Projet de technologie de l'information
Examen final

28 mai 2013

Indications :

- l'examen dure de 10h15 à 12h00,
- aucun document ni appareil électronique n'est autorisé,
- indiquez votre nom, prénom et numéro SCIPER ci-dessous et sur toutes les éventuelles feuilles additionnelles que vous rendriez,
- placez votre carte d'étudiant sur la table.

Bon travail !

Nom : _____

Prénom : _____

SCIPER : _____

1 Première moitié

Les exercices de cette première moitié concernent la première partie du projet, depuis son début et jusqu'au rendu intermédiaire.

1.1 Accumulateur et systèmes de coordonnées [11 points]

La figure 1 montre le système de coordonnées du plan formé de l'origine O_1, de l'axe des abscisses x_1 et de l'axe des ordonnées y_1, ainsi qu'un cadre de dessin pour une fractale. De plus, quatre points générés par l'algorithme du chaos, P_1 à P_4, sont représentés.

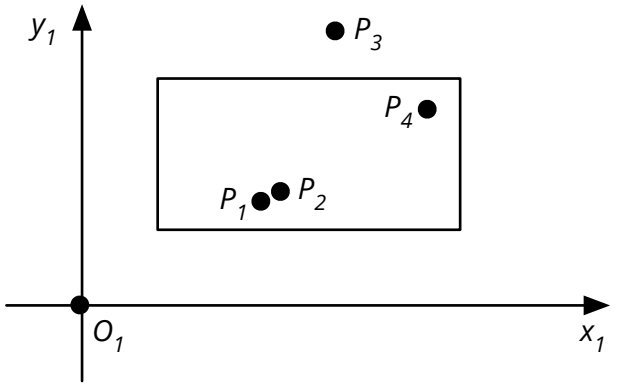

FIGURE 1 – Le plan, un cadre de dessin et des points

Partie 1 [4 points] Lors du calcul et du dessin d'une fractale dans un fichier image, un total de trois systèmes de coordonnées sont utilisés. Dans l'ordre de leur utilisation, ce sont :

1. le système de coordonnées du plan,
2. le système de coordonnées de l'accumulateur,
3. le système de coordonnées de l'image.

Représentez l'origine et les deux axes de chacun des deux derniers systèmes de coordonnées sur la figure 1, en prenant garde à ne pas oublier les pointes de flèches au bout des axes. Nommez ces éléments O_2, x_2 et y_2 pour le système de l'accumulateur et O_3, x_3 et y_3 pour celui de l'image.

Faites bien attention à orienter le système de coordonnées de l'accumulateur comme dans l'énoncé, c-à-d de manière identique au système du plan ! Quant au système de coordonnées de l'image, rappelez-vous que le premier pixel écrit dans le fichier a la coordonnée (0,0), le second la coordonnée (1,0), et ainsi de suite.

Partie 2 [2 points] Admettons que l'on utilise un accumulateur de quatre cases de large et deux de haut pour dessiner une fractale dans le cadre ci-dessus. Représentez sur la figure 1, et de manière assez précise, les huit cases de cet accumulateur, en numérotant les lignes et les colonnes (à partir de 0).

Partie 3 [5 points] Admettons que l'on désire produire une image en niveaux de gris à partir de l'accumulateur décrit ci-dessus et des quatre points P_1 à P_4 produits par l'algorithme du chaos (après les 20 itérations d'initialisation). On utilise pour cela la technique vue à l'étape 3.

- Combien de pixels comportera cette image ? _____
- Combien d'entre-eux seront noirs (intensité 0) ? _____
- Combien d'entre-eux seront blancs (intensité 1) ? _____
- Le (ou les) pixel(s) ni noirs ni blancs auront-ils une intensité inférieure, égale ou supérieure à $\frac{1}{2}$, et pourquoi ? _____

1.2 Enumérations [3 points]

La version actuelle du projet n'utilise pas d'énumérations Java car vous ne connaissiez pas leur fonctionnement jusqu'à récemment. Nommez la classe du projet, écrite avant le rendu intermédiaire, qui pourrait être transformée en une énumération, en expliquant brièvement les éventuels bénéfices que cela apporterait.

Réponse : _____

1.3 Transformations [7 points]

Dans la version actuelle du projet, les transformations sont représentées par l'interface `Transformation`, qui est implémentée par les classes `Variation`, `AffineTransformation` et `FlameTransformation`. Au moyen de la méthode `composeWith` de la classe `AffineTransformation`, il est possible de composer deux transformations affines entre-elles.

Une autre manière d'organiser ces classes serait de faire une classe abstraite de l'interface `Transformation` et d'y placer la méthode de composition. On obtiendrait alors la définition suivante :

```
public abstract class Transformation {
  abstract public Point transformPoint(Point p);

  public Transformation composeWith(Transformation that){
    return new ComposedTransformation(this, that);
  }
}

class ComposedTransformation extends Transformation {
  private final Transformation t1, t2;

  public ComposedTransformation(Transformation t1,
                                Transformation t2) {
    this.t1 = t1;
    this.t2 = t2;
  }

  @Override
  public Point transformPoint(Point p) { à faire }
}
```

Cette version, qui utilise le patron *Composite*, a l'avantage de permettre la composition de deux transformations quelconques, et pas uniquement de deux transformations affines.

Partie 1 [4 points] Ecrivez le code de la méthode `transformPoint` de la classe `ComposedTransformation`. Notez que comme les deux transformations à composer (`t1` et `t2`) ont le type `Transformation`, vous ne pouvez pas utiliser de multiplication matricielle pour les composer ! Ne cherchez pas une solution compliquée, la réponse tient en une (courte) ligne.
Réponse :

Partie 2 [3 points] Une fois la définition ci-dessus en place, il serait possible de supprimer la méthode `composeWith` de `AffineTransformation` et d'utiliser celle de la classe `Transformation` à sa place. Pensez-vous que le programme résultant serait plus ou moins rapide que le programme actuel ? Justifiez votre réponse.
Réponse :

2 Seconde moitié

Les exercices de cette seconde moitié concernent la deuxième partie du projet, depuis le rendu intermédiaire jusqu'au rendu final.

2.1 Redessin inutile [7 points]

Partie 1 [4 points] Comme cela était mentionné dans l'énoncé de l'étape 9, lorsque le programme Flame Maker est terminé, il souffre du problème suivant : la fractale est redessinée chaque fois que l'utilisateur sélectionne une autre transformation Flame dans la liste de ces transformations, ce qui est clairement inutile.

Expliquez pourquoi le problème se produit en complétant la liste ci-dessous qui décrit la chaîne des observateurs avertis lorsque la sélection change. Notez qu'il ne vous est pas demandé de lister la totalité des observateurs avertis, mais seulement ceux qui causent le problème, c-à-d ceux qui mènent du premier au dernier point de cette liste.

1. L'observateur de la `JList` est averti du changement de la sélection et appelle la méthode `setSelectedTransformationIndex` de la classe `FlameMakerGUI` afin de mettre à jour l'index de la transformation sélectionnée.
2. _____
3. _____
4. L'observateur du `ObservableFlameBuilder` créé par le composant affichant la fractale est averti d'un changement et appelle la méthode `repaint` de ce composant, ce qui provoque son redessin, en l'occurrence inutile.

Partie 2 [3 points] Pour éviter le redessin inutile, il faut d'une manière ou d'une autre interrompre la chaîne de notification décrite par la liste ci-dessus lorsque la sélection change.

A quel point de cette chaîne êtes-vous intervenu dans votre projet pour l'interrompre (entre 1 et 4) ? _____

Brièvement, comment fonctionne votre solution ? _____

2.2 Dessin de la grille [5 points]

Le composant qui affiche les parties affines des transformations Flame est doté d'une grille. Comme vous le savez, les lignes de cette grille sont composées des points dont au moins une des coordonnées — dans le système de coordonnées du plan — est entière.

L'extrait de programme ci-dessous dessine une partie des lignes de cette grille. La variable `g` contient le contexte graphique, `frame` contient le cadre — dans le système de coordonnées du plan — pour lequel on désire dessiner la grille et `planeToComponent` est une transformation permettant de passer des coordonnées du plan à celles du composant.

```
Graphics2D g = /* ... */;
Rectangle frame = /* ... */;
Transformation planeToComponent = /* ... */;

double xLeft = Math.ceil(frame.left());
double xRight = Math.floor(frame.right());
for (double x = xLeft; x <= xRight; x = x + 1.0) {
  Point yMin = à faire;
  Point yMax = à faire;
  g.draw(new Line2D.Double(yMin.x(), yMin.y(),
                           yMax.x(), yMax.y()));
}
```

Pour mémoire, la méthode `Math.floor` retourne le plus grand entier inférieur ou égal à son argument, tandis que `Math.ceil` retourne le plus petit entier supérieur ou égal à son argument. Quant à la classe `Rectangle`, souvenez-vous qu'elle possède, en plus de `left` et `right` utilisées ci-dessus, des méthodes permettant d'obtenir la largeur (`width`), la hauteur (`height`), la plus petite (`bottom`) et la plus grande (`top`) coordonnée y et le centre (`center`) du rectangle.

Partie 1 [1 point] Les lignes dessinées par le code ci-dessus sont-elles horizontales ou verticales ? _____

Partie 2 [4 points] Donnez le code permettant de calculer la valeur des points `yMin` et `yMax`.

_____

_____

_____

_____