Correction détaillée de l'exercice 1, programmation structurée

Ce document montre comment utiliser la programmation structurée pour trouver le code de l'exercice 1.

Programmation structurée

Introduction

Un programme peut comporter des milliers, et même des millions d'instructions. Impossible pour un développeur d'avoir une vision d'ensemble et de gérer cette complexité.

Il faut décomposer le programme en éléments plus simples à appréhender.

La programmation structurée permet de bien décomposer un programme complexe en parties plus simples à comprendre.

Principe

Un programme est décomposé en modules. Chaque module a une des structures de la programmation structurée

En plus de permettre une bonne décomposition, ce principe offre un bon guide lors de l'écriture du programme puisque les structures de la programmation structurée sont simples et peu nombreuses.

Pseudo-langage

La programmation structurée utilise un pseudo-langage qui se rapproche d'un langage informatique mais qui n'est pas aussi formalisé et qui évite les difficultés techniques liées à un langage particulier.

Quand on a décrit un algorithme avec la programmation structurée il faut ensuite le traduire dans le langage que l'on veut utiliser, par exemple le langage C ou le langage Java. Cette traduction est le plus souvent très simple.

Structures de la programmation structurée

Chaque module a une de ces 3 structures :

- suite de modules ;
- répétition ;
- alternative.

Suite de modules

Faire traitement1 puis traitement2

(peut se généraliser à une suite de plusieurs modules).

Quand on choisit cette structure pour décomposer un module, on doit spécifier ce que feront traitement1 et traitement2. « Spécifier » signifie dire exactement ce qui sera fait par les traitements mais ça n'est pas dire comment ça sera fait. Le « comment » sera donné par les décompositions

ultérieures des modules traitement1 et traitement2. C'est le principe des poupées russes : une poupée contient une autre poupée qui contient une poupée, etc. Dans la programmation structurée une « poupée » (un module) peut contenir plusieurs « poupées ».

Exemple :

La structure globale pour l'exercice 1 est une suite de module. On initialise la somme à 0, ensuite on traite tous les paramètres de la ligne de commande (si le paramètre est un nombre on l'ajoute à la somme sinon on affiche un message d'erreur), ensuite on affiche la somme.

Les modules 1 et 3 sont réduits à une seule instruction et il ne reste donc qu'à décomposer le module 2 (traitement de tous les paramètres).

Répétition

Répéter un traitement

Il y a 3 sortes de répétitions :

- répéter le traitement un certain nombre de fois (boucle for) ;
- répéter le traitement tant qu'une condition est remplie (boucle while) ;
- répéter le traitement jusqu'à ce qu'une condition de fin soit remplie (boucle do … while).

Comment choisir la bonne :

1. Sait-on combien de fois le traitement sera répété ? Si oui c'est le 1er choix.
2. Est-ce que le traitement peut ne jamais être exécuté ? Si oui c'est le 2ème choix ; sinon c'est le dernier choix.

Exemple :

La structure du module 2 de l'exercice est une répétition d'un module que nous appelerons « Traitement 1 paramètre ».

Sait-on dès l'écriture du code combien de fois « Traitement paramètres » ? Oui, autant de fois qu'il y a de paramètres dans la ligne de commande.

Nous choisissons donc la structure « for ». Si on suppose que tous les paramètres sont dans le tableau de String « args », le code est :

for (String parametre : args) répéter UnParamètre.

UnParamètre est le module qu'il faut répéter. Sa description est : si le paramètre est un nombre entier on l'ajoute à la somme sinon on affiche un message d'erreur.

Alternative

Si une condition est remplie faire traitement1 sinon faire traitement2

Evidemment traitement1 et traitement2 sont des modules qu'il faut spécifier et qui seront décomposés eux-mêmes selon les principes de la programmation structurée (« poupées russes »).

Exemple :

C'est évidemment la structure de « Traitement 1 paramètre » :

si le paramètre représente un nombre entier faire « ajouter le nombre à la somme »
sinon afficher « paramètre n'est pas un entier ».

Ecriture du code Java

La seule difficulté est la traduction de l'alternative car il va falloir utiliser une exception pour savoir si une String représente un nombre entier ou pas.

Le traitement normal est « on traduit la String en nombre entier et on l'ajoute à la somme » (en fait on aura ici une suite de 2 modules, chaque module étant composé d'une seule instruction). Le traitement de l'exception sera d'afficher le message « paramètre n'est pas un entier ».

Je vais écrire le code en suivant le cheminement de la programmation structurée.

```
public static void main(String[] args) {
  int somme = 0;
  Traiter tous les paramètres
  System.out.println("La somme est " + somme);
}
```

Décomposition suivante :

```
public static void main(String[] args) {
  int somme = 0;
  for (String parametre : args) {
    Traiter 1 paramètre
  }
  System.out.println("La somme est " + somme);
}
```

Décomposition suivante :

```
public static void main(String[] args) {
  int somme = 0;
  for (String parametre : args) {
    try {
      int n = Integer.parseInt(parametre);
      somme += n; // ou bien somme = somme + n;
    catch(NumberFormatException e) {
      System.out.println(arg + " n'est pas un entier ! ");
  }
  System.out.println("La somme est " + somme);
}
```

Et voilà, c'est fini !