Université Denis Diderot - DEUG MIAS/MASS - IF121
Examen du 17/01/2005 - Durée : 3 heures - Aucun document n'est autorisé

Exercice 1.

Rappels préliminaires :

- étant donnés deux points A et B de coordonnées respectives (x_A, y_A) et (x_B, y_B), le vecteur $\overrightarrow{AB}$ a pour coordonnées $x_B - x_A$ et $y_B - y_A$;
- deux vecteurs $\vec{u}$ et $\vec{v}$ de coordonnées (u, u') et (v, v') sont colinéaires si et seulement si $u \times v' = u' \times v$;
- pour montrer que trois points A, B et C sont alignés, il suffit de montrer que deux des vecteurs construits à partir de ces points (par exemple $\overrightarrow{AB}$ et $\overrightarrow{AC}$) sont colinéaires.

1.1. Écrire en Java une fonction ayant en arguments les coordonnées entières de trois points et renvoyant une valeur booléenne indiquant si les trois points sont alignés ou non.

1.2. Écrire en Java une fonction recevant en arguments deux tableaux x et y à une dimension contenant les coordonnées entières de points du plan (le i-ème point a comme coordonnées $(x[i], y[i])$) et renvoyant le nombre de points alignés avec les deux premiers (c'est-à-dire ceux de coordonnées $(x[0], y[0])$ et $(x[1], y[1])$).
La fonction vérifiera que les deux tableaux ont la même taille supérieure ou égale à 2 et renverra -1 si ce n'est pas le cas.

1.3. Ce qui a été fait pour des coordonnées entières peut-il être utilisé directement en changeant simplement le type `int` en `float` pour traiter des coordonnées réelles?

Exercice 2.

2.1. Un diviseur strict d'un entier $n > 0$ est un entier > 0, différent de n, et qui le divise (reste de la division entière nul).

2.1.1. Donner une borne supérieure raisonnable du plus grand diviseur strict d'un entier $n > 0$.

2.1.2. Décrire de manière informelle (c'est-à-dire dans un langage différent de Java) un algorithme permettant d'afficher tous les diviseurs stricts d'un entier $n > 0$ donné.

2.1.3. Écrire en Java une fonction qui, étant donné un nombre entier strictement positif n, renvoie un tableau d'entiers contenant exactement la liste de ses diviseurs stricts (et ayant exactement comme taille le nombre de ses diviseurs). On pourra commencer par utiliser un tableau de taille plus (et suffisamment) grande pour mémoriser les diviseurs, puis une fois le nombre de diviseurs connu, construire le tableau de taille exacte (sans pour autant refaire tous les calculs déjà faits). Si n est ≤ 0, la fonction renvoie `null`.

2.1.4. Écrire en Java une fonction qui, étant donné un nombre entier $n > 0$, renvoie la somme de ses diviseurs stricts et -1 si $n \leq 0$.

2.2. Deux nombres entiers positifs a et b sont dits amis si a est égal à la somme des diviseurs stricts de b et b est égal à la somme des diviseurs stricts de a.
Ainsi, 6 est ami de lui-même : ses diviseurs stricts sont 1, 2 et 3 dont la somme est 6 (un tel nombre est dit parfait). Les nombres 220 et 284 sont deux nombres amis différents : les diviseurs stricts de 220 sont 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 et 110 dont la somme est 284 et ceux de 284 sont 1, 2, 4, 71 et 142 dont la somme est 220.

2.2.1. Écrire en Java une fonction qui, étant donnés deux nombres entiers > 0, renvoie la valeur booléenne `true` s'ils sont amis et `false` dans le cas contraire (en particulier si au moins l'un au moins des nombres est ≤ 0).

2.2.2. Écrire la fonction `main` d'une application Java qui, après avoir lu deux nombres entiers, affiche un message indiquant si ces deux nombres sont ou non amis.

Exercice 3.

Écrire en Java une fonction qui, étant donnés un tableau t de nombres entiers à 2 dimensions et un nombre entier n, renvoie :

- la valeur `null` si le tableau donné contient (au moins) un nombre x tel que $x < 0$ ou $x > n$;
- un tableau tt de $n+1$ entiers tel que $tt[i]$ soit égal au nombre d'éléments de t égaux à i si le tableau ne contient que des nombres compris, au sens large, entre 0 et n.

———————

Exercice 4.

Un échiquier est un carré constitué de 64 cases. Dans cet exercice, l'échiquier sera sous-jacent et ne sera pas matérialisé par un tableau.

On s'intéresse ici à trois pièces particulières de ce jeu : les cavaliers représentés symboliquement par le caractère `'C'`, les fous représentés symboliquement par le caractère `'F'` et les tours représentées symboliquement par le caractère `'T'`.

Les mouvements possibles de chacune de ces pièces sur l'échiquier, supposé par ailleurs vide de toute autre pièce, et la manière dont les cases de l'échiquier sont repérées (dans cet exercice) sont résumés sur la figure donnée en annexe du sujet.

Écrire en Java une fonction qui, étant donnés deux positions sur l'échiquier et un caractère désignant une des pièces précédentes, renvoie la valeur booléenne `true` si la deuxième position est différente de la première et est accessible, en un seul mouvement, par la pièce spécifiée à partir de la première position. Sinon, la fonction renverra la valeur `false`.

———————

Exercice 5.

5.1. Écrire le code d'une classe `ChequeAuPorteur` encapsulant un entier correspondant au montant du chèque et possédant un constructeur unique à un paramètre entier correspondant au montant du chèque créé lors d'une instanciation de la classe.

5.2. On souhaite définir une classe `Compte` correspondant à un compte bancaire. Un objet de cette classe encapsulera deux entiers correspondant respectivement au solde courant du compte et au découvert autorisé et un booléen permettant le blocage du compte du point de vue des retraits.

5.2.1. Écrire le code de cette classe contenant :

- les différentes variables encapsulées privées (variable booléenne initialisée à `false`) ;

- un constructeur ayant un entier en paramètre : cet entier définit le dépôt initial sur le compte, le découvert autorisé étant égal à 0 ;

- un constructeur ayant deux entiers en paramètres : le premier est le dépôt initial réalisé sur le compte et le second le découvert autorisé ;

- un constructeur ayant un premier paramètre du type `ChequeAuPorteur` et un second de type entier en paramètres : le montant du chèque constituera le solde initial du compte créé et le second fixera le découvert autorisé ;

- une méthode renvoyant le solde du compte sur lequel elle est invoquée ;

- une méthode permettant de déposer une somme d'argent sur le compte sur lequel elle est invoquée et débloquant un compte bloqué si son solde devient positif. La fonction renverra le nouveau solde du compte ;

- une méthode permettant de retirer immédiatement une somme d'argent (un entier strictement positif) sur le compte sur lequel elle est invoquée. Si ce retrait est possible (compte non bloqué et nouveau solde compatible avec le découvert autorisé), la fonction renverra la valeur de son paramètre et 0 si le retrait n'est pas possible.

- une méthode réalisant un débit différé d'une certaine somme. Le débit du compte se fera toujours (même si le compte est bloqué) mais bloquera le compte s'il provoque un dépassement du découvert autorisé. La fonction renverra, en toutes circonstances, le nouveau solde du compte.

5.2.2. Écrire une application utilisant cette classe dont la fonction `main` :

- crée un compte `c1` avec le premier constructeur et l'entier 1000 ;
- crée un compte `c2` avec le second constructeur et les entiers 2000 et 500 ;
- crée un compte `c3` avec le troisième constructeur après avoir créé un chèque au porteur d'une valeur de 1200 utilisé en premier paramètre et avec un découvert autorisé de 150 ;
- illustre et permet de réaliser le test des différentes situations possibles par une suite d'opérations de dépôt, de retrait (immédiat ou différé) et de consultation sur les comptes précédents.