# Génie logiciel

—

# TD4 - Questions complémentaires

Pietri Yoann (A11)
Levy–Falk Hugo (A11)

Mars 2018

# Table des matières

# 1 lecture dans un fichier et transformation d'une chaîne de caractères

Classe `Utilitaire` :

```
package questioncomplementaire;

import java.io.BufferedReader;
import java.io.FileReader;

/**
 * @author galtier
 *
 */
public class Utilitaire {

	/**
	 * Lit un fichier texte et place les caractères lus dans une chaîne de
	 * caractères.
	 *
	 * @param nomFichier
	 *            le nom complet du fichier à lire
	 * @return une chaine de caractères contenant le texte lu dans le fichier.
	 *         Si le fichier n'existe pas ou ne peut pas être lu, retourne null.
	 *
	 */
	public static String lireTexte(String nomFichier) {
		try {
			String texte = "";
			FileReader fr = new FileReader(nomFichier);
			BufferedReader br = new BufferedReader(fr);
			String ligne;
			while ((ligne = br.readLine()) != null)
				texte += ligne + "\n";
			return texte;
		} catch (Exception e) {
			e.printStackTrace();
			return null;
		}
	}

	/**
	 * Supprime tous les caractères qui ne sont pas des lettres
	 * et met les autres en majuscules.
	 * @param input La chaîne à traîter.
	 * @return La chaîne de caractères traîtée.
	 */
	public static String getLettres(String input) {
		return input.replaceAll("[^a-zA-Z]*", "").toUpperCase();
	}
}
```

# 2 Codage complet des classes Morse et Cesar

## 2.1 Classe `Cesar`

```
package questioncomplementaire;

public class Cesar extends AlgoCodage{
	private int decalage;
	private String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
        public Cesar(int decalage){
                this.nom = "César";
                this.decalage = decalage;
        }

        @Override
        public String toString() {
                return "codage de César (décalage=" + decalage + ")";
        }

        @Override
        public String encode(String s) {
                String encode = "";
                for(int i=0;i<s.length();i++){
                        encode += alphabet.charAt((decalage +
                                                alphabet.indexOf(s.charAt(i)))%alphabet.length());
                }
                return encode;
        }

        @Override
        public String decode(String s) {
                String decode = "";
                int index;
                for(int i=0; i<s.length(); i++) {
                        index = (alphabet.indexOf(s.charAt(i))-decalage) % alphabet.length();
                        index += (index<0)?alphabet.length():0;
                        decode += alphabet.charAt(index);
                }
                return decode;
        }
}
```

## 2.2 Classe Morse

```
package questioncomplementaire;

import java.util.HashMap;

public class Morse extends AlgoCodage{

        private String[] alphabet;
        private HashMap<String, String> decoder;

        public Morse(){
                this.nom = "Morse";
                alphabet = new String[] {
                                ".-", "-...", "-.-.", "-..", ".", "..-.",
                                "--.", "....", "..", ".---", "-.-", ".-..",
                                "--", "-.", "---", ".--.", "--.-", ".-.",
                                "...", "-", "..-", "...-", ".--", "-..-",
                                "-.--", "--.."
                };
                decoder = new HashMap<String, String>();
                int a = 65;
                for (int i = 0; i < alphabet.length; i++) {
                        decoder.put(alphabet[i], String.valueOf((char)(a+i)));
                }
        }

        @Override
```

```
        public String toString() {
                return "Morse";
        }

        @Override
        public String encode(String s) {
                String encode = "";
                int a = 65; // Code ASCII de A
                for (int i = 0; i < s.length(); i++) {
                        encode += alphabet[(int)s.charAt(i)-a] + "/";
                }
                return encode;
        }

        @Override
        public String decode(String s) {
                String decode = "";
                int a = 65; // Code ASCII de A
                String[] lettres = s.split("/");
                for (int i = 0; i < lettres.length; i++) {
                        decode += decoder.get(lettres[i]);
                }
                return decode;
        }
}
```

## 3 Mesure des temps d'exécution

Classe Comparateur :

```
package questioncomplementaire;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class Comparateur {

        public static void main(String[] arg) {
                PrintWriter out;
                try {
                        out = new PrintWriter("resultats.txt");
                } catch (FileNotFoundException e) {
                        System.out.println("Impossible d'ouvrpour les résultats.");
                        e.printStackTrace();
                        return;
                }
                int TAILLE_EXTRAIT = 50;
                String texteClair = Utilitaire.lireTexte("texte1.txt");
                out.println("début du texte original : "
                                + texteClair.substring(0, Math.min(50, texteClair.length())));
                texteClair = Utilitaire.getLettres(texteClair);
                out.println("texte après `getLettres` : "
                                + texteClair.substring(0, Math.min(50, texteClair.length())));

                ArrayList<AlgoCodage> algoAComparer = new ArrayList<AlgoCodage>();
                algoAComparer.add(new Cesar(3));
                algoAComparer.add(new Morse());
                algoAComparer.add(new Cesar(10));
                algoAComparer.add(new CodageAClef("mon secret"));
                algoAComparer.add(new CodageAClef(""));
```

```
                algoAComparer.add(new Huffman(texteClair));

                String encodee = "";
                for (AlgoCodage algo : algoAComparer) {
                        out.println("algorithme : " + algo.toString());
                        long avant = System.currentTimeMillis();
                        encodee = algo.encode(texteClair);
                        long apres = System.currentTimeMillis();
                        long duree = apres - avant;
                        out.println("\tencodage : "
                                + "\t durée : "
                                + duree
                                + "\t résultat : "
                                + encodee.substring(0, Math.min(50, encodee.length())));
                        avant = System.currentTimeMillis();
                        String decodee = algo.decode(encodee);
                        apres = System.currentTimeMillis();
                        duree = apres - avant;
                        out.println("\tdécodage : "
                                        + "\t durée : "
                                        + duree
                                + "\t résultat : "
                                + decodee.substring(0, Math.min(50, decodee.length())));
                }
                out.close();
        }
}
```

## 4 Utilisation d'un patron pour ajouter un algorithme existant

Classe `CodageAClef` :

```
package questioncomplementaire;

import chiffrement.EncodeurAClef;

public class CodageAClef extends AlgoCodage {

        EncodeurAClef encodeur;
        String clef;

        public CodageAClef(String clef) {
                if(clef.equals(""))
                        this.clef = "clef par défaut";
                else
                        this.clef = clef;

                try {
                        encodeur = new EncodeurAClef(this.clef);
                } catch (Exception e) {
                        System.out.println("Échec de la création de l'encodeur");
                        e.printStackTrace();
                }
        }

        @Override
        public String encode(String s) {
                return encodeur.chiffre(s);
        }

        @Override
```

```
        public String decode(String s) {
                return encodeur.dechiffre(s);
        }

        @Override
        public String toString() {
                return "codage à clef (clef = \"" + clef + "\")";
        }

}
```

# 5 Codage d'Huffman

Classe `Huffman` :

```
package questioncomplementaire;

import java.util.ArrayList;

public class Huffman extends AlgoCodage {
        private String generatrice;
        private ArrayList<HuffmanCode> codes;
        private Arbre arbre;

        public Huffman(String generatrice) {
                this.generatrice = generatrice;
                arbre = Arbre.buildTree(generatrice);
                codes = arbre.collectCodes(new StringBuffer(""));

        }

        @Override
        public String encode(String s) {
                String encode = s;
                for (HuffmanCode huffmanCode : codes) {
                        encode = encode.replaceAll(
                                        String.valueOf(huffmanCode.lettre), huffmanCode.code);
                }
                return encode;

        }

        @Override
        public String decode(String s) {
                String decode = "";
                Arbre current = arbre;
                for (int i=0; i<s.length(); i++) {
                        if(current instanceof Feuille) {
                                decode += ((Feuille) current).lettre;
                                current = arbre;
                        }
                        if(current instanceof Noeud) {
                                if(s.charAt(i) == '0')
                                        current = ((Noeud)current).filsGauche;
                                else
                                        current = ((Noeud)current).filsDroit;
                        }
                }
                return decode;
        }
```

```
        @Override
        public String toString() {
                return "Huffman (génératrice = "
                                + generatrice.substring(0, Math.min(50, generatrice.length()))
                                + ")";
        }

}
```

# 6 Documentation

## 6.1 Classe Noeud

```
package questioncomplementaire;

/**
 * @author galtier
 *
 * Cette classe représente un noeud d'un arbre de Huffman.
 *
 */
public class Noeud extends Arbre {
        final Arbre filsGauche, filsDroit;

        /**
         *
         * @param filsGauche fils gauche du noeud
         * @param filsDroit fils droit du noeud
         */
        public Noeud(Arbre filsGauche, Arbre filsDroit) {
                super(filsGauche.frequence + filsDroit.frequence);
                this.filsGauche = filsGauche;
                this.filsDroit = filsDroit;
        }
}
```

## 6.2 Classe Arbre

```
package questioncomplementaire;

/**
 * @author galtier
 *
 * Cette classe représente un noeud d'un arbre de Huffman.
 *
 */
public class Noeud extends Arbre {
        final Arbre filsGauche, filsDroit;

        /**
         *
         * @param filsGauche fils gauche du noeud
         * @param filsDroit fils droit du noeud
         */
        public Noeud(Arbre filsGauche, Arbre filsDroit) {
                super(filsGauche.frequence + filsDroit.frequence);
                this.filsGauche = filsGauche;
                this.filsDroit = filsDroit;
        }
}
```

### 6.3 Classe HuffmanCode

```
package questioncomplementaire;

/**
 * @author galtier
 *
 * Cette classe représente un noeud d'un arbre de Huffman.
 *
 */
public class Noeud extends Arbre {
	final Arbre filsGauche, filsDroit;

	/**
	 *
	 * @param filsGauche fils gauche du noeud
	 * @param filsDroit fils droit du noeud
	 */
	public Noeud(Arbre filsGauche, Arbre filsDroit) {
		super(filsGauche.frequence + filsDroit.frequence);
		this.filsGauche = filsGauche;
		this.filsDroit = filsDroit;
	}
}
```

## 7 Sauvegarde des résultats

Voir la classe `Comparateur` dans la section "Mesure des temps d'exécution".