Architecture Logicielle pour l'IHM

durée : **1h30**
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 (5 points)

Question 1

Dans le projet sur lequel votre équipe de développement travaille apparaissent les codes suivant :

```
package fc;
import ui.View;

class Model {
   View view;
   public Model(View v) { view = v; }
   public void setValue(String value) { v.valueChange(value); }
   ...
}
```

```
package ui;
import fc.Model;

class View {
   JLabel label;
   public View() { label = new JLabel(); }
   public void valueChange(String value) { label.setText(value); }

   public static void main(String args[]) {
      view = new View();
      model = new Model(view);
   }
}
```

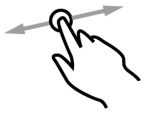
a) Citez les trois grands principes sur la séparation de l'interface utilisateur et indiquez le(s)quel(s) est(sont) respecté(s) ou non dans cet extrait de code.
b) Modifiez les deux classes (en ajoutant si nécessaire des éléments) pour que les trois principes soient respectés et que quand la méthode setValue du modèle `model` est appelée, le texte de l'étiquette `label` change bien pour refléter la nouvelle valeur.

Programmation par événements (9 points)

Question 2

L'interface graphique d'un logiciel qui permet de manipuler des photos sur une surface interactive (tablette ou smartphone) en utilisant les doigts comme pointeurs.

Ceux-ci génèrent trois types d'événements : ↓ lorsqu'un doigt entre en contact avec la surface ; D lorsqu'il se déplace au contact de la surface ; et enfin ↑ lorsqu'il quitte cette dernière.
Ces événements contiennent la position (`e.getPoint()`) où ils ont eu lieu ainsi qu'un identifiant du doigt utilisé (`e.getFingerID()`).
On donne ci-dessous le code qui réalise l'interaction suivante :



- avec 1 doigt, la photo est déplacée (déplacement géré par les fonctions données suivantes : `start_drag(Point start), drag(int dx, int dy), end_drag()`) ;

- avec 2 doigts, la photo est déplacée/tournée/zoomée (redimensionnement géré par la fonction donnée suivante : `pinch(Point old_p1, Point p1, Point old_p2, Point p2)` où old_p1 (resp. p1) désigne la position précédente (resp. courante) de l'un des doigt et old_p2 (resp. p2) celle de l'autre doigt).

a) Dessinez la machine à état qui est implémentée dans la classe `Interaction` du code ci-dessous pour réaliser ces interactions.

```
public class Surface extends JComponent {
   Interaction interaction;
   public Surface() { interaction = new Interaction(this); }

   public void start_drag(Point start) { ... }
   public void drag(int dx, int dy) { ... }
   public void end_drag() { ... }

   public void pinch(Point old_p1, Point p1, Point old_p2, Point p2) { ... }
}
```

```
public class Interaction {
   enum State { IDLE, DRAG, PINCH };
   State state = State.IDLE;

   protected Surface surface;
   int f1; int f2;
   Point p1; Point p2;

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

   void mousePressed(MouseEvent e) {
      switch(state) {
      case IDLE:
         f1 = e.getFingerID();
         p1 = e.getPoint();
         surface.start_drag(p1);
         state = State.DRAG;
         break;
      case DRAG:
         surface.end_drag();
         f2 = e.getFingerID();
         p2 = e.getPoint();
         state = State.PINCH;
         break
      default:
         break;
      }
   }
```

```
   void mouseDragged(MouseEvent e) {
      switch(state) {
      case DRAG:
         surface.drag(e.getX()-p1.getX(), e.getY()-p1.getY());
         p1 = e.getPoint();
         state = State.DRAG;
         break;
      case PINCH:
         if(e.getFingerID() == f1) {
            surface.pinch(p1, e.getPoint(), p2, p2);
            p1 = e.getPoint();
         elif(e.getFingerID() == f2) {
            surface.pinch(p1, p1, p2, e.getPoint());
            p2 = e.getPoint();
         }
         state = State.PINCH;
         break;
      default:
         break;
      }
   }

   void mouseRelease(MouseEvent e) {
      switch(state) {
      case DRAG:
         surface.end_drag();
         state = State.IDLE;
         break;
      case PINCH:
         state = State.IDLE;
         break;
      default:
         break;
      }
   }
}
```

b) La classe `Interaction` telle que donnée ci-dessus ne recevra pas d'évènements. Donnez les modifications qu'il faut faire à sa déclaration (première ligne) et à son constructeur (3ème ligne) pour que les évènements lui arrivent bien via les interfaces fournies par `java.awt.event` données ci-dessous :

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

c) Dans l'implémentation donnée, le relevé d'un doigt quand on est en mode "pinch" interrompt complètement l'interaction. Modifiez la machine à état pour que lorsqu'on relève un doigt, on reprenne un drag avec le doigt restant sur l'écran (qui peut être n'importe lequel des deux doigts identifiés par `f1` et `f2`).

Composants interactifs (6 points)

Question 3

Les figures ci-dessous donnent la fenêtre des contacts de l'application SF Symbol sous Mac OS X avant et après un redimensionnement. Sachant que la largeur de la colonne de gauche peut être modifiée par l'utilisateur, mais que celle de droite a une largeur fixe, donnez l'arbre des composants 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 enfants. Ne détaillez ni le contenu des catégories (à gauche), ni celui de la table (au milieu), ni celui du panneau des détails (à droite).