

École Polytechnique de Montréal  
Département de Génie Informatique  
INF-4100, Sujets spéciaux : le langage JAVA  
Examen de reprise automne 2006

## Corrigé

### Question 1 (3 pts)

1 a (0.5 pts) Donner deux avantages du Java par rapport au C++, lors de la compilation.

**Rep (deux parmi les suivantes):**

- Pas de #include: nombre de ligne à compiler moindre.
- Langage plus simple que le C/C++: compilation plus simple.
- Génère du Bytecode: c'est plus simple que de générer des instructions machine.
- Pas d'étape d'édition des liens.
- Pas de recherche dans les bibliothèques.
- La compilation est Ultra-Rapide.

1 b (0.5 pts) Quelles sont les caractéristiques de la représentation UNICODE de caractères en Java.

**Rep :**

La représentation UNICODE est une norme internationale de représentation des caractères. Elle comprend plus de 130 milles caractères. Cette représentation permet l'uniformité des caractères peu importe le système d'opération.

1 d (0.5 pts.) Expliquer quelles sont les caractéristiques des classes comprises dans la bibliothèque AWT de Java.

**Rep :**

- Ils sont liés directement aux possibilités de l'interface utilisateur graphique de la plate-forme locale;
- Ils s'affichent différemment dépendent de la plate-forme utilisée;
- Ils sont définis comme des composantes lourdes puisqu'ils ne sont pas totalement programmés en JAVA;

1 e (0.5 pts.) Expliquer quelles sont les caractéristiques des classes comprises dans la bibliothèque SWING de Java.

**Rep :**

- Ils sont principalement écrits en Java.
- Ils sont plus lents et plus gourmand en mémoire.
- Ils ont une apparence uniforme peu importe la plate-forme utilisée.

1 f (1.0 pts.) Quelles sont les méthodes essentielles de la classe *JApplet*? Explique ces méthodes brièvement.

**Rep :**

- Méthode "**init**"
  - appelée une seule fois au moment du démarrage de l'applet
  - sert, entre autres, à créer l'interface graphique d'utilisateur
- Méthode "**start**"
  - appelée à chaque fois que le navigateur revient à la page qui contient l'applet y compris lors de l'initialisation
  - utilisé intensément dans les applets multifilaires
- Méthode "**paint**"
  - appelée chaque fois qu'il faut redessiner le contenu de l'écran
  - par le système, ou par l'applet via la méthode "repaint"
- Méthode "**stop**"
  - appelée lorsqu'il est nécessaire de suspendre l'exécution de l'applet; e.g. lorsque l'utilisateur quitte temporairement la page qui est présentée
- Méthode "**destroy**"
  - appelée lors que vient le temps de retirer l'applet de la mémoire

## Question 2 (2 pts)

Soit un programme qui exécute deux « *threads* » *A* et *B* qui doivent synchroniser leurs activités pour gérer deux variables *X* et *Y*. Le thread *B* ne peut pas lire *X* avant que le thread *A* l'écrive. Inversement, le thread *A* ne peut pas lire *Y* avant que *B* l'écrive. Dans le pseudo code suivant, déclarez, initialisez et manipulez les sémaphores pour synchroniser le travail entre les deux threads. (Complétez le code avec les méthodes *P()* et *V()* en utilisant les bonnes sémaphores en paramètre).

```
// Déclaration et initialisation des Sémaphores
```

```
.....
```

```
// Manipulation des Sémaphores
```

```
Thread_A {
    while ( true ) {
        //Produit X
        ecrire (X);

        //Consomme Y
        lire (Y);
    }
}
```

```
Thread_B {
    while ( true ) {
        // Consomme X
        lire (X);

        // Produit Y
        ecrire (Y);
    }
}
```

**Rép :**

```
// Déclaration et initialisation des Sémaphores
```

```
Semaphore s1 = 0;
```

```
Semaphore s2 = 0;
```

```
// Manipulation des Sémaphores
```

```
Thread_A {
    while ( true ) {
        //Produit X
        ecrire (X);
        V(s1)

        //Consomme Y
        P(s2)
        lire (Y);
    }
}
```

```
Thread_B {
    while ( true ) {
        // Consomme X
        P(s1)
        lire (X);

        // Produit Y
        ecrire (Y);
        V(s2)
    }
}
```

### Question 3 (3 pts)

Écrire dans un fichier, ayant le nom « fic.log », tout les noms des fichiers ayant l'extension .jpeg et .jpg qui se trouvent dans le répertoire courant

(`System.getProperty("user.dir")`).

Vous devez utiliser un filtre de noms des fichiers.

Vous devez écrire un nom de fichier par ligne.

Rep :

```
import java.io.*;

public class exercice {

    public static void main(String[] args) {

        File fichier = new File(System.getProperty("user.dir"));
        String[] li = fichier.list(new NameFilter());

        try{
            BufferedWriter myOutput = new BufferedWriter(new FileWriter("fic.log"));

            for(int i=0;i<li.length;i++)
            {
                myOutput.write(li[i]);
                myOutput.newLine();
            }
            myOutput.close();

        }catch (IOException ex) {}
    }

    static public class NameFilter implements FilenameFilter{

        public boolean accept(File dir, String name)
        {
            String ext = name.substring(name.lastIndexOf(".")+1);
            if(ext.equalsIgnoreCase("jpg")) return true;
            if(ext.equalsIgnoreCase("jpeg")) return true;
            return false;
        }
    }
}
```

## Question 4 (12 pts)

Vous devez réaliser un programme qui affiche une image au centre de la fenêtre. Les dimensions de l'image sont spécifiées dans des champs de texte. L'image s'appelle "java.jpg" et elle se trouve dans le répertoire de travail. Les dimensions de l'image sont spécifiées dans des champs de texte. Par la suite, vous devez afficher une chaîne de caractères qui est spécifiée également dans un champ de texte. La chaîne doit être placée au centre de l'écran en y et sur l'axe des x. Il faut également afficher les axes x et y. Le programme contient une interface graphique usager qui doit être créée en utilisant un gestionnaire de disposition de type *GridBagLayout*. (Figure 1). Sur la première ligne de l'interface, on retrouve le champ de texte *lignetext* qui contient la chaîne de caractères à afficher. Le champ de texte doit prendre toute la largeur de la fenêtre, peut importe la taille de celle-ci. Sur la deuxième ligne on retrouve deux champs de texte (un pour la largeur de l'image et un pour la longueur) et un bouton. Les trois éléments prennent un tiers de la largeur de la fenêtre peu importe la taille de celle-ci. Il n'y a pas d'écouteur attaché aux champs de texte, par contre lorsqu'on appuie sur le bouton "Créer" la fenêtre se redessine.

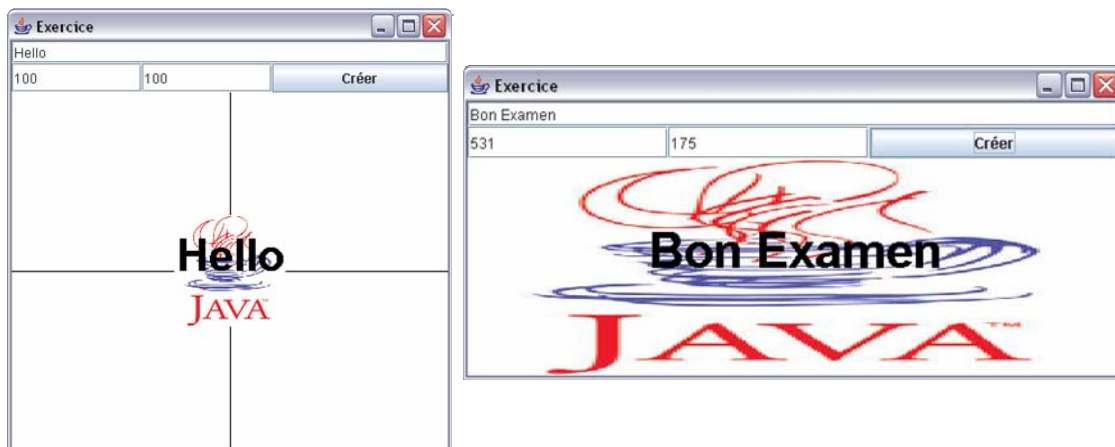


Figure 1: Exécution du programme

```
public class exercice extends JFrame{  
  
    // Le panneau contenant l'affichage  
    private MonPanneau panneau;  
    // Le conteneur contenant le GUI  
    private MonContainer haut;  
    // Le champ de texte qui contient la chaîne de caractères  
    private JTextField lignetext;  
    // Le champ de texte qui contient la coordonnée la longueur du rectangle  
    private JTextField deltax;  
    // Le champ de texte qui contient la coordonnée la largeur du rectangle  
    private JTextField deltay;  
    // Le bouton pour créer le rectangle contenant l'image  
    private JButton boutoncreate;
```

- a) (1.0 pt.) Le constructeur de la classe *exercice*. Vous devez ajouter les dispositions (*MonContainer* et *MonPaneau*) à la disposition de *JFrame*. L'objet *monContainer* doit être en haut et l'objet *Monpaneau* doit être au centre. Vous devez également lancer l'application.

```
public exercice() {
```

```
    super("Exercice");

    // Creer la disposition
    Container c = this.getContentPane();

    // le panneau des boutons
    haut = new MonContainer();
    // le panneau de l'affichage
    panneau = new MonPaneau();

    c.add(haut, BorderLayout.NORTH);
    c.add(panneau, BorderLayout.CENTER);

    // Lancer l'application
    setLocation(100,100);
    setSize(400,400);
    setVisible(true);
```

```
}
private class MonContainer extends Container implements ActionListener{
```

```
    GridBagLayout gbl;
    GridBagConstraints constraints;
```

- b) (1.0 pt.) Le constructeur de la classe *MonContainer*. On doit initialiser le *GridBagLayout* et le *GridBagConstraints*. Il faut créer aussi la disposition.

```
MonContainer()
{
```

```
    super();
    // Creation du layout du haut
    gbl = new GridBagLayout();
    constraints = new GridBagConstraints();
    this.setLayout(gbl);
    AjouterPremiereLigne();
    AjouterDeuxiemeLigne();
```

```
}
```

- c) (2.5 pts.) Les différentes fonctions nécessaires pour la création de la disposition. (si vous n'avez pas besoin des fonctions, le point b vaudra 3.5pts)

```
private void AjouterPremiereLigne()
{
    liegnetext = new JTextField("Hello");
    constraints.weightx = 1;
    constraints.weighty = 1;
    constraints.fill = GridBagConstraints.BOTH;
    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(liegnetext);
}
```

```

private void AjouterDeuxiemeLigne()
{
    // Créer les champs de texte
    deltax = new JTextField("100");
    deltay = new JTextField("100");

    // Ajouter les champs de texte
    constraints.gridwidth = 1;
    addComponent(deltax);
    addComponent(deltay);

    // Créer le bouton
    boutoncreate = new JButton("Créer");
    // Ajouter le bouton
    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(boutoncreate);
    // Ajouter l'écouteur du bouton;
    boutoncreate.addActionListener(this);
}

private void addComponent(Component component)
{
    gbl.setConstraints(component, constraints);
    this.add(component);
}

```

**d) (1 pts) L'implémentation de l'écouteur du bouton qui réaffiche le panneau.**

```

public void actionPerformed(ActionEvent e)
{
    if(e.getSource() == boutoncreate)
    {
        panneau.repaint();
    }
}

```

```

}
private class MonPanneau extends JPanel {

```

```

    // L'image à afficher
    private Image img;

```

**e) (1.5 pt) Le constructeur de la classe *MonPanneau*. Vous devez lire une image sans utiliser *ImageIcon*. Vous devez quand même charger l'image lors de la création. La couleur de l'arrière plan doit être blanche.**

```

public MonPanneau(){
    setBackground(Color.WHITE);

    // On lit l'image
    img = Toolkit.getDefaultToolkit().getImage("java.jpg");

    // On attends la fin du chargement
    MediaTracker traker = new MediaTracker(this);
    traker.addImage(img,0);
    try{
        traker.waitForAll();
    }catch (InterruptedException e) {System.err.println(e.toString());}
}

```

```

}

public void paintComponent(Graphics g){
    super.paintComponent(g);
    int taillex;
    int tailley;
}

```

- f) (1 pt) Vous devez lire les champs de texte `taillex` et `tailley`. Lors de l'entrée d'une chaîne de caractère qui ne contient pas des chiffres dans ces champs, vous devez donner comme valeur à votre image 100 x 100.

```
try{
    taillex= Integer.parseInt(deltax.getText());
    tailley = Integer.parseInt(deltay.getText());
}catch(NumberFormatException nfe)
{
    taillex = 100;
    tailley = 100;
    deltax.setText(String.valueOf(taillex));
    deltay.setText(String.valueOf(tailley));
}
```

- g) (0.5 pts) Vous devez lire la dimension du panneau `dimx` et `dimy`. Si elles sont nulles il faut lancer une exception de type `NumberFormatException`

```
int dimx = this.getWidth();
int dimy = this.getHeight();

if (dimx == 0) throw new NumberFormatException("Taille x de 0");
if (dimy == 0) throw new NumberFormatException("Taille y de 0");
```

```
Graphics2D g2d = (Graphics2D) g;

// Se placer au centre de l'image
g2d.translate(dimx*0.5,dimy*0.5);
```

- h) (0.5 pts) Vous devez dessiner les axes du panneau en noir. Le centre est au milieu de la fenêtre.

```
g2d.setColor(Color.BLACK);
g2d.drawLine((int)(dimx*-0.5),0,(int)(dimx*0.5),0);
g2d.drawLine(0,(int)(dimy*-0.5),0,(int)(dimy*0.5));
```

- i) (1.5 pts) Vous devez dessiner l'image au milieu de la fenêtre. Cette image doit avoir la taille spécifiée dans les champs de texte en x et en y. Il faut absolument utiliser la fonction `DessinerImage(g2d)` pour dessiner l'image.

```
// Afficher le rectangle
AffineTransform aft = g2d.getTransform();
double scalex = (double)taillex/img.getWidth(this);
double scaley = (double)tailley/img.getHeight(this);
g2d.scale(scalex,scaley);
g2d.translate(img.getWidth(this)*-0.5,img.getHeight(this)*-0.5);
DessinerImage(g2d);
g2d.setTransform(aft);
```

- j) (1.5 pts) Vous devez afficher le texte qui est dans le champ de texte `lignetext` centre sur l'axe y et placé sur l'axe des x. Le texte doit être « Arial » avec une fonte gras et une taille de 40. Vous devez absolument utiliser la fonction `DessinerTexte(g2d)` pour dessiner le texte.

```
// Afficher le texte
Font fnt = new Font("Arial",Font.BOLD,40);
g2d.setFont(fnt);
```

```
FontMetrics fm = g2d.getFontMetrics();
int wf = (int) (fm.stringWidth(lignetext.getText())*0.5);
g2d.translate(-wf,0);
DessinerTexte(g2d);
```

```
// si la fenetre est nulle, il ne se passe rien
}catch(NumberFormatException nfe) {}
}
```

```
private void DessinerImage(Graphics2D g2d)
{
    g2d.drawImage(img,0,0,this);
}
```

```
private void DessinerTexte(Graphics2D g2d)
{
    g2d.setColor(Color.BLACK);
    g2d.drawString(lignetext.getText(),0,0);
}
```

```
}
```

```
public static void main(String[] args) {
    exercice exo = new exercice();
    exo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

```
}
```