Université Paris XII
IUT de Sénart-Fontainebleau
Département Informatique
S3 - APL
2016–2017

PARTIEL 1

Seuls sont autorisés, à titre de documents, les programmes imprimés comprenant explicitement le nom imprimé de l'étudiant sur chaque page (et celui-ci seulement), un extrait de table Unicode et les notes manuscrites.

Les exercices sont indépendants mais ils doivent être présentés dans l'ordre (prévoir deux pages par exercice).

Exercice 1.- (**Ordre de mérite**)

Un enseignant corrige les copies d'une promotion, entre le nom et la note correspondante de chaque étudiant et veut voir afficher les noms par ordre de mérite : un nom par ligne, le nom de létudiant ayant obtenu la meilleure note en premier.

Une session est par exemple :

```
Nom :  Bernard
Note :  12
Nom :  Arthur
Note :  15
Nom :  Paul
Note :  14
Nom :  Etienne
Note :  13
Nom :

L'ordre de mérite est :
Arthur
Paul
Etienne
Bernard
```

- 1^o) Concevoir une classe Java `Etudiant` ayant un attribut chaîne de caractères, le nom, et un attribut entier, la note, un constructeur à deux arguments et des accesseurs pour récupérer la valeur de chacun des attributs.

- 2^o) Écrire une application Java répondant au problème ci-dessus, sachant qu'une promotion ne comporte jamais plus de 200 étudiants.

[*On pourra utiliser un tableau d'*`Etudiant` *que l'on triera, grâce à un tri à bulle, selon le deuxième attribut.*]

Exercice 2.- (**Intersection de droites**)

Écrire une application Java qui fait apparaître un cadre fermant. Un texte dit « Entrer une première droite ». Lorsqu'on clique sur deux points de la fenêtre active avec la souris, le segment de droite déterminé par les deux points est affiché et le texte devient « Entrer une seconde droite ». Lorsqu'on clique sur deux autres points de la fenêtre active avec la souris, le nouveau segment de droite déterminé par ces deux derniers points est affiché, le texte devient « Les coordonnées de l'intersection sont : » et, à l'intersection de ces deux droites, un texte donne ces coordonnées sous la forme `(1.23, 2.345)`.

[Pour ceux qui ne sont pas à l'aise en Géométrie analytique, on a , avec des notations intuitives :

$$\Delta = (y_2 - y_1)(x_3 - x_4) - (x_1 - x_2)(y_4 - y_3)$$

Si Δ est non nul, on a :

$$\Delta.x = (x_3 - x_4)(x_1y_2 - x_2y_1) - (x_1 - x_2)(x_3y_4 - x_4y_3)$$

$$\Delta.y = (y_2 - y_1)(x_3y_4 - x_4y_3) - (y_4 - y_3)(x_1y_2 - x_2y_1)]$$

Documentation Événements souris

Les événements souris ont pour classe d'écoute :

```
public interface MouseListener
   {
   void mousePressed(MouseEvent event);
   // Appelée lorsqu'on a appuyé sur un bouton de souris
   void mouseReleased(MouseEvent event);
   // Appelée lorsqu'on a relâché un bouton de souris
   void mouseClicked(MouseEvent event);
   // Appelée lorsqu'on a cliqué sur un bouton de souris
   void mouseEntered(MouseEvent event);
   // Appelée lorsque la souris pénètre dans un composant
   void mouseExited(MouseEvent event);
   // Appelée lorsque la souris quitte un composant }
```

dont **toutes** les méthodes doivent être surchargées.

La classe `MouseEvent` possède les deux méthodes :

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

qui renvoient l'abscisse et l'ordonnée (en pixels) du curseur de la souris au moment de l'occurrence de l'événement.

La mise à l'écoute d'un événement souris sur un composant s'effectue grâce à la méthode :

```
addMouseListener()
```