UFR IM²AG

UNIVERSITÉ Grenoble Alpes

INF362 - Examen de seconde session
Année 2015-2016

Durée : 2h.
Tout document interdit. Calculatrices interdites.
Pour chaque question, une partie des points peut tenir compte de la présentation.
Le barême est indicatif.

1 Questions de cours (3 points)

1. (2 points) Qu'est-ce qu'un paquetage java ? A quoi sert un paquetage en java ? Comment fait-on pour créer un paquetage en java ? Comment utilise-t-on le contenu d'un paquetage dans un programme en java ?
2. (1 points) Y-a-t'il une différence à l'exécution entre les deux extraits de programme suivants ? Si oui, laquelle, si non, justifiez.

```
Scanner s = new Scanner(System.in);
int monEntier;
try {
    System.out.println("Attention ...");
    monEntier = s.nextInt();
    System.out.println("J'ai lu !!!!!");
} catch (Exception e) {
    System.out.println("Oups ...");
}
```

```
Scanner s = new Scanner(System.in);
int monEntier;
System.out.println("Attention ...");
try {
    monEntier = s.nextInt();
} catch (Exception e) {
    System.out.println("Oups ...");
}
System.out.println("J'ai lu !!!!!");
```

2 Application graphique (10 points)

La classe `Timer` présente dans `javax.swing` sert à générer des événements de manière périodique. Une partie de ses méthodes se présente comme suit :

`Timer(int delay, ActionListener listener)`

prend en paramètre un delai entre deux évènements (en millisecondes) et une référence à un objet d'écoute.

`void start()`

démarre le `Timer` associé. Il enverra alors régulièrement un `ActionEvent` à son objet d'écoute.

`void stop()`

stoppe le `Timer` associé. Il arrête alors d'envoyer des évènements.

Question : En utilisant la classe `Timer`, écrivez une application qui :

— ouvre deux fenêtres de dimensions distinctes et remplit leur contenu de blanc

— dessine un rectangle noir à une position aléatoire de l'une des fenêtres. Les dimension de ce rectangle devront être égales à $\frac{1}{10}$ de celles de la fenêtre
— change le rectangle de fenêtre (en le plaçant à une nouvelle position aléatoire) à chaque seconde. Changer le rectangle de fenêtre signifie que le rectangle disparaît de la fenêtre dans laquelle il se trouve et apparaît dans l'autre
— quitte l'application si l'utilisateur clique à l'intérieur du rectangle

3 Threads (7 points)

Attention, dans toutes vos réponses, les threads doivent s'exécuter en concurrence.

1. (2 points) écrire un programme qui crée et lance l'exécution de N threads où N sera donné en argument de la ligne de commande du programme. Il faudra attribuer à chaque thread un numéro distinct (entre 0 et $N-1$) et chacun des threads devra afficher son numéro avant de se terminer. Une fois l'exécution de tous les threads terminée, le programme devra afficher un message et se terminer également.
2. (1 point) la classe `Thread` dispose d'une méthode statique `sleep` permettant de faire dormir le thread qui l'appelle pendant le nombre de millisecondes passé en argument. En utilisant cette méthode, modifiez votre programme pour que les threads affichent leur message par ordre de numéro croissant.
3. (3 points) l'utilisation de `sleep` pour imposer un ordre entre threads est à la fois innefficace et peu sure, il vaut mieux utiliser une vraie synchronisation. En utilisant les méthodes `wait` et `notifyAll`, modifiez votre programme initial (celui de la question 1) pour que les threads affichent leur message par ordre de numéro croissant.
4. (1 point) si l'ordre dans la question précédente avait été décroissant, cela aurait-il fait une différence (justifiez) ?

Choses utiles

```
Interface MouseListener (dans java.awt.event.*) :
    void mouseClicked(MouseEvent e)
    void mousePressed(MouseEvent e)
    void mouseReleased(MouseEvent e)
    void mouseEntered(MouseEvent e)
    void mouseExited(MouseEvent e)

Interface ActionListener (dans java.awt.event.*) :
    void actionPerformed(ActionEvent e)

Interface Runnable :
    void run()

Classe MouseEvent (dans java.awt.event.*) :
    int getX()
    int getY()

Classe JFrame (dans javax.swing.*) :
    static int EXIT_ON_CLOSE
    JFrame(String title)
    void add(Component comp, Object constraints)
    void setSize(int width, int height)
    void setVisible(boolean b)
    void setDefaultCloseOperation(int operation)

Classe JComponent (dans javax.swing.*) :
```

```
    void paintComponent(Graphics g)
    void addMouseListener(MouseListener l)
    void addActionListener(ActionListener l)
    Dimension getSize()
    void repaint()

Classe Graphics2D (dans java.awt.*) :
    void setPaint(Paint paint)
    void fillRect(int x, int y, int width, int height)

Classe Random (dans java.util.*) :
    int nextInt(int n)

Classe SwingUtilities (dans javax.swing.*) :
    static void invokeLater(Runnable doRun)

Classe Point (dans java.awt.*) : attributs x et y
    Point(int x, int y)

Classe Dimension (dans java.awt.*) : attributs width et height
    Dimension(int width, int height)

Classe implémentant Paint, Color (dans java.awt.*) : attributs black et white
```