

1. Expliquez quelles sont les principales différences entre la conception d'interfaces Web et WIMP (« windows, icons, menus, pointing devices ») d'une perspective ergonomique. Que faut-il tenir compte en particulier pour la conception d'interfaces Web? (¾ page max; 3 pts)

Les différences principales se situent au plan des contextes d'utilisation qui sont passablement différents entre les WIMP et le Web. Les différences les plus marquées ont trait à la fréquence d'utilisation et les segments d'utilisateurs qui varient beaucoup selon les sections d'un site. Il faut aussi évaluer si la tâche nécessite un haut degré d'interactivité ce qui n'est pas aisé avec une interface basée sur un clavier. De plus, on peut aussi mentionner que la variabilité des plateformes est beaucoup plus grande sur le Web et que la navigation dans un site pose souvent des problèmes importants. Finalement, on note que tous les sites permettent l'enregistrement du trafic de visiteurs ce qui offre des données intéressantes pour la révision de la conception d'un site. S'il y a des différences entre les interfaces WIMP et Web, il faut aussi noter que ce sont néanmoins les mêmes principes de conception qui s'appliquent.

2. Référez-vous au texte suivant pour répondre à la question qui le suit.

Positionnement Global inc. est une entreprise qui produit des systèmes de navigation pour l'automobile. Ces systèmes permettent de localiser le véhicule précisément sur une carte routière grâce à une technologie GPS et une application de géomatique. L'appareil se situe à côté du conducteur et lui permet notamment de se situer sur une carte à échelle variable, de trouver le chemin le plus court ou le plus rapide entre deux points et de le guider tout au long du trajet. Il possède des fonctionnalités comme la mémorisation de plusieurs points de départ et d'arrivés (ex. maison, bureau, chalet), la localisation d'intersections ou d'adresses civiques, etc. Toute cette fonctionnalité est présentement accessible par l'interface de l'appareil par le truchement de menus et de boutons physiques.

Elle associe à une entreprise qui possède une des technologies de reconnaissance vocales les plus performantes, Sucol inc., pour développer une nouvelle version de l'appareil dont les fonctions seraient disponibles par commandes vocales.

En présumant que le projet de développement de la nouvelle version dure 8 mois, définissez les grandes étapes d'un processus de développement centré-utilisateur selon la norme ISO13407 et les méthodes prescrites par Maguire. Précisez les activités principales en justifiant pourquoi vous choisissez chaque activité plutôt qu'une autre. (1 page max; 3 pts.)

Il y a plusieurs réponses valables. On peut notamment évoquer le cycle ISO13407 (schéma de la figure 1 de Maguire) pour suggérer d'effectuer quelques itérations de solutions de conception (2 à trois semblent raisonnable ici) afin de raffiner les exigences. Ce cycle inclut la planification du développement centré-utilisateur, la définition du contexte d'utilisation, la spécification des exigences, la définition de solutions potentielles et leur évaluation. Une

définition des tâches ou à tout le moins de cas d'utilisation est indiquée. Des scénarios d'usage pourrait apporter une aide à la définition de solutions. La méthode du Magicien d'Oz est particulièrement indiquée pour valider des solutions de conception avec des interfaces vocales. De même, une étude des produits concurrents est essentielle si d'autres produits existent. L'inspection cognitive peut aussi s'avérer intéressante.

3. *Eclipse*, le logiciel utilisé durant les laboratoires pour le développement, possède une interface très riche en fonctionnalité et qui comporte des éléments de plusieurs heuristiques de conception d'interface. Pour chacun des heuristiques suivants, identifiez un élément de l'interface qui le favorise. Justifiez votre réponse et indiquez le ou les bénéfices potentiels de l'heuristique. Au besoin, utilisez la figure de la feuille de l'annexe A pour identifier les éléments auxquels vous référez. (3 pts.)

- (a) Anticipation (Tognazzini)
- (b) Efficacité de l'utilisateur (Tognazzini)
- (c) Exploration (Tognazzini)
- (d) Apprentissage (Tognazzini)
- (e) Métaphore (Tognazzini)
- (f) Garder l'état (Tognazzini)
- (g) Guidage (Bastien et Scapin)
- (h) Incitation (Bastien et Scapin)
- (i) Protection contre les erreurs (Bastien et Scapin)
- (j) Adaptabilité (Bastien et Scapin)

Quelques exemples :

- a. Anticipation (Tognazzini) : le 'X' qui ferme un onglet
 - b. Efficacité de l'utilisateur (Tognazzini) : touches de raccourci
 - c. Exploration (Tognazzini) : le « défaire » universel
 - d. Apprentissage (Tognazzini) : possibilité d'accéder à toute la documentation des librairies
 - e. Métaphore (Tognazzini) : l'icône en forme de 'bug'
 - f. Garder l'état (Tognazzini) : on revient à l'état de la dernière session au démarrage d'Eclipse
 - g. Guidage (Bastien et Scapin) : des valeurs de défauts sont fournies un peu partout
 - h. Incitation (Bastien et Scapin) : indicateurs en marge des programmes pour les erreurs de compilation
 - i. Protection contre les erreurs (Bastien et Scapin) : identification immédiate des erreurs de syntaxe
 - j. Adaptabilité (Bastien et Scapin) : perspectives en fonction des tâches et langages
4. Emacs offre une fonctionnalité très puissante mais comporte aussi une interface complexe à apprendre et que l'on peut considérer à contre courant de plusieurs principes de conception d'interface. Identifiez dans quelle circonstance une telle interface peut s'avérer adéquate et le, ou les facteurs qui expliquent qu'Emacs est encore utilisé de nos jours. (½ page; 2 pts.)
5. Expliquez l'utilité des langages de définition d'interfaces XUL et UIML. (½ page max; 2 pts.)

Ces langages permettent de définir des interfaces utilisateurs, y compris leur comportement. Ils permettent non seulement le prototypage rapide et la personnalisation d'une interface, mais aussi une adaptation en fonction de différentes plate-formes (grandeur d'écran, appareils mobiles). La possibilité d'interpréter ces langages permettrait éventuellement de créer des appareils comme des télécommandes universelles beaucoup plus efficacement.

6. Expliquez la différence fondamentales d'architecture entre les plateformes X-Window et MS-Windows. (½ page max; 2 pts.)

La plateforme X-Window découple les composants comme la bibliothèque graphique, le système de fenêtrage de base, les boîtes à outils et le gestionnaire de fenêtres en modules clairement définis et même en programmes indépendants (sauf pour la bibliothèque graphique). De plus, X-Window définit une couche réseau (client-serveur) entre le système de fenêtrage et l'application (qui utilise une boîte à outil), ce qui permet aux applications d'être d'emblée orientées réseau et multiplate-forme dans la mesure où le serveur peut être implanté sur plusieurs plate-formes. Au contraire, MS-Windows n'est pas orienté client-serveur et ne sépare pas aussi clairement les différents composants mentionnés.

7. Répondez aux questions suivantes par vrai ou faux (2 pts.)

- (a) Le langage Javascript permet d'associer des fonctions à des événements dans un navigateur Web
- (b) XSL permet de transformer des structures XML en HTML
- (c) L'inspection cognitive est une technique d'évaluation d'interface qui s'applique souvent dans la phase de conception de l'interface
- (d) Avec Swing, on établit une connexion au serveur graphique en tout début avec la fonction `System.serverConnect(IPAddress,Port)`

a) vrai, b) vrai, c) vrai, d) faux

8. Consultez, à l'annexe B, le code source du démonstrateur `ToolBarDemo2.java` et l'illustration de la fenêtre qui résulte de son exécution pour répondre aux questions suivantes. (3 pts. au total)

- (a) Vous voulez remplacer les lignes 17 à 21 par l'instruction suivante :

```
addWindowListener(this);
```

Indiquez la (les) ligne (s) que vous devez modifier et le code supplémentaire que vous devez ajouter une fois les lignes 9 à 13 remplacées. Utilisez les numéros de lignes pour préciser où ces modifications se feraient. (2 pts.)

- (b) Un collègue vous recommande de réécrire la fonction `main` de la façon suivante :

```
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        ToolBarDemoTrois frame = null;
        public void run() {
            frame = new ToolBarDemoTrois();
            frame.pack();
            frame.setVisible(true);
        }
    });
}
```

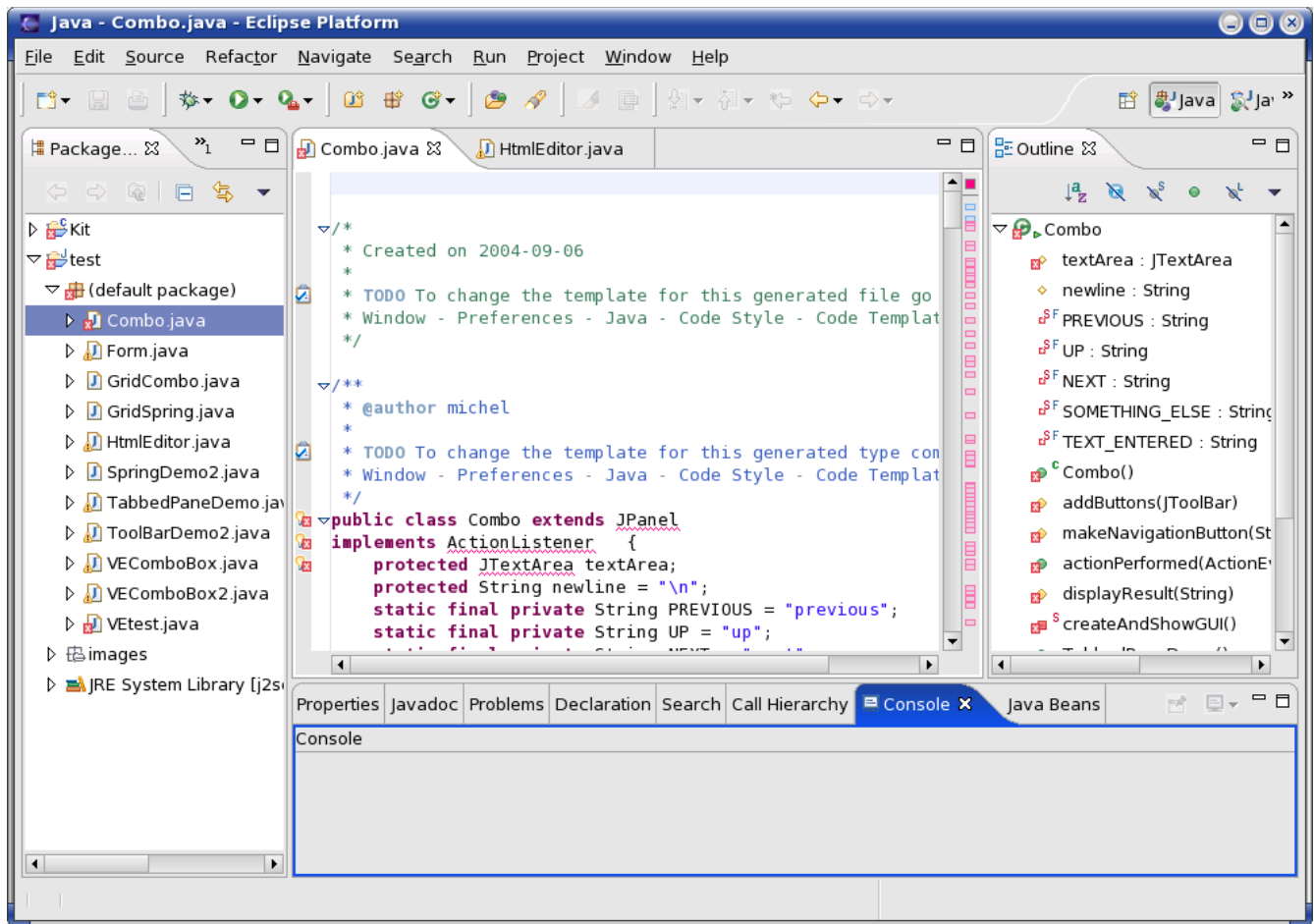
Expliquez pourquoi cette version serait meilleure que l'originale. (1 pt.)

Votre NOM :

MATRICULE :

Annexe A

Interface d'Eclipse. Vous pouvez détacher la feuille et l'insérer dans le cahier de réponse afin de référer à des éléments précis de votre réponse à la question 3. N'oubliez pas d'inscrire votre nom et matricule sur la feuille.



Annexe B : code source de ToolBarDemo2.java (adapté) et illustration de la fenêtre du code exécuté.

```
1 import javax.swing.*
2 import java.awt.*;

3 public class ToolBarDemo2 extends JFrame {
4     protected JTextArea textArea;
5     protected String newline = "\n";

6     public ToolBarDemo2() {
7         //Do frame stuff.
8         super("ToolBarDemo2");
9         addWindowListener(new WindowAdapter() {
10             public void windowClosing(WindowEvent e) {
11                 System.exit(0);
12             }
13         });
14         //Create the toolbar.
15         JToolBar toolBar = new JToolBar();
16         addButtons(toolBar);
17         //Create the text area used for output.
18         textArea = new JTextArea(5, 30);
19         JScrollPane scrollPane = new JScrollPane(textArea);
20         //Lay out the content pane.
21         JPanel contentPane = new JPanel();
22         contentPane.setLayout(new BorderLayout());
23         contentPane.setPreferredSize(new Dimension(400, 100));
24         contentPane.add(toolBar, BorderLayout.NORTH);
25         contentPane.add(scrollPane, BorderLayout.CENTER);
26         setContentPane(contentPane);
27     }

28     protected void addButtons(JToolBar toolBar) {
29         JButton button = null;
30         //first button
31         button = new JButton("Bouton1");
32         button.setToolTipText("This is the left button");
33         button.addActionListener(new ActionListener() {
34             public void actionPerformed(ActionEvent e) {
35                 displayResult("Action for first button");
36             }
37         });
38         toolBar.add(button);
39         //second button
40         button = new JButton(new ImageIcon("images/middle.gif"));
41         button.setToolTipText("This is the middle button");
42         button.addActionListener(new ActionListener() {
43             public void actionPerformed(ActionEvent e) {
44                 displayResult("Action for second button");
45             }
46         });
47         toolBar.add(button);
48     }

49     protected void displayResult(String actionDescription) {
50         textArea.append(actionDescription + newline);
51     }

52     public static void main(String[] args) {
53         ToolBarDemo2 frame = new ToolBarDemo2();
54         frame.pack();
```

```
55         frame.setVisible(true);  
56     }  
57 }
```

Fenêtre correspondant à l'exécution du code et d'un clique de souris sur chaque bouton :

