Architecture Logicielle pour l'Interaction

durée indicative : **2h**
documents autorisés : **1 feuille A4 recto-verso manuscrite de votre main**

consignes :
Le barème est **indicatif**. La qualité de la **présentation**, de l'**expression**, de l'**orthographe** sera prise en compte dans la notation de **manière significative**.
Si le sujet présente des ambiguïtés, précisez vos choix. Il sera tenu compte de vos hypothèses. Lorsqu'il est demandé de produire du code, ne vous focalisez pas sur la correction de la syntaxe.

Généralités (6 points)

Question 1

a) Comment distinguer ce qui fait partie du noyau fonctionnel de ce qui fait partie de l'interface d'une application ?
b) Expliquez pourquoi il faut séparer le code du noyau fonctionnel de celui de l'interface.
c) Expliquez pourquoi le mécanisme des *callbacks* (fonctions de rappel) est important dans cette séparation.
d) Selon Fekete [1996], le noyau fonctionnel doit fournir les trois services suivants : notification, prévention des erreurs, et annulation. Expliquez pourquoi en donnant un exemple d'interaction impossible ou difficile à réaliser pour chacun de ces services lorsqu'il est absent.

Programmation par événements (8 points)

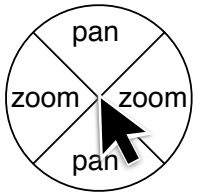
Question 2

Vous allez spécifier et réaliser le comportement d'une interaction, le ***control-menu***, qui permet de (dé)zoomer et de déplacer un graphique. Celui-ci fonctionne de la manière suivante :

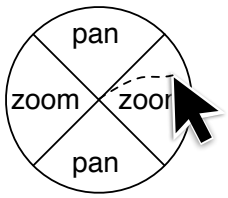
- lorsque l'utilisateur enfonce le bouton de la souris (évènement noté ↓), un menu circulaire divisé en quatre secteurs apparait sous le curseur de la souris ;
- lorsque la souris bouge, avec le bouton toujours enfoncé, (évènement noté M), il ne se passe rien tant que le curseur de la souris ne sort pas du cercle ;
- lorsque le curseur sort du cercle, le menu disparaît et une interaction contrôlée par les déplacements du curseur commence : un (dé)zoom si il sort par les côtés droit ou gauche ou un déplacement si il sort par les côtés haut ou bas.
- l'interaction se prolonge jusqu'au relâché du bouton de la souris (évènement noté ↑).

Soit, en images :

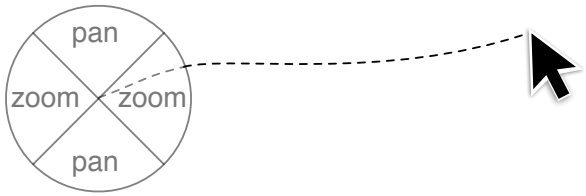
- le menu au départ



- le menu pendant la sélection de l'interaction



- le menu qui a disparu et l'interaction de zoom contrôlée par le déplacement du curseur



Le déplacement est géré par les fonctions suivantes, fournies :
- `void begin_pan(Point start)` avec `start` le point initial (centre du menu) au début du déplacement ;
- `void pan(int dx, int dy)` avec `dx` et `dy` les déplacements horizontaux et verticaux du curseur au cours du déplacement ; et
- `void end_pan()` pour la fin du déplacement.

Le zoom est géré par les fonctions suivantes (qui prennent les mêmes paramètres) fournies :
- `void begin_zoom(Point start)` au début du déplacement ;
- `void zoom(int dx, int dy)` au cours du déplacement ; et
- `void end_zoom()` pour la fin du déplacement.

Le rayon du cercle est de 40 pixels, pour déterminer par quel côté le curseur sort, on peut remarquer qu'il sort par la droite ou la gauche si et seulement si son déplacement depuis le centre du menu est plus important horizontalement que verticalement (en valeurs absolues).

a) **Dessinez la machine à états qui spécifie cette interaction.**

On rappelle que les évènements ont une méthode `getPoint()` qui retourne un `Point` avec la position du curseur. Ce point à des méthodes `getX()` et `getY()` qui en donnent les coordonnées, et une méthode `distance(Point p)` qui donne sa distance à un autre point. On dispose également de la fonction `abs(int i)` qui retourne la valeur absolue d'un entier. Il n'est pas demandé de gérer le feedback (affichage et masquage du menu).

b) Réalisez en java cette machine à états.

Pour cela, vous est donnée une classe `Canvas` (page suivante) qui utilisera votre classe `Interaction`.
C'est cette dernière que vous devez réaliser.

```
package exam.ui;

import javax.swing.JComponent;
import java.awt.Point;

public class Canvas extends JComponent {
	Interaction interaction;
	public Canvas() {
		interaction = new Interaction(this);
		addMouseListener(interaction);
		addMouseMotionListener(interaction);
	}

	public void begin_pan(Point start) { ... }
	public void pan(int dx, int dy) { ... }
	public void end_pan() { ... }

	public void begin_zoom(Point start) { ... }
	public void zoom(int dx, int dy) { ... }
	public void end_zoom() { ... }
}
```

Les interfaces fournies par `java.awt.event` permettant de recevoir les événements sont données ci-dessous, elles disposent toutes d'un adaptateur associé (`XxxxAdapter` pour `XxxxListener`).

Pour écouter la **souris** :

```
public interface MouseListener {
	public void mousePressed(MouseEvent e);
	public void mouseReleased(MouseEvent e);
	...
}

public interface MouseMotionListener {
	public void mouseDragged(MouseEvent e);
	...
}
```

L'interface `MouseInputListener` qui réunit ces deux interfaces est également disponible.

Les `MouseEvent` ont les méthodes utiles suivantes :
- `Point getPoint()` qui retourne le lieu de l'événement ;
- `int getX()` et `int getY()` qui retournent la même information.

Votre classe `Interaction` pourra commencer ainsi :

```
package exam.ui;

public class Interaction ... {
	protected Canvas canvas;
	public Interaction(Canvas canvas) {
		this.canvas = canvas;
		...
	}
	...
}
```

Composants interactifs (6 points)

Question 3

Les figures ci-dessous donnent la fenêtre des téléchargements du navigateur Firefox sous Mac OS X avant et après un redimensionnement. Sachant que la largeur de la colonne de gauche peut être modifiée par l'utilisateur, donnez l'arbres des widgets que vous utiliseriez pour coder cette interface avec Java/SWING en spécifiant si nécessaire pour les conteneurs les gestionnaires de géométrie utilisés et la localisation de leurs fils.
Certains aspects ne peuvent être rendus simplement avec SWING : les boutons collés dans la barre de boutons pourront être considérés comme un seul bouton.
Ne détaillez pas le contenu de la liste des téléchargement.