UFR IM²AG

UNIVERSITÉ Grenoble Alpes

GBIN6U03 - Examen
Année 2016-2017

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

1 *Design Patterns* (3 points)

1. Expliquez le fonctionnement ainsi que l'intérêt du *design pattern* nommé *observateur*. Donnez des cas d'utilisation de ce *pattern* ;
2. Donnez une implémentation possible pour l'ensemble des classes constituant le *pattern observateur*.

2 Quatre cent vingt et un (9 points)

Le 421 est un jeu de dés dans lequel un ensemble de joueurs jouent chacun leur tour à tirer 3 dés à 6 faces. Lors du tour d'un joueur, si les trois chiffres obtenus lors du tirage des trois dés sont 4, 2 et 1 (dans n'importe quel ordre), celui-ci gagne la partie. Dans le jeu original, un joueur peut retirer une partie des dés lors de son tour, nous n'en tiendrons pas compte ici. Dans les questions qui suivent, il vous est demandé d'écrire différentes versions successives d'une simulation du jeu. Pour chaque question indiquez clairement dans votre réponse les parties de votre programme qui ne changent pas par rapport à la question précédente, celles qui changent et celles qui sont nouvelles.

Attention : dans toute la suite, les programmes multi-threadés qui vous sont demandés doivent impérativement exécuter les différents threads de manière concurrente (et non les uns après les autres). De plus, dans les questions qui vous demandent de synchroniser les threads entre eux, une attente active sera pénalisée par rapport à une synchronisation correcte utilisant les primitives de synchronisation fournies par java.

Questions :

1. (5 points) dans un premier temps, nous supposerons que les joueurs disposent chacun de leurs propres dés et jouent tous en même temps sans tenir compte de leur tour. Implémentez un programme de simulation d'une partie de 421 à 3 joueurs dans lequel chaque joueur est simulé par un thread distinct. Chacun de ces threads se bornera à tirer ses 3 dés en boucle jusqu'à obtenir 421 ou bien jusqu'à ce qu'un autre thread obtienne 421, puis affichera un message indiquant s'il a obtenu 421 ou non. Le code exécuté par les threads ne devra pas contenir de synchronisations, sauf éventuellement le programme principal qui devra afficher un message à la fin de la simulation.
2. (2 points) en suivant la spécification de la question précédente, il est possible que deux joueurs gagnent en même temps, expliquez comment. Modifiez votre programme pour faire en sorte qu'il n'y ait toujours qu'un seul gagnant.
3. (2 points) modifiez votre programme pour que les trois joueurs ne jouent que chacun leur tour de manière cyclique.

3 Ligne brisée (8 points)

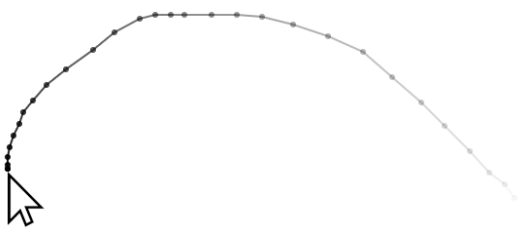
Dans cette partie, vous devez écrire les parties manquantes du programme suivant en vous aidant des fonctions données en annexes :

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.stage.Stage;
import javafx.scene.layout.BorderPane;

public class ProgrammePrincipal extends Application {
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Examen, poil au dessin");
        Canvas c = new Canvas();
        MonApplication app = new MonApplication(c);
        BorderPane b = new BorderPane(c);
        c.widthProperty().bind(b.widthProperty());
        c.heightProperty().bind(b.heightProperty());
        Scene s;
        s = new Scene(b, 800, 600);
        primaryStage.setScene(s);
        primaryStage.show();
        app.start();
    }

    public static void main(String [] args) {
        launch(args);
    }
}
```

Le but est d'obtenir un programme ouvrant une fenêtre et permettant de dessiner une ligne brisée en déplaçant la souris dans cette fenêtre tout en appuyant sur le bouton. Cette ligne brisée doit ressembler à la ligne ci-dessous :



Cette ligne brisée a les caractéristiques suivantes :

- elle est constituée de segments reliant les points par lesquels la souris est passée. Chacun de ces points est fourni par JavaFX lors de la remontée d'un évènement de déplacement de souris en maintenant de bouton pressé (`MouseDragged`) ;
- chaque point de passage est marqué par un cercle de rayon 2 ;
- elle est constituée de 30 segments maximum. Si la souris continue de se déplacer alors que la ligne comporte déjà 30 segments, les plus anciens sont supprimés ;
- l'opacité des segments décroit en fonction de leur ancienneté : de 1 (opacité maximale) pour le plus récent à 0 (opacité minimale) pour le plus ancien. Cette opacité correspond au canal alpha utilisé lors des tracés dans un `GraphicsContext` ;
- lorsque l'utilisateur ne bouge plus la souris ou lorsque le bouton est relaché, la ligne disparait avec le temps : elle doit perdre 1% d'opacité à chaque *pulse* produit par JavaFX ;
- l'opacité repasse au maximum lorsque le bouton de la souris est pressé ou lorsque celle ci est déplacée en maintenant le bouton enfoncé ;
- lorsque l'utilisateur presse le bouton de la souris une nouvelle ligne démarre et toute ligne précédente encore à l'écran est effacée.

Annexes

```
Classe Canvas (dans javafx.scene.canvas) :
    public GraphicsContext getGraphicsContext2D();
    public final double getWidth();
    public final double getHeight();
méthodes héritées de Node :
    public final void setOnMousePressed(EventHandler<? super MouseEvent> value);
    public final void setOnMouseReleased(EventHandler<? super MouseEvent> value);
    public final void setOnMouseDragged(EventHandler<? super MouseEvent> value);

Classe AnimationTimer (dans javafx.animation) :
    public void handle(long time);
    public void start();

Classe EventHandler<T> (dans javafx.event) :
    public void handle(T event);

Classe MouseEvent (dans javafx.scene.input) :
    public final double getX();
    public final double getY();

Classe GraphicsContext (dans javafx.scene.canvas) :
    public void clearRect(double x, double y, double w, double h);
    public void setGlobalAlpha(double alpha);
    public void strokeLine(double x1, double y1, double x2, double y2);
    public void strokeOval(double x, double y, double w, double h);

Interface Runnable :
    public void run()

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