UFR IM²AG

UNIVERSITÉ Grenoble Alpes

INF362 - Examen
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 Threads

1.1 Questions de cours (4 points)

1. (2 points) Expliquez comment, en l'absence de synchronisation, un programme multi-threadé peut aboutir à une incohérence. Donnez un exemple d'une telle situation.
2. (1 point) Quel mot clé propose le langage java pour résoudre ce problème de cohérence mémoire ? Quel est l'effet de ce mot clé ?
3. (1 point) Pourquoi a-t-on besoin des opérateurs `wait` et `notify` ? Donnez un exemple de situation dans laquelle un tel besoin apparaît.

1.2 Chaîne aléatoire (6 points)

Ecrivez un application multithreadée comportant N threads numérotés de 1 à N (N donné en argument de la ligne de commande), dans laquelle :

— initialement, l'un des N threads (déterminé au hasard) prend la main, c'est-à-dire qu'il démarre son exécution tandis que les autres attendent
— le thread qui a la main affiche un message indiquant son numéro et le fait qu'il a la main. Il tire ensuite un entier i compris entre 1 et N et passe la main au thread i après avoir affiché un message indiquant à qui il passe la main. Autrement dit, il se débrouille pour que le thread i reprenne son exécution et se met lui-même en attente
— les threads se passent ainsi la main jusqu'à ce que tous les threads aient eu la main. L'application devra alors se terminer

Attention, dans votre réponse, les threads doivent s'exécuter en concurrence et attendre grâce aux mécanismes de synchronisation fournis par java.

2 Une petite application graphique (10 points)

En utilisant Swing, écrivez une application qui crée une fenêtre graphique permettant de dessiner des rectangles avec gestion de l'historique :

— Un partie de la fenêtre (que nous appellerons zone de dessin), devra permettre à l'utilisateur de dessiner un rectangle en effectuant deux clics (dans deux coins diagonalement opposés du rectangle). Le dessin d'un nouveau rectangle n'efface pas les précédents
— La fenêtre devra comporter deux boutons :
  — Annuler : permettant, le cas échéant, d'effacer le rectangle le plus récent visible à l'écran
  — Refaire : permettant, le cas échéant, permettant de redessiner le rectangle effacé le plus récemment à l'aide du bouton annuler

Le nombre d'annulations de rectangle ne doit pas être borné : si l'utilisateur clique de manière répétée sur le bouton annuler, il doit pouvoir revenir jusqu'à la zone de dessin vierge initiale. De même, le nombre de restaurations ne doit pas non plus être borné : s'il clique de manière répétée sur le bouton refaire, l'utilisateur doit pouvoir restaurer tous les rectangles annulés. Les boutons annuler et refaire devront, idéalement, être grisés losque l'opération n'est pas possible. Enfin, lors du dessin d'un nouveau rectangle, toutes les annulations effectuées jusqu'alors ne pourront plus être restaurées à l'aire du bouton "Refaire".

Si cela vous facilite la tâche, vous pouvez partir du squelette de programme ci-dessous en indiquant ce que vous ajoutez aux points A et B. Bien entendu cela ne vous empêche en aucune manière d'écrire vos propre classes en complément, si besoin est.

```
import java.awt.*;
import javax.swing.*;

public class TestFenetre implements Runnable {
/* BEGIN SOLUTION */
/* ALTERNATIVE SOLUTION */

    // Point A

/* END SOLUTION */
    public void run() {
        // Creation d'une fenetre
        JFrame frame = new JFrame("Super examen de PROG6");

/* BEGIN SOLUTION */
        // Creation du composant de dessin, du bouton et ajout de l'objet de
        // traitement des evenements provenant du bouton
        AireDeDessin mon_dessin = new AireDeDessin();
        mon_dessin.addMouseListener(new EcouteurDeSouris(mon_dessin));
        JPanel panel = new JPanel();
        JButton bouton = new JButton("Annuler");
        bouton.addActionListener(new EcouteurDeBouton(mon_dessin, 0));
    panel.add(bouton);
        bouton = new JButton("Refaire");
        bouton.addActionListener(new EcouteurDeBouton(mon_dessin, 1));
    panel.add(bouton);

        // Ajout de nos composants dans la fenetre
        frame.add(mon_dessin, BorderLayout.CENTER);
        frame.add(panel, BorderLayout.SOUTH);
/* ALTERNATIVE SOLUTION */

    // Point B
/* END SOLUTION */

        // Un clic sur le bouton de fermeture clos l'application
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // On fixe la taille et on demarre
```

```
        frame.setSize(500, 500);
        frame.setVisible(true);
    }

    public static void main(String [] args) {
        SwingUtilities.invokeLater(new TestFenetre());
    }
}
```

Choses utiles

```
Classe Thread (dans java.lang.*) :
    Thread(Runnable comportement);
    void start();

Interface Runnable :
    void run();

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

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)

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 JButton (dans javax.swing.*) :
    JButton(String text)
    void addActionListener(ActionListener l)

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

```
    void add(Component comp);

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

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

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

Classe Rectangle (dans java.awt.*) :
    attributs x, y, width et height

Classe BorderLayout (dans java.awt.*) :
    attributs statiques NORTH, SOUTH, EAST, WEST

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