UNIVERSITÉ LIBRE DE BRUXELLES ULB

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Langages de programmation 2 : Projet Java

Yves Roggeman *

INFO-F202 — Année académique 2018–2019

Résumé

Ceci est le second énoncé de l'épreuve de première session qui se déroule en janvier. Il sert de base à l'épreuve orale ; l'évaluation finale porte sur l'acquisition des concepts démontrée à cette occasion. Le but de cet exercice de programmation est donc de démontrer une connaissance approfondie et un usage adéquat des constructions du langage de programmation Java, en particulier de l'usage du parallélisme (à l'aide de *threads*) et des classes génériques.

Le problème posé consiste à écrire un algorithme efficace déterminant si une valeur d'un tableau non trié y est présente au moins une deuxième fois ; le résultat est donc un simple booléen. Cette recherche est normalement de complexité linéaire en la taille du tableau, mais elle peut être fortement accélérée si le traitement se fait en parallèle simultanément sur plusieurs sous-tableaux.

1 La méthode

Le tableau est une simple structure homogène à n composantes indicées de 0 à $n-1$, donc un *array* au sens Java. Le type de ses composantes doit être paramétrique ; il doit simplement accepter les tests d'égalité et d'inégalité.

L'élément pour lequel il faut rechercher un doublon est identifié par son indice au sein de ce tableau.

La recherche s'effectue en parallèle sur K sous-tableaux de taille n/K, certains ayant une composante de plus que d'autres si K n'est pas un diviseur de n. Un sous-tableau est évidemment un sous-ensemble contigu d'éléments du tableau donné.

L'algorithme lance donc K recherches partielles, mais tous ces threads doivent s'arrêter dès que l'un d'entre eux a trouvé un doublon. Chacun effectue par contre une simple recherche linéaire dans l'étendue qui le concerne. Un seul devra éviter de s'arrêter sur l'élément recherché : celui dont l'étendue couvre son indice.

On suppose, pour que cet algorithme soit efficace, que le tableau est grand (n est grand) et que la probabilité qu'un élément y soit répété est très grande, puisque la détection d'absence de doublon reste linéaire.

2 Implantation et tests

Il est demandé de réaliser ce parallélisme exclusivement à l'aide des constructions de base du langage (celles d'origine, présentes dans le package « `java.lang` »).

Cette recherche de doublon est une simple méthode statique `isDuplicate` à valeur booléenne qui reçoit comme seuls paramètres le tableau, l'indice de l'élément cherché et le nombre de processus parallèles K à lancer. Ce dernier paramètre peut être omis et vaut alors 4.

*Université libre de Bruxelles (ULB) <yves.roggeman@ulb.ac.be>

Il vous est demandé d'écrire en Java la définition de la ou des classes nécessaires et de toutes leurs méthodes dont vous auriez besoin, ainsi qu'un programme principal servant de test pour différentes valeurs des paramètres et différents types de base, en n'oubliant pas les cas limites ou particuliers. Respectez bien les principes de base de l'encapsulation et, plus généralement, de la programmation orientée objet.

3 Remise et évaluation

L'évaluation portera essentiellement sur la pertinence des choix effectués dans l'écriture : la codification, la présentation et l'optimisation du programme justifiées par la maîtrise des mécanismes mis en œuvre lors de la compilation et l'exécution du code. D'une manière générale, le respect strict des directives, la concision, la précision, la lisibilité (clarté du texte source), l'efficacité (pas d'opérations inutiles ou inadéquates) et le juste choix des syntaxes typiques de Java seront des critères essentiels d'appréciation. De brefs commentaires dans le code source sont souhaités pour éclairer les choix de codification.

Votre travail doit être réalisé pour le vendredi 21 décembre 2018 à 10 heures au plus tard. Vous remettrez tout votre travail empaqueté en un seul fichier compacté (« .zip » ou autre) *via* le site du cours (INFO-F202) sur l'Université Virtuelle (https ://uv.ulb.ac.be/). Ceux-ci devront contenir en commentaire vos matricule, nom et prénom. Le jour de l'examen, vous viendrez avec une version imprimée — un *listing* — de ces divers fichiers. Une impression du résultat d'une exécution du programme est également demandée.