne peut pas faire de tableau d'objets « `const` » en Java,
  à moins que le type des objets soit immutable en lui-même,
  mais ce n'est pas le mot-clé `final` qui permettrait de le faire)

Jean-Cédric Chappelier

const vs. final

- une variable `const` *doit* être initialisée,
  alors qu'un objet `final` peut être affecté une fois
  (rappel : Java initialise par défaut)

```
public class A {
   void f() {
     final int a;
     a = 42;
   }
}
```

- (mineur : `final` s'utilise pour plusieurs autres choses en Java)

Passage d'argument aux fonctions

Supposons que l'on veuille incrémenter une variable, à l'aide d'une fonction.

Comment faire ?

- solution « immutable » :
  argument non modifié, retourne une nouvelle valeur
- solution « in-place » :
  modifier l'argument

  ☞ Quelle valeur de retour ?

  Deux cas typiques :
  - soit la valeur modifiée : permet les appels enchassés :
    ```
    f( incremente(x) )
    ```
  - soit un code d'erreur (pas pertinent dans ce cas précis, mais très fréquent dans d'autres cas) :
    ```
    if (incremente(x) == CODE_ERREUR) { ... }
    ```


Jean-Cédric Chappelier

Révision de code

Que pouvons-nous améliorer dans le code proposé ?

- copiés-collés $\longrightarrow$ **JAMAIS !**
- test d'égalité avec des `double` $\longrightarrow$ **JAMAIS !**