Interaction Homme-Machine

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

Commencez par lire l'intégralité du sujet en détail.
Le barème est **indicatif**. La qualité de la **présentation**, de l'**expression**, ainsi que la **précision** et la **concision** seront prises en compte dans la notation de **manière significative**. Si le sujet présente des ambiguïtés, précisez vos choix. Lorsqu'il est demandé de produire du code, ne vous focalisez pas sur la correction de la syntaxe.

Généralités (7 points)

Question 1
Selon Fekete [1996], le noyau fonctionnel doit fournir les trois services suivants : **notification**, **prévention des erreurs**, et **annulation**.
Expliquez pourquoi en donnant pour chacun de ces services un exemple d'interaction impossible ou difficile à réaliser s'il est absent.

Question 2
Selon le modèle de Seeheim, le code de l'interface utilisateur peut se décomposer en trois parties : la **présentation**, le **contrôleur de dialogue** et l'**interface du noyau fonctionnel**, correspondant respectivement aux niveaux **lexical**, **syntaxique** et **sémantique** de l'interaction.
a) Dans un explorateur de fichiers, on utilise la souris pour glisser l'icone d'un fichier sur celle d'un répertoire pour y déplacer le fichier. Expliquez à quoi correspondent les trois niveaux ci-dessus pour l'interaction en entrée à la souris sur cet exemple.
b) Le(s)quel(s) de ce(s) niveau(x) sont fournis par les boîtes à outils de construction d'interface ?

Question 3
Vous faites partie d'une équipe chargée de développer une nouvelle interface pour un logiciel existant. Dans le code du noyau fonctionnel, vous découvrez le passage suivant :

```
package nf;

class UIUpdater {
   ...
   public void loop() {
      while(not ended) {
         // met à jour l'affichage de la valeur mesurée par le capteur
         ui.sensor_label.setText(sensor.getValue().toString());
         // attend 1/10 de seconde avant de remesurer
         Thread.sleep(100);
      }
   }
   ...
}
```

a) Quels sont tous les problèmes que vous identifiez dans ce code ?
b) Quelles modifications allez-vous suggérer à l'équipe chargée du noyau fonctionnel ?

Programmation par événements (7 points)

Question 4
Vous allez spécifier et réaliser le comportement d'une interaction qui permet de dessiner des segments dans une application de dessin.
Lorsqu'on enfonce le bouton de la souris (↓), on débute un segment à cet endroit.
Lorsqu'on déplace ensuite la souris (M), l'autre extrémité du segment suit ce déplacement, et quand on relâche le bouton (↑), le segment est terminé.
Si, alors qu'on bouge l'extrémité du segment, la touche *shift* est maintenue enfoncée, le déplacement est contraint pour que le segment soit horizontal, vertical ou incliné à ±45°.
La touche *shift* envoie les évènements P (*press*) quand elle est enfoncée et R (*release*) quand elle est relâchée.

Le segment est géré par une classe, fournie, qui dispose des méthodes suivantes :
- le constructeur `Segment(Point2D origin)` qui crée un segment au début du déplacement avec `origin` le point initial qui sera utilisé pour la première extrémité ;
- `void move(Point2D end)` qui déplace la seconde extrémité du segment, de manière non-contrainte ; et
- `void constrained_move(Point2D end)` qui déplace la seconde extrémité du segment de manière contrainte.

La touche *shift* peut être enfoncée ou relâchée à tout moment pendant qu'on dessine un segment.

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

On supposera que les évènements souris (↓, M et ↑) ont une méthode `getPoint()` qui retourne un `Point2D` avec la position du curseur et que les évènements clavier (P et R) ont une méthode `getCode()` qui retourne un `KeyCode` que l'ont peut comparer à la constante `KeyCode.SHIFT` pour savoir si c'est bien la touche *shift* qui a été actionnée.

b) **Réalisez en java cette machine à états**.

Pour cela, vous est donnée une classe `Surface` qui utilisera votre classe `Interaction`, et à laquelle on peut ajouter de segments grâce à la méthode `void addSegment(Segment s)`.
On donne également une classe `Segment` (page suivante).
C'est la classe `Interaction` que vous devez réaliser.

```
package exam.ui;

import javafx.scene.canvas.Canvas;

public class Surface extends Canvas {
      Interaction interaction;
      public Canvas() {
            interaction = new Interaction(this);
      }
      public void addSegment(Segment s) { ... }
}
```

```
package exam.ui;
```

```
import javafx.geometry.Point2D;

public class Segment {
      public Segment(Point2D origin) { ... }
      public void move(Point2D end) { ... }
      public void constrained_move(Point2D end) { ... }
}
```

Votre classe `Interaction` pourra commencer ainsi :

```
package exam.ui;

public class Interaction ... {
      protected Surface surface;
      public Interaction(Surface surface) {
            this.surface = surface;
            ...
      }
      ...
}
```

On rappelle qu'on peut s'abonner à une instance de `Canvas` via la méthode `addEventHandler` qui prend en premier paramètre le type d'évènement attendu, et en second paramètre une instance spécialisant l'interface fournie par `javafx.event` donnée ci-dessous :

```
public interface EventHandler<T> {
      public void handle(T e);
}
```

Les évènements qui nous intéressent sont ici de cinq types répartis dans deux classes :

- `MouseEvent` pour la souris pour les types `MouseEvent.MOUSE_PRESSED` (↓), `MouseEvent.MOUSE_RELEASED` (↑) et `MouseEvent.MOUSE_DRAGGED` (M).
- `KeyEvent` pour le clavier pour les types `KeyEvent.KEY_PRESSED` (P) et `KeyEvent.KEY_RELEASED` (R).

`InputEvent` est une superclasse de `MouseEvent` et `KeyEvent`. Pour simplifier, on supposera qu'elle dispose de la méthode `getPoint()` qui retourne un `Point2D` avec la position du curseur et de la méthode `getCode()` qui retourne un `KeyCode` (en réalité il faudrait transtyper les `InputEvent` en `MouseEvent` ou en `KeyEvent` suivant les cas pour accéder à ces méthodes).

Pour tout évènement `e`, on peut accéder à son type grâce à `e.getEventType().getName()` qui renvoie une `String` que l'on peut comparer à `"MOUSE_PRESSED"`, `"MOUSE_RELEASED"`, `"MOUSE_DRAGGED"`, `"KEY_PRESSED"`, `"KEY_RELEASED"`.

Composants interactifs (6 points)

Question 5

Les figures ci-dessous donnent la fenêtre de l'application VLC 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'arbre des widgets que vous utiliseriez pour coder cette interface avec JavaFX.

Certains aspects ne peuvent être rendus simplement avec JavaFX : les boutons collés dans la barre de boutons pourront être considérés comme n'étant pas collés.

Ne détaillez pas le contenu de la liste de lecture ni celui des catégories dans la colonne de gauche.