Université Paris XII
IUT de Fontainebleau
Département informatique
FI-2
2001-2002

PARTIEL 2

Seuls sont autorisés, à titre de documents, les listings comprenant explicitement le nom de l'étudiant (et celui-ci seulement) et les notes manuscrites.

Exercice 1.- (**Événements souris**)

Écrire une application Java permettant de faire apparaître un cadre à l'écran dans lequel :

- sur une ligne est indiqué le dernier événement souris ainsi que les coordonnées du pointeur de celle-ci à ce moment-là ;
- un rectangle de hauteur 30 pixels et de largeur 20 pixels apparaît lorsqu'on clique sur la souris dans la partie active du cadre, l'emplacement du pointeur de la souris étant le sommet en haut à gauche de celui-ci.

Exercice 2.- (**Complexité du tri à bulle**)

Nous avons analysé la complexité de deux algorithmes de tri, le tri-insertion et le tri-fusion, donnant $O(n^2)$ et $O(n.log(n))$, ce qui montre qu'il vaut mieux utiliser la deuxième méthode. Le tri à bulle, quant à lui, semble rapide sur des tableaux presque triés.

- 1^o) Rappeler l'algorithme du tri à bulle en écrivant une méthode Java permettant de trier un tableau d'entiers.

- 2^o) Analyser le coût de cet algorithme en déterminant le nombre maximum de comparaisons nécessaires pour trier un tableau de n éléments. On donnera le nombre exact de comparaisons, au vu du programme écrit ci-dessus, puis l'ordre de grandeur en O.

Exercice 3.- (**Serveur bancaire**)

- 1^o) Concevoir une classe Java `CompteBancaire` possédant une donnée de type entier et comme méthodes le constructeur par défaut, un constructeur dont le seul paramètre est un entier, une méthode `deposer` dont le paramètre est un entier, une méthode `retirer` dont le paramètre est un entier et enfin une méthode `solde` sans paramètre et renvoyant un entier.

Lors de la création d'un nouveau compte avec le constructeur par défaut, le montant sera zéro, et la somme indiquée pour l'autre constructeur.

- 2^o) Concevoir une classe Java `Banque` possédant comme donnée un tableau de comptes bancaires et comme méthodes un constructeur ayant un paramètre entier (représentant le nombre de comptes bancaires, tous initialisés à zéro), une méthode `deposer` avec deux arguments entiers (le numéro de compte et le montant déposé), une méthode `retirer` avec également deux arguments entiers et une méthode `solde`, renvoyant un entier, ayant un argument entier (le numéro de compte).

- 3^o) Concevoir une classe Java `BanqueServeur` qui implémente le serveur de consultation des comptes bancaires d'une banque. Ses données sont un socket serveur, évidemment, et une banque possédant dix comptes bancaires. Le numéro de port du serveur est 8888.

Le serveur se contentera de lancer des processus du type `BanqueService` (implémenté ci-dessous), dont le constructeur demande un socket et une banque.

- 4^o) Concevoir une classe Java de processus légers `BanqueService` qui implémente les services demandés aux serveur. Les requêtes du client sont de l'un des quatre types suivants :

```
SOLDE n
DEPOSER n m
RETIRER n m
QUIT
```

où `n` et `m` sont des entiers représentant le numéro de compte et le montant. Le serveur renvoie systématiquement le solde après l'opération effectuée (dans les trois premiers cas) ou quitte la connexion (dans le dernier cas).

Une requête étant compliquée par le fait qu'il peut y avoir plusieurs paramètres, on a intérêt à la traiter dans une méthode à part, disons :

```
String executeCommande(String )
```

en utilisant la classe `StringTokenizer` (voir la documentation).

On ne cherchera pas à synchroniser.

- 5^o) Concevoir une classe Java `BanqueClient` qui permet de saisir l'une des quatre requêtes ci-dessus au clavier, qui l'envoie au serveur et qui affiche la réponse. L'appel d'un client se fera sur l'ordinateur sur lequel travaille le serveur.

DOCUMENTATION

1 Événements souris

Pour pouvoir interagir avec les événements souris, il faut implémenter l'interface `MouseListener`. Elle comprend cinq méthodes dont on surcharge celles qui nous intéressent :

```
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseClicked(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

qui activent les réactions lorsqu'on a appuyé sur un bouton, lorsqu'on a relaché le bouton, lorsqu'on a cliqué, lorsque le curseur est entré dans le composant ou lorsqu'il sort du composant.

Lors d'un événement souris, on peut récupérer l'abscisse et l'ordonnée du curseur (dans les coordonnées de la fenêtre activée) grâce aux méthodes :

```
int getX()
int getY()
```

de la classe `MouseEvent`.

2 Rectangle

On peut dessiner un rectangle grâce à la méthode :

```
drawRect(int, int, int, int)
```

de la classe `Graphics`, dont les paramètres sont l'abscisse et l'ordonnée du sommet en haut à gauche du rectangle ainsi que la largeur et la hauteur du rectangle.

3 Tokenisation

Lorsqu'on veut analyser une commande obtenue sous forme d'une phrase, c'est-à-dire d'une chaîne de caractères, chaque mot représente en général un opérateur ou un opérande de cette commande. On veut donc décomposer cette phrase en mots, ou plus exactement en **tokens** (terme introduit par Peirce à la fin du XIX^{e} siècle).

La classe `StringTokenizer` du paquetage `java.util` nous aide pour cela. Elle comprend un constructeur dont le paramètre est un objet de la classe `String`. La méthode `nextToken()` de cette classe donne le token suivant, qui est un `String`.