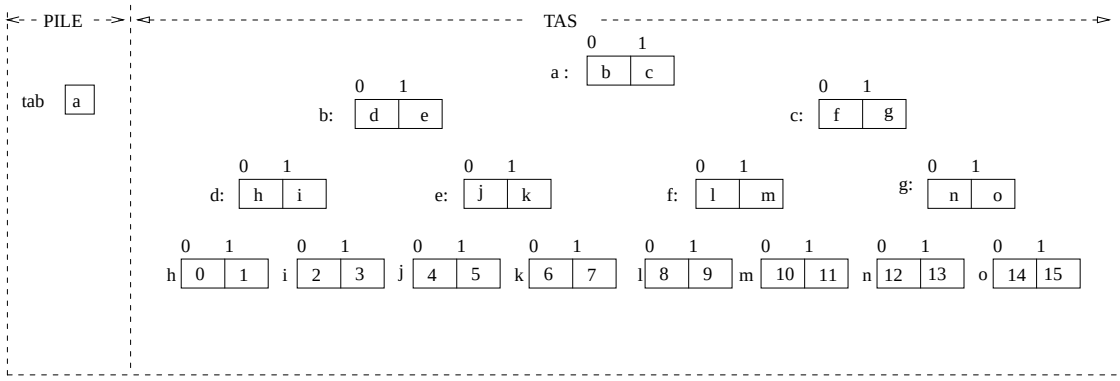
Université Paris Diderot
Concepts Informatiques
Durée : 2 heures 30 minutes
Documents, calculatrices, ordinateurs et téléphones interdits
Examen du 22 mai 2012

Exercice 1. On suppose que l'organisation des données en mémoire (pile et tas) d'une application correspond au schéma suivant où a, b, $\ldots n$, o sont des références de tableaux. Les références h, $\ldots o$ sont des références de tableaux d'`int`.
La variable `tab` est une variable de type référence de tableau et c'est la seule variable de type référence de tableau dans le programme..



1. Comment la variable `tab` a-t-elle dû être déclarée dans le programme ?
2. Ecrire une séquence de code Java conduisant à cette organisation de la mémoire.
3. Donner une expression utilisant la variable `tab` dont la valeur est l'entier `7`.
   Même question pour l'entier `11`.
   Même question pour une valeur n quelconque comprise entre 0 et 15.
4. Parmi tous les emplacements en mémoire, un seul ne peut pas être modifié en Java au travers d'une méthode sans valeur de retour.
   Lequel (expliquer pourquoi) ?
5. Ecrire une méthode Java sans valeur de retour qui permettrait, au travers de ses paramètres, de désigner et d'augmenter d'une certaine quantité la valeur de l'un des 16 éléments des tableaux d'`int` du tas.
   Quel serait alors l'appel à réaliser pour changer la valeur 11 en 111 ?

Exercice 2. On définit la fonction récursive suivante :

$$\forall n, m \in \mathbb{N}^+,\ f(n,m) = \begin{cases} n & \text{si } n = m \\ f(n-m, m) & \text{si } m < n \\ f(m-n, n) & \text{sinon} \end{cases}$$

1. Calculer à la main la valeur de $f(1,5)$ et $f(24,30)$, en détaillant les valeurs intermédiaires.
2. Ecrire en Java une méthode récursive dont le calcul conduira à produire la valeur de la fonction décrite ci-dessus.
3. Traduire tel qu'étudié en cours et TD le programme Java précédent en un programme ne contenant qu'une seule boucle `while` contenant elle-même un unique `switch` dans lequel les `case` ne contiennent que des instructions simples (affectations, opérations arithmétiques, tests élémentaires) et qu'un seul tableau d'entiers représentant la mémoire utilisée par la fonction précédente. Il est par ailleurs permis d'utiliser une `Pile` mais on peut s'en passer. Pourquoi ?

Exercice 3. Vous disposez d'un ensemble de N objets que vous devez placer dans un conteneur de volume V et pouvant supporter un poids maximal P. Pour chaque objet (numéroté de 0 à $N-1$) son volume et son poids sont donnés dans des tableaux d'entiers $volume$ et $poids$. Les objets sont par ailleurs supposés malléables et peuvent donc prendre la forme qu'il convient pour que les volumes s'ajoutent purement et simplement. Des objets peuvent avoir, le cas échéant, les mêmes attributs (même poids et même volume).
L'objet de cet exercice est de rechercher toutes les combinaisons d'objets optimisant l'utilisation du conteneur, c'est-à-dire la recherche de sous-ensembles d'objets dont la somme des poids et celle des volumes sont égales au poids et au volume autorisés pour le conteneur.

1. Combien de candidats-solutions une recherche par la force brute devrait-elle tester et donc générer?

2. Ecrire un programme réalisant la recherche de solutions par la force brute, c'est-à-dire générant et testant systématiquement tous les candidats-solutions potentiels.

3. Soit l'ensemble d'objets correspondant au tableau suivant :

	0	1	2	3	4
volume	3	2	1	3	2
poids	2	1	2	1	2

   Pour un volume total autorisé V égal à 4 et un poids total autorisé P égal à 3, décrire sous forme d'arbre le sous-espace de candidats-solutions qui serait exploré avec une approche *backtracking*.

4. Donner une version récursive d'un programme qui fournirait toutes les solutions par *backtracking*.
   On supposera que les données du problème sont des variables (poids total P, volume total V et tableaux $poids$ et $volume$ des caractéristiques des objets).
   Préciser quel appel permet de générer toutes les solutions.