Examen de Programmation Objet (Java)
2022–2023

— Durée : 2h
— Type : papier
— Aucun document n'est autorisé.
— Toutes vos affaires (sacs, vestes, *etc.*) doivent être placées à l'avant de la salle.
— Aucun téléphone ne doit se trouver sur vous ou à proximité, même éteint.
— Les déplacements et les échanges ne sont pas autorisés.
— Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.

Question de cours (4pts)

1. Citez les classes sur lesquelles reposent l'architecture d'une application JavaFX. (1pt)
2. Écrivez un code Java qui ajoute à un bouton un gestionnaire d'événement affichant dans la console `Bien joué !` lorsque l'utilisateur clique dessus. (1.5pts)
3. Décrivez schématiquement le principe du modèle abstrait d'architecture logicielle pour les interfaces homme-machine vu en cours. (1.5pts)

Problème (16pts)

1. Étant donné une liste d'itérateurs sur des entiers, nous souhaitons obtenir un itérateur produisant la somme (ou le produit) des entiers produits individuellement par chacun de ces itérateurs. Dès que l'un d'entre eux ne produit plus d'entiers, l'itérateur n'en produit plus non plus. La méthode `remove()` de cet itérateur lèvera une exception non vérifiée de type `OpérationNonSupportéeException` (à écrire) pour signaler qu'elle n'est pas implémentée.
   (a) Écrivez des classes `AdditionIterator` et `MultiplicationIterator` permettant de créer de tels itérateurs pour répondre au problème, en évitant la duplication de code. (10pts)
   (b) Écrivez une méthode `main` qui crée une instance d'`AdditionIterator` et affiche dans la console tous les entiers produits par cet itérateur. (1pt)
2. L'addition et la multiplication ne sont pas les seules opérations que nous pourrions vouloir utiliser pour combiner les éléments de plusieurs itérateurs sur des entiers. Par exemple, d'autres opérations arithmétiques, comme la soustraction ou la division entière, pourraient aussi être utilisées, et plus généralement n'importe quelle fonction binaire sur les entiers. Pour cela, nous souhaitons séparer l'implémentation de l'opération de combinaison de celle de l'itérateur, afin de pouvoir les modifier indépendamment l'une de l'autre.
   (a) Écrivez une interface `FonctionBinaire` pour représenter des fonctions binaires sur les entiers. (1pt)
   (b) Écrivez des classes `Addition` et `Multiplication` qui réalisent l'interface `FonctionBinaire`. (0.5pt)
   (c) Écrivez une classe `CombinIterator` permettant de généraliser les classes `AdditionIterator` et `MultiplicationIterator` à une opération de combinaison quelconque. (3pts)
   (d) Écrivez un code client qui crée une instance de `CombinIterator` avec la fonction binaire `Multiplication` pour combiner les éléments de plusieurs itérateurs sur des entiers. (0.5pt)

Extrait de la javadoc de l'interface `java.util.Iterator<E>`

— `boolean hasNext()` Returns true if the iteration has more elements.

— `E next()` Returns the next element in the iteration.

— `void remove()` Removes from the underlying collection the last element returned by this iterator (optional operation).