Université Paris XII
IUT de Sénart-Fontainebleau
Département Informatique
S3 - APL
2015–2016

PARTIEL 1

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

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

Exercice 1.- (**Crible d'Ératosthène**)

Le crible d'Ératosthène *est une méthode pour obtenir la liste des nombres premiers inférieurs à un entier naturel n donné. Il consiste à écrire les entiers naturels compris entre 2 et n, puis à barrer ceux qui sont divisibles par les multiples de 2 sauf 2 lui-même (4, 6, 8, 10, ...), puis ceux qui sont divisibles par les multiples de 3 sauf 3 lui-même, et ainsi de suite jusqu'aux multiples de* $\sqrt{n}$*. Les entiers non barrés représentent l'ensemble recherché.*

- 1^o) Concevoir une classe Java **IntSet** d'ensembles d'entiers. L'attribut sera un `TreeSet`. Les méthodes seront :

- le constructeur par défaut qui crée un ensemble vide ;
- `add(int x)` qui ajoute `x` s'il n'est pas présent ;
- `remove(int x)` qui enlève `x` s'il est présent ;
- `print()` pour afficher l'ensemble des éléments de l'ensemble (entourés par des accolades et séparés par des espaces).

- 2^o) Écrire un programme Java qui demande un entier naturel n et qui affiche l'ensemble des nombres premiers inférieurs à n.

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

Vous voulez aider les lycéens qui ont à calculer les coordonnées des points d'intersection de deux cercles. Pour cela vous allez écrire une application qui demande à l'utilisateur les coordonnées du centre et le rayon d'un premier cercle, puis les coordonnées et le rayon d'un second cercle, qui dessine ces deux cercles dans un cadre à l'écran et qui, enfin, reporte les coordonnées du point sur lequel on clique avec la souris (ce qui donne une valeur approchée de ce qui est demandé).

Écrire une application Java répondant à ce cahier des charges.

[*On laissera la possibilité de cliquer autant de fois que nécessaire. On terminera abruptement en tuant le processus.*]

Documentation

1 Ensembles

La classe :

`TreeSet`

du paquetage :

`java.util`

possède un constructeur par défaut pour créer un ensemble vide d'objets et trois méthodes :

```
add(Object)
remove(Object)
iterator()
```

pour insérer et retirer un élément ainsi que pour obtenir un itérateur.

Attention ! Rappelons que le type `int` n'est pas une classe et donc ne dérive pas de la classe `Object`.

2 Itérateurs

La classe :

`Iterator`

du paquetage :

`java.util`

permet de parcourir un ensemble (entre autres) grâce aux deux méthodes :

```
boolean hasNext()
Object next()
```

qui dit s'il y a encore des éléments non parcourus et qui en choisit un s'il y en a (en tant qu'objet de la classe `Object`, qu'il faut donc convertir en tant qu'objet de la classe voulue).

3 Ellipse

La classe :

`Ellipse2D.Double`

du paquetage :

`java.awt.geom`

a pour constructeur :

```
Ellipse2D.Double(Double x, Double y, Double hauteur,
Double largeur)
```

où x et y désignent les coordonnées du coin inférieur gauche du rectangle de hauteur `hauteur` et de largeur `largeur` dans lequel l'ellipse est inscrite.

Si `hauteur = largeur`, on a un cercle.

La méthode :

`draw(Geom)`

de la classe abstraite :

`Graphics2D`

(la seule façon d'en obtenir une instance est de convertir de façon explicite un objet de la classe abstraite `Graphics`) permet de dessiner une ellipse.

4 Événements souris

Les événements souris ont pour classe d'écoute :

```
public interface MouseListener
{
  void mousePressed(MouseEvent event);
    // Appelee lorsqu'on a appuye sur un bouton de souris
  void mouseReleased(MouseEvent event);
    // Appelee lorsqu'on a relache un bouton de souris
  void mouseClicked(MouseEvent event);
    // Appelee lorsqu'on a cliqu sur un bouton de souris
  void mouseEntered(MouseEvent event);
    // Appelee lorsque la souris penetre dans un composant
  void mouseExited(MouseEvent event);
    // Appelee lorsque la souris quitte le 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'événement.

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

`addMouseListener()`