# Développement Logiciel
## Examen 2010-2011 (vendredi 27 mai 2011, 9h-11h)

*Seuls les documents liés **directement** au cours sont autorisés (comme dit sur le site): sujets de TD, notes de cours, notes personnelles manuscrites, version imprimée du cours. En cas de doute, donnez la réponse qui vous semble la plus vraisemblable (et justifiez au besoin). Ce sujet contient 10 pages.*

***N'OUBLIEZ PAS DE NUMÉROTER VOS FEUILLES, DE REPORTER VOS NOMS/PRÉNOMS ET DE RENDRE VOS COPIES ANONYMES.** Les parties 1 et 2 doivent être rendues sur des feuilles séparées.*

**PARTIE 1 : Questions de cours** — **1 point par question**

On vous demande des réponses courtes et précises (1 à 2 lignes max).

### MÉTHODES DE PROGRAMMATION

Voici le contenu du fichier Rule.java :

```
package fr.google.search
public class Rule{
        char operator;
        String title;
        private int index;
        protected int nbCalls;
        public char returnCharAt ( char[] query, int index ) { return query[index]; }
        public static void test(){
                Rule r = new Rule();
                r.returnCharAt("abc",5);
        }
        public static void main (String[] args) { }
}
```

**Q1.** Quelle est le nom complet de la classe?

**Q2.** Donnez la visibilité des champs "title" et "nbCalls" pour tous les cas (avec ou sans relation d'héritage, même package ou non)

**Q3.** Que se passe t'il lors de l'appel à la méthode test? (écrivez le résultat de l'appel)

**Q4.** Proposez une amélioration de la méthode returnCharAt.

### ENTRÉES-SORTIES

**Q5.** Ecrivez le plus simplement possible une méthode qui compte le nombre d'occurrences du caractère 'a' dans un fichier "input.txt" (le fichier n'est pas ouvert au début de la méthode). La signature de la méthode est la suivante: "int compteur_de_a() {...}"

**Q6.** A quoi sert le mot clé "transient"? (donnez le contexte)



### PROGRAMMATION PARALLÈLE

**Q7.** A quoi sert la méthode "yield()" ? (donnez le contexte)

**Q8.** Qu'est ce qu'un interblocage? Donnez un exemple simple.

### INTERFACE GRAPHIQUE

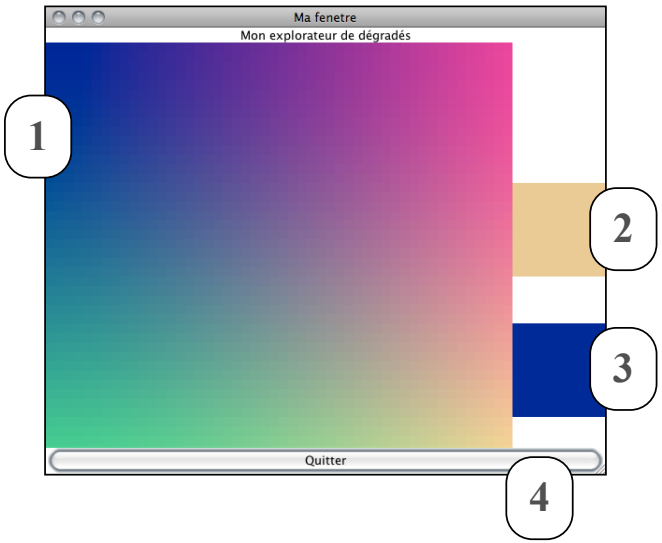
**Q9.** Qu'est qu'un LayoutManager? Peut-on les empiler?

**Q10.** Donnez un exemple de composant racine, de composant intermédiaire, de composant atomique

**Q11.** Comment capturer les actions de l'utilisateur dans une fenêtre graphique?

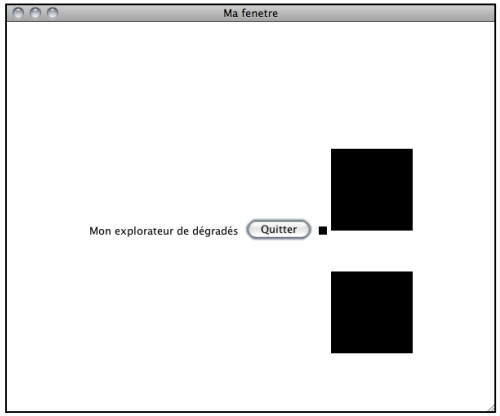
**PARTIE 2.A : SWING** — **5 points**

On souhaite réaliser une fenêtre permettant à un utilisateur de choisir une couleur dans un dégradé :



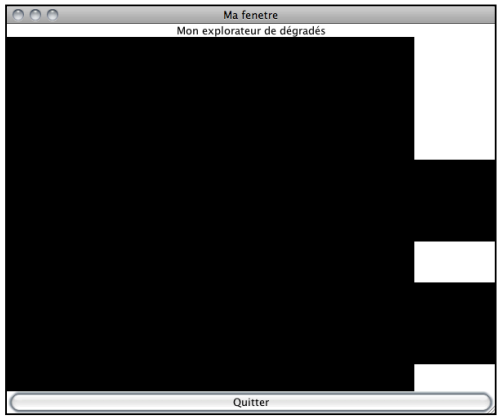

Quand l'utilisateur passe son curseur sur le dégradé (1), la couleur survolée est mise à jour dans le cadre (2) ; s'il clique, la couleur sélectionnée est affichée dans le cadre (3). L'utilisateur peut aussi fermer la fenêtre à l'aide du bouton "Quitter" (4).

Le code en annexe I tente d'implémenter cette fenêtre. Cependant, en l'état, il produit le résultat suivant :




**Question 1**: D'où vient le problème de mise en page ? Pourriez-vous le résoudre en modifiant une seule instruction ?

On suppose que vous avez résolu la question précédente. Vous obtenez ceci :




**Question 2 :** Pourquoi le dégradé ne s'affiche-t-il pas ?

**Question 3 :** Quand l'utilisateur passe son curseur au dessus de la zone de dégradé, les zones de couleurs ne se mettent pas à jour. Qu'a oublié le développeur ?

**Question 4 :** Le bouton "Quitter" ne fait rien. Que manque-t-il au code ?



## PARTIE 2.B : THREADS — 2 points

Le code présenté en annexe II simule une situation dans laquelle plusieurs threads (les objets `Worker`) accèdent à une ressource partagée (`data`) en lecture et en écriture.

Ces objets `Worker` lisent puis détruisent un certain nombre (`READ_SIZE`) d'éléments de `data`, effectuent des calculs à partir de ce qu'ils ont lu, et remettent dans data le résultat de leurs calculs. Ce résultat sera donc intégré au calcul du prochain `Worker` qui accédera à `data`.

Les `Workers` doivent s'arrêter quand le nombre d'éléments dans `data` est inférieur à `READ_SIZE`. Le premier `Worker` à observer cela doit afficher ces éléments, puis tous les threads doivent s'arrêter. Un seul thread doit afficher ces éléments.

Actuellement, trois types d'erreurs surviennent :

- Des `NullPointerExceptions` sont levées à la ligne 32,
- Des `ConcurrentModificationExceptions` sont levées à la ligne 23,
- Tous les threads affichent le contenu de `data` quand `data.size()` est inférieur à `READ_SIZE`.

**Question 1:** Comment résoudre simplement ces problèmes ?

## PARTIE 2.C : SERIALISATION — 3 points

La classe `WritableClass` (annexe III) est sérialisable. Ecrivez le contenu des méthodes `writeObject` et `readObject` en prenant en compte le fait qu'on ne veut pas inclure l'objet `File` (paramètre `f`) dans la sérialisation, on ne veut garder que la chaîne de caractères correspondant au chemin du fichier. Cette chaîne de caractères sera utilisée lors de la dé-sérialisation pour récupérer l'objet `File`.

**Question 1:** Complétez les fonctions correspondantes.

La méthode `main(…)` permet de tester la sérialisation et la dé-sérialisation. Cependant, en l'état, une erreur est retournée :

```
java.io.FileNotFoundException: monFichierLecture.sav (No such file or directory)
      at java.io.FileInputStream.open(Native Method)
      at java.io.FileInputStream.<init>(FileInputStream.java:106)
      at java.io.FileInputStream.<init>(FileInputStream.java:66)
      at exoSerial.Exercice.main(Exercice.java:76)
4, 9, d, Une chaine de test, [3.0, 0.5, 0.8, 44.33], /Users/Exam2011/fichier.txt
null
```

**Question 2 :** Quelle erreur a été commise par le développeur dans sa méthode de test ?

FIN DU SUJET.

# Annexe I

```
static Color background = Color.white;
static Color couleurChoisie = new Color(0);
static Color couleurSurvolee = new Color(0);

public static void main(String[] args) {

	// Création de la fenêtre

	JFrame frame = new JFrame("Ma fenetre");
	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	frame.setBackground(background);
	frame.setPreferredSize(new Dimension(600, 500));

	frame.setLayout(new FlowLayout());

	// Un titre, en haut de la fenêtre

	JLabel title = new JLabel("Mon explorateur de dégradés");
	title.setHorizontalAlignment(JLabel.CENTER);
	frame.add(title, BorderLayout.PAGE_START);

	// Un bouton pour quitter, en bas de la fenêtre

	JButton exit = new JButton("Quitter");
	ActionListener exitAction = new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {
			System.out.println("Quit");
			System.exit(0);
		}
	};
	frame.add(exit, BorderLayout.PAGE_END);

	// Dessin du dégradé

	JPanel gradient = new JPanel() {
		@Override
		public void paint(Graphics g) {
			super.paint(g);

			for ( int i = 0 ; i < 255 ; i++ ) {
				for ( int j = 0 ; j < 255 ; j++ ) {
					new Color( i, j, 0 );
					g.fillRect(i * 2, j * 2, 2, 2);
				}
			}

		}
	};
	frame.add(gradient, BorderLayout.CENTER);

	// Affichage de la couleur choisie

	final JPanel chosenColor = new JPanel() {
		@Override
		public void paint(Graphics g) {
			super.paint(g);

			g.setColor(couleurChoisie);
			g.fillRect(0, 150, 100, 100);

			g.setColor(couleurChoisie);
			g.fillRect (0, 300, 100, 100);
		}
	};
	chosenColor.setBackground(background);
	chosenColor.setPreferredSize(new Dimension(100, 500));

	// Sélection de la couleur

	MouseAdapter colorPicker = new MouseAdapter() {
		@Override
		public void mouseMoved(MouseEvent e) {
			couleurChoisie = new Color(e.getX() / 2, e.getY() / 2, 0);
		}

		@Override
		public void mouseClicked(MouseEvent e) {
			couleurChoisie = couleurSurvolee;
		}
	};
	gradient.addMouseMotionListener(colorPicker);
	gradient.addMouseListener(colorPicker);
	frame.add(chosenColor, BorderLayout.LINE_END);

	// Affichage de la fenêtre

	frame.pack();
	frame.setVisible(true);
}
```



Développement Logiciel, Examen 2010-2011, Session 1

# Annexe II

```
static ArrayList<Integer> data;

public static class Worker extends Thread {

	private static final int READ_SIZE = 10;
	private static int currentId = 0;

	private int id;

	public Worker() {
		id = currentId;
		currentId++;
	}

	@Override
	public void run() {

		while (true) {

			int result = 0;

			if (READ_SIZE > data.size()) {
				for (Integer i : data) {
					System.out.println("[" + id + "] " + i);
				}
				return;
			}

			for (int i = 0 ; i < READ_SIZE && i < data.size() ; i++) {

				try {
					result += data.get(0).intValue();
				} catch (NullPointerException e) {
					e.printStackTrace();
				}

				data.remove(0);

			}

			result /= READ_SIZE;

			data.add(new Integer(result));
		}
	}

}

public static void main(String[] args) {

	System.out.println("start");

	data = new ArrayList<Integer>(100000);

	for (int i = 0 ; i < 100000 ; i++) {
		data.add(new Integer((int)(Math.random() * 2000)));
	}

	for (int i = 0 ; i < 10 ; i++) {
		Worker r = new Worker();
		r.start();
	}
}
```



Développement Logiciel, Examen 2010-2011, Session 1

# Annexe III

```
1     public static class WritableClass implements Serializable {

            int a;
            int b;
5           char c;
            String d;
            Vector<Double> e;
            File f;

10          public WritableClass(int a, int b, char c, String d, double[] ePrime,
                        File f) {
                  this.a = a;
                  this.b = b;
                  this.c = c;
15                this.d = d;
                  this.e = new Vector<Double>(ePrime.length);
                  for (double ev : ePrime) {
                        e.add(new Double(ev));
                  }
20                this.f = f;

            }

            @Override
25          public String toString() {

                  return "" + a + ", " + b + ", " + c + ", " + d + ", " + e + ", "
                              + f.getAbsolutePath();

30          }

            private void writeObject(ObjectOutputStream out) throws IOException {

                  // PARTIE A COMPLETER
35
            }

            private void readObject(ObjectInputStream in) throws IOException,
                        ClassNotFoundException {
40
                  // PARTIE A COMPLETER

            }

45    }

      public static void main(String[] args) {

            WritableClass test = new WritableClass(4, 9, 'd', "Une chaine de test",
50                      new double[]{3, .5, .8, 44.33}, new File("fichier.txt"));
            WritableClass test2 = null;

            try {
                  ObjectOutputStream out = new ObjectOutputStream(
55                            new FileOutputStream("monFichierEcriture.sav"));
                  out.writeObject(test);
                  out.close();
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
60          } catch (IOException e) {
                  e.printStackTrace();
            }

            try {
65                ObjectInputStream in = new ObjectInputStream(
                              new FileInputStream("monFichierLecture.sav"));
                  test2 = (WritableClass)(in.readObject());
                  in.close();
            } catch (FileNotFoundException e) {
70                e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ClassNotFoundException e) {
                  e.printStackTrace();
75          }

            System.out.println(test);
            System.out.println(test2);

80    }
```



Développement Logiciel, Examen 2010-2011, Session 1