Distributeur de boissons [ss11] - Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 16 mai 2018

Table des matières

Java - Distributeur de boissons (Solution)

Mots-Clés Algorithmes paramétrés, Analyse descendante ■
Requis Structures de base, Structures conditionnelles ■
Difficulté ●●○ (50 min) ■

Objectif
Cet exercice simule un petit distributeur de boissons. Il montre en quoi les procédures et fonctions sont l'outil même de l'**analyse descendante** et de la structuration des algorithmes. Il fournit aussi l'occasion de travailler sur la construction incrémentale des chaînes de caractères.

...(énoncé page suivante)...

1 Énoncé

On veut simuler un distributeur de boissons qui propose du *thé*, du *café* ou du *chocolat*. Chaque boisson peut être servie non sucrée, sucrée ou très sucrée, avec, en option, un supplément de lait :

- Le café et le thé coûtent 40 centimes, le chocolat 60 centimes.
- Une portion de sucre coûte 5 centimes (le choix boisson sucrée correspond à une portion de sucre et le choix boisson très sucrée à deux).
- Le supplément de lait coûte 15 centimes.

Contrairement à un vrai distributeur qui dispose de boutons de commande, le programme qui simule le distributeur doit, avec contrôle de saisie, récupérer les données au cours d'un dialogue. L'utilisateur devra entrer :

- Pour commander du café, le code « CAF ».
- Pour commander du chocolat, le code « CHO ».
- Et pour commander du thé le code « THE ».

L'utilisateur doit aussi donner le nombre de doses de sucre (0, 1 ou 2) et préciser s'il veut un supplément de lait (0 ou 1).

Une fois les données récoltées, le programme affiche deux phrases construites sur l'un des deux modèles suivants, toutes les combinaisons étant possibles :

- Une phrase qui annonce le prix à payer (en centimes) telle que :

```
Vous devez payer 60 centimes
```

- Une phrase qui décrit la boisson commandée telle que :

```
Votre cafe tres sucre sans lait est pret
```

Exemple

Voici un exemple de dialogue entre le distributeur et l'usager :

```
Choix de la boisson (CAF, THE ou CHO)? CAF
Nombre de doses de sucre (dans [0..2])? 2
Supplement de lait (dans [0..1])? 1
------
Vous devez payer 65 centimes
Votre cafe tres sucre avec lait est pret
```

Objectif

Simuler le distributeur en notant le statut des paramètres de chaque opération.

...(suite page suivante)...

2 Algorithmique, Programmation

2.1 L'algorithme principal

Cette simulation ne respecte pas complètement ce qui se passe dans un vrai distributeur puisqu'elle ne prend pas en compte la phase de paiement. Ici elle se compose en les trois étapes :

- La prise de commande.
- Le traitement de la commande.
- L'affichage des résultats.

Liste des variables

Trois variables sont nécessaires pour caractériser la boisson :

- Le code de la boisson (une chaîne de caractères : `"CAF"`, `"CHO"` ou `"THE"`)
- Le nombre de doses de sucre (un entier : 0, 1 ou 2)
- La présence ou non d'un supplément de lait (un entier : 0 ou 1)

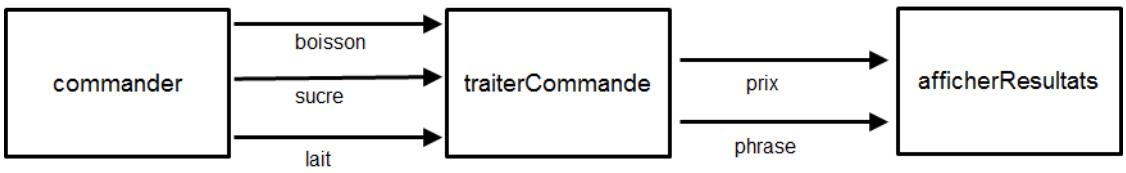
Pour les résultats, il faudra une variable entière pour le prix (estimé en centimes) ainsi qu'une chaîne de caractères pour la phrase décrivant la boisson.

Écrivez un programme qui déclare :

- Une chaîne `boisson` (code de la boisson).
- Un entier `sucre` (nombre de doses de sucre).
- Un entier `lait` (supplément de lait).
- Un entier `prix` (prix de la boisson).
- Une chaîne `phrase` (composition de la boisson).

Liste des opérations

Dans le schéma suivant, nous représentons les trois étapes réalisées par trois procédures ainsi que la circulation des informations :



- La procédure `commander` interroge l'utilisateur et produit les valeurs qui caractérisent la boisson. Elle les transmet à l'algorithme principal par ses paramètres de sortie.
- La procédure `traiterCommande` reçoit de l'algorithme les trois valeurs précédentes en paramètres d'entrée, les traite et produit le prix et la phrase de description de la boisson qu'elle transmet à l'algorithme sous forme de paramètres de sortie.
- La procédure `afficherResultats` reçoit de l'algorithme le prix et la phrase en paramètres d'entrée et les utilise pour communiquer les résultats en les affichant à l'utilisateur.

2.2 Saisie des données

On suppose que l'utilisateur donne des entrées valides, à savoir :

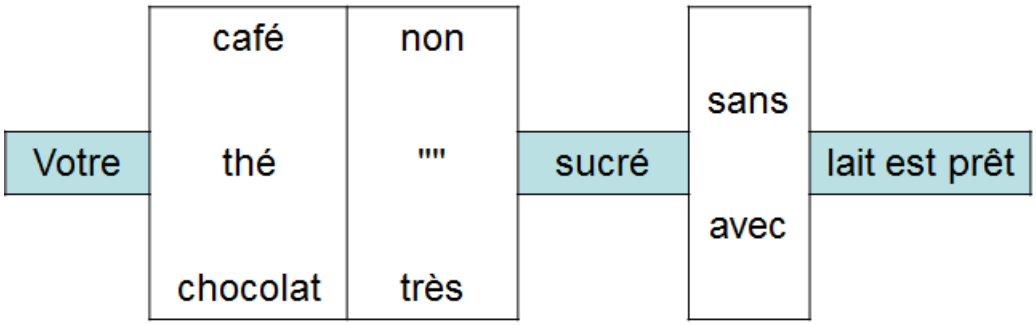
- Un code boisson parmi `CAF`, `THE`, `CHO`.
- Un entier compris entre 0 et 2 pour la dose de sucre.
- Un entier 0 ou 1 pour le supplément de lait.

Écrivez la saisie des données dans votre programme.
Affichez les invites :

```
Choix de la boisson?
Nombre de doses de sucre?
Supplement de lait?
```

2.3 Traitement de la commande

Analyse
La phrase est une chaîne de caractères construite en concaténant certains fragments constants et certains fragments qui dépendent des données. Voilà un schéma de la phrase à construire dans laquelle les fragments « variables » sont présentés dans les grands cadres blancs :



La phrase sera initialisée avec la chaîne `"Votre"`.

Une première conditionnelle va tester le code de la boisson et concaténer le nom de la boisson à la phrase qui contient `"Votre"`. En même temps le prix pourra être initialisé puisqu'il dépend de la même condition sur le code de la boisson.

Une deuxième conditionnelle, s'appuyant sur le nombre de doses de sucre, va prolonger la phrase avec le fragment adéquat concernant le sucre. Dans le cas de la boisson sucrée avec une dose, le fragment est une chaîne vide (`""`). Comme il y a trois cas correspondant à trois valeurs entières, on peut utiliser une instruction `Selon`.

Il faut ensuite concaténer la chaîne `"sucré"`.

Une troisième conditionnelle va ajouter à la phrase en construction celui des deux chaînes `"sans"` ou `"avec"` qui convient.

La phrase sera terminée en concaténant à la fin de la phrase : `"lait est prêt"`.

Notons qu'on aurait pu choisir de modifier le prix dans les deux dernières conditionnelles, mais il est aussi simple de le mettre à jour par un calcul puisque le type des variables du sucre et du lait est numérique.

Décomposition

Il est possible de scinder la procédure `traiterCommande` en deux procédures avec l'objectif de mieux spécialiser le rôle de chacune. C'est-à-dire qu'on peut écrire le corps de l'algorithme principal sous la forme :

```
Début
  commander(boisson,sucre,lait)
  traiterPrix(boisson,sucre,lait,prix)
  traiterPhrase(boisson,sucre,lait,phrase)
  afficherResultats(prix,phrase)
Fin
```

dont la première procédure `traiterPrix` calcule le prix et la deuxième `traiterPhrase` construit la phrase de description.

On constate alors que les deux procédures de traitement ont chacune trois paramètres entrants et un paramètre sortant. Elles peuvent donc être remplacées par des fonctions qui ont les mêmes paramètres entrants et qui renvoient la valeur portée par le paramètre sortant de la procédure. Dans ce cas, l'algorithme principal s'écrit :

```
Début
  commander(boisson,sucre,lait)
  prix <- prixBoisson(boisson,sucre,lait)
  phrase <- phraseBoisson(boisson,sucre,lait)
  afficherResultats(prix,phrase)
Fin
```

où la fonction `prixBoisson` renvoie le prix et `phraseBoisson` la phrase de description de la boisson.

L'inconvénient de ce choix est de re-exécuter, dans chaque fonction, la même structure conditionnelle sur le code de la boisson. En revanche un argument en faveur de ce choix est qu'il spécialise chaque fonction, mieux que ne le fait la procédure `traiterCommande` qui s'occupe à la fois de la construction de la phrase et du prix.

Selon cette analyse,

Écrivez une fonction `calcPrix(boisson,sucre,lait)` qui calcule et renvoie le prix, étant donnés le code de la boisson, le nombre de doses de sucre et le nombre de supplément de lait.

Puis écrivez une fonction `calcPhrase(boisson,sucre,lait)` qui calcule et renvoie la phrase, étant donnés le code de la boisson, le nombre de doses de sucre et le nombre de supplément de lait.

Validez vos opérations avec la solution.

Solution Java @[pgboissons.java]

```java
/**
  Prix de la boisson
  @param[in] boisson - code de la boisson: {"CAF","CHO","THE"}
  @param[in] sucre - nombre de doses de sucre (dans [0..2])
  @param[in] lait - nombre de supplément de lait (dans [0..1])
  @return le prix de la boisson commandée
*/

public static int calcPrix(String boisson, int sucre, int lait)
{
  int prix = 0;
  if (boisson.equals("CAF"))
  {
    prix = 40;
  }
  else if (boisson.equals("CHO"))
  {
    prix = 60;
  }
  else
  {
    prix = 40;
  }
  return (prix + sucre * 5 + lait * 15);
}
```

```java
/**
  Phrase décrivant la boisson
  @param[in] boisson - code de la boisson: {"CAF","CHO","THE"}
  @param[in] sucre - nombre de doses de sucre (dans [0..2])
  @param[in] lait - nombre de supplément de lait (dans [0..1])
  @return la phrase décrivant la boisson commandée
*/

public static String calcPhrase(String boisson, int sucre, int lait)
{
  String phrase = "Votre ";
  if (boisson.equals("CAF"))
  {
    phrase += "cafe ";
  }
  else if (boisson.equals("CHO"))
  {
    phrase += "chocolat ";
  }
  else
  {
    phrase += "the ";
  }
  switch (sucre)
  {
    case 0:
      phrase += "non ";
      break;
    case 1:
      phrase += "";
      break;
```

```
    case 2:
      phrase += "tres ";
      break;
  }
  phrase += "sucre ";
  phrase += (lait == 0 ? "sans " : "avec ");
  phrase += "lait est pret";
  return phrase;
}
```

Solution simple
Nous avons pris soin de toujours ajouter un espace à la fin de chaque chaîne (sauf la dernière) pour que les mots ne soient pas collés les uns aux autres.

2.4 Affichage des résultats

Écrivez une procédure `afficherResultats(prix,phrase)` où `prix` est le prix de la boisson commandée et `phrase` la phrase décrivant la boisson commandée. Affichez (où `[x]` désigne le contenu de `x`) :

```
Vous devez payer [prix] centimes
[phrase]
```

Validez votre procédure avec la solution.

Solution Java @[pgboissons.java]

```
/**
  Affichage des résultats
  @param[in] prix - prix de la boisson
  @param[in] phrase - phrase décrivant la boisson commandée
*/

public static void afficherResultats(int prix, String phrase)
{
  System.out.println("------");
  System.out.println("Vous devez payer " + prix + " centimes");
  System.out.println(phrase);
}
```

2.5 Vérification et jeux d'essais

Calculs et résultats
Pour vérifier les calculs, il faut passer au moins une fois dans chaque branche de chaque conditionnelle :

- Une fois pour chaque boisson.
- Une fois pour chaque quantité de sucre.

- Une fois pour la présence ou non de lait.

Le nombre de cas effectifs est :
3(choix de boissons) * 3(choix de sucre) * 2(choix de lait) = 18
Mais comme les conditionnelles sont consécutives, ce nombre de cas est réduit.

Testez. Le tableau ci-dessous indique une série de tests minimaux à réaliser :
(Rappel : on suppose les données valides)

Boisson	Sucre	lait	Prix	Phrase
CHO	0	1	75	Votre chocolat non sucré avec lait est prêt
CAF	1	0	45	Votre café sucré sans lait est prêt
THE	2	1	65	Votre thé très sucré avec lait est prêt

Validez votre programme avec la solution.

Solution Java @[pgboissons.java]

```
public static void main(String[] args)
{
  Scanner input = new Scanner(System.in);
  String boisson;
  System.out.print("Choix de la boisson (CAF, THE ou CHO)? ");
  boisson = input.next();
  int sucre;
  System.out.print("Nombre de doses de sucre (dans [0..2])? ");
  sucre = input.nextInt();
  int lait;
  System.out.print("Supplement de lait (dans [0..1])? ");
  lait = input.nextInt();
  int prix = calcPrix(boisson,sucre,lait);
  String phrase = calcPhrase(boisson,sucre,lait);
  afficherResultats(prix,phrase);
}
```

3 Que retenir de cet exercice?

Les procédures (et les fonctions) sont les outils fondamentaux de la méthode d'analyse descendante d'un problème. Pendant que l'on construit une solution, dès que l'on identifie une étape, on peut charger une procédure (ou une fonction) de s'occuper de cette tâche. Ces procédures (ou fonctions) seront analysées dans un deuxième temps, en suivant éventuellement le même processus. On peut ainsi écrire chaque niveau de l'algorithme principal sans se perdre dans les détails du niveau suivant.

Dès qu'une procédure ne comporte que des paramètres entrants et un unique paramètre sortant, elle est théoriquement équivalente à une fonction ayant les mêmes paramètres d'entrée et renvoyant la valeur qui passait par le paramètre de sortie. Lorsque les deux

choix sont possibles, d'autres arguments liés au contexte ou au langage de programmation envisagé, peuvent faire pencher en faveur d'une fonction ou d'une procédure.

4 Références générales

Comprend [Tartier-AL1 :c7 :ex27] ■