C. Recanati
JGRA
AIR2

Feuille 8

Exercice 1 (`AnimatorApplicationTimer.java`)
On va créer une application utilisant un `timer`. Conformément au squelette d'animation vu en cours, la classe de l'application étendra `JFrame` et implémentera `ActionListener`. L'animation consistera simplement à afficher un texte dans un `JLabel` indiquant le nombre d'animations (appels d'actions du `timer`) ayant eu lieu. Le nombre d'animations devant se produire par seconde (aps) sera passé en argument au programme java et utilisé pour initialiser le `timer`. Le constructeur du `JFrame` de l'application prendra en argument ce nombre et une chaîne de titre pour la fenêtre principale.

On spécifiera en outre un écouteur pour les événements sur la fenêtre (utilisez un `WindowAdapter`) qui permettra d'arrêter l'animation quand la fenêtre sera iconifiée, et de la relancer quand la fenêtre réapparaîtra à l'écran. On quittera l'application si l'utilisateur ferme la fenêtre. En outre, l'utilisateur pourra arrêter ou relancer l'animation en cliquant sur le label (utilisez un `MouseAdapter`).

Pour gérer les arrêts et reprises de l'animation, on utilisera un booléen (indiquant si l'animation est gelée ou non) et deux procédures `stopAnimation()` et `startAnimation`().

Exercice 2 (Un peu compliqué car les méthodes de Date sont obsolètes).
Modifier le programme précédent pour en faire une horloge digitale indiquant l'heure dans un label qui se réaffichera toutes les secondes.

Exercice 3 (`MovingImageTimer.java`)
On va créer une application (ou une applet) permettant de déplacer une image d'avant-plan représentant une fusée (à récupérer dans "rocketship.gif") sur une image de fond représentant un ciel étoilé ("starfield.gif"). Le principe du programme est le même que celui de l'exercice précédent : on utilise un timer, avec arrêt et reprise de l'animation sur un clic de souris ou sur l'iconification de la fenêtre.
On simplifie le programme précédent en fixant à 10 le nombre d'animations par seconde.
1) Version applet. La classe de l'application étend `JApplet` et implémente `ActionListener`. On déclarera le nombre d'animations,

un booléen pour savoir si l'animation est gelée, le timer et un panneau d'animation. Plus éventuellement des variables pour le nom des fichiers.

1a) La procédure `init`() chargera les images d'arrière plan et d'avant-plan (`bgImage` et `fgImage`) avec `getImage` :

```
Image bgImage =
    getImage(getCodeBase(),"images/file.gif"));
```

puis appellera une procédure `buildUI(Container container, Image bgImage, Image fgImage)` pour construire l'interface utilisateur. La procédure `buildUI` initialisera le timer et construira le panneau d'animation (une extension de `JPanel` ayant pour membres les deux images et une méthode `paintComponent` permettant d'afficher les deux images. On affichera d'abord l'image de fond au centre du panneau, puis l'image d'avant-plan en variant l'abscisse d'affichage selon l'indice du nombre d'animations courant).
Le panneau créé sera ajouté au conteneur et gèrera les clics de souris.

1b) Les procédures `start`() et `stop`() de l'applet appelleront comme dans l'exemple du poly des procédures `startAnimation`() et `stopAnimation`() qui gèrent le `timer`.

2) Version application (ou version mixte). On garde l'architecture précédente (la classe d'application `MyAppletClass` étend `JApplet`) mais on rajoute un `main` qui initialise les images `bgImage` et `fgImage` avec `Image bgImage =`

```
    Toolkit.getDefaultToolkit().getImage("images/file.gif");
```

On crée un `JFrame` et on initialise son panneau de contenu en utilisant la méthode `buildUI`. On a donc quelque chose comme :

```
    JFrame f = new JFrame("MovingImageTimer");
    final MyAppletClass controller = new MyAppletClass();
    controller.buildUI(f.getContentPane(), bgImage, fgImage);
```

On enregistre ensuite un écouteur de fenêtre sur le `JFrame`, on affiche le `JFrame`, puis on appelle `controller.startAnimation`().
NB : le fichier peut maintenant être utilisé soit pour une application, soit pour une applet.

Eléments de correction :

```
//  exercice 1 : AnimatorApplicationTimer.java
//  les import nécessaires...
/*
 * Un moule pour des applications d'animation.
 */
public class AnimatorApplicationTimer extends JFrame implements
                                                    ActionListener {
    int AnimationNumber = -1;
    Timer timer;
```

```
boolean frozen = false;
JLabel label;

 AnimatorApplicationTimer(int aps, String windowTitle) {
        super(windowTitle);
        int delay = (aps > 0) ? (1000 / aps) : 100;

        //Initialise un timer qui appelle l'action handler de cet objet.
        timer = new Timer(delay, this);
        timer.setInitialDelay(0);
        timer.setCoalesce(true);

        addWindowListener(new WindowAdapter() {
                public void windowIconified(WindowEvent e) {
                        stopAnimation();
                }
                public void windowDeiconified(WindowEvent e) {
                        startAnimation();
                }
                public void windowClosing(WindowEvent e) {
                        System.exit(0);
                }
        });

        label = new JLabel("Animation     ", JLabel.CENTER);
        label.addMouseListener(new MouseAdapter() {
                public void mousePressed(MouseEvent e) {
                        if (frozen) {
                                frozen = false;
                                startAnimation();
                        } else {
                                frozen = true;
                                stopAnimation();
                        }
                }
        });
        getContentPane().add(label, BorderLayout.CENTER);
}

//Peut être invoqué par n'importe quel thread
// puisque timer est thread-safe
public void startAnimation() {
        if (frozen) {
```

```
                //Ne rien faire.  L'utilisateur demande qu'on
                //arrête de changer l'image.
        } else {
                //Commencer l'animation
                if (!timer.isRunning()) {
                        timer.start();
                }
        }
}

//Peut etre invoque par n'importe quel thread
public void stopAnimation() {
        //Arrete le thread d'animation.
        if (timer.isRunning()) {
                timer.stop();
        }
}

// l'action declenchee par le timer
public void actionPerformed(ActionEvent e) {
        //Incremente le nombre d'animations et l'affiche
        AnimationNumber++;
        label.setText("Animation " + AnimationNumber);
}

public static void main(String args[]) {
        AnimatorApplicationTimer animator = null;
        int aps = 10;

        //Récupère le nombre d'animation par seconde (aps)
        // passé sur la ligne de commande en argument
        if (args.length > 0) {
                try {
                        aps = Integer.parseInt(args[0]);
                } catch (Exception e) {}
        }
        animator = new AnimatorApplicationTimer(aps,
                                                "Animation avec Timer");
        animator.setBounds(50,100,400,200);
        animator.setVisible(true);

        //OK pour commencer l'animation ici puisque
        //startAnimation peut etre invoquee dans n'importe quel thread.
```

```
            animator.startAnimation();
        }
}

//  exercice 3 : MovingImageTimer.java
//  (version mixte Applet/Application)

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
/*
 * Bouge une image d'avant plan
 * devant une image de fond
 */
public class MovingImageTimer extends JApplet implements
                                                    ActionListener {
        int AnimationNumber = -1;
        boolean frozen = false;
        Timer timer;
        AnimationPane animationPane;

        static String fgFile = "images/rocketship.gif";
        static String bgFile = "images/starfield.gif";

        //Invoquée seulement quand on tourne en applet.
        public void init() {
         //Recupere les fichiers images dans le repertoire du fichier code.
            Image bgImage = getImage(getCodeBase(), bgFile);
            Image fgImage = getImage(getCodeBase(), fgFile);
            buildUI(getContentPane(), bgImage, fgImage);
        }

        void buildUI(Container container,
                            Image bgImage, Image fgImage) {
            int aps = 10;

            //Combien de millisecondes entre les animations
            int delay = (aps > 0) ? (1000 / aps) : 100;

            //Configure un timer qui appelle l'action handler de cet objet.
            timer = new Timer(delay, this);
            timer.setInitialDelay(0);
            timer.setCoalesce(true);
```

```
        animationPane = new AnimationPane(bgImage, fgImage);
        container.add(animationPane, BorderLayout.CENTER);

        animationPane.addMouseListener(new MouseAdapter() {
                public void mousePressed(MouseEvent e) {
                        if (frozen) {
                                frozen = false;
                                startAnimation();
                        } else {
                                frozen = true;
                                stopAnimation();
                        }
                }
        });
}

//Invoquée seulement avec un navigateur
public void start() {
        startAnimation();
}

//Invoquée seulement avec un navigateur
public void stop() {
        stopAnimation();
}

//Peut etre invoquée par n'importe quel thread
public synchronized void startAnimation() {
        if (frozen) {
                // Ne rien faire. L'utilisateur a demandé
                // d'arrêter de changer l'image.
        } else {
                // lancer l'animation
                if (!timer.isRunning()) {
                        timer.start();
                }
        }
}

// Peut être invoquée par n'importe quel thread
public synchronized void stopAnimation() {
        // Stoppe le thread d'animation.
```

```
        if (timer.isRunning()) {
                timer.stop();
        }
}

public void actionPerformed(ActionEvent e) {
        // Incrémente le nombre d'animations
        AnimationNumber++;

         // Réaffiche le pannneau d'animation
        animationPane.repaint();
}

class AnimationPane extends JPanel {
        Image background, foreground;

        public AnimationPane(Image background,
                                        Image foreground) {
                this.background = background;
                this.foreground = foreground;
        }

        // Peint le cadre d'animation courant.
        public void paintComponent(Graphics g) {
                super.paintComponent(g);

                int compWidth = getWidth();
                int compHeight = getHeight();
                int imageWidth, imageHeight;

                // Si on a des largeur et hauteur valides pour
                // l'image de fond, la dessiner au centre.
                imageWidth = background.getWidth(this);
                imageHeight = background.getHeight(this);
                if ((imageWidth > 0) && (imageHeight > 0)) {
                        g.drawImage(background,
                                        (compWidth - imageWidth)/2,
                                        (compHeight - imageHeight)/2, this);
                }

                // Si on a des largeur et hauteur valides pour
                // l'image d'avant-plan, la dessiner
                imageWidth = foreground.getWidth(this);
```

```
                imageHeight = foreground.getHeight(this);
                if ((imageWidth > 0) && (imageHeight > 0)) {
                        g.drawImage(foreground,
                                        ((AnimationNumber*5)
                                                % (imageWidth + compWidth))
                                                 - imageWidth,
                                        (compHeight - imageHeight)/2,
                                        this);
                }
        }
    }

    // Invoquée seulement si on tourne comme application
    public static void main(String[] args) {
        Image bgImage = Toolkit.getDefaultToolkit().getImage(
                                        MovingImageTimer.bgFile);
        Image fgImage = Toolkit.getDefaultToolkit().getImage(
                                        MovingImageTimer.fgFile);

        JFrame f = new JFrame("Timer déplaçant une image");
        final MovingImageTimer controller = new
                                                MovingImageTimer();
        controller.buildUI(f.getContentPane(), bgImage, fgImage);

        f.addWindowListener(new WindowAdapter() {
                public void windowIconified(WindowEvent e) {
                                        controller.stopAnimation();
                }
                public void windowDeiconified(WindowEvent e) {
                        controller.startAnimation();
                }
                public void windowClosing(WindowEvent e) {
                        System.exit(0);
                }
        });
        f.setSize(new Dimension(500, 125));
        f.setVisible(true);
        controller.startAnimation();
    }
}
```