

Interface-Homme Machine / Travaux pratiques

S03 – Générateur de mots de passe –PASTIS
Conception d'interfaces / Maquettes



19.05.2015

Adriano De Almeida Silva – T-1f

Alex Travasso – T-1f

Tables des matières

1.	Introduction.....	3
2.	Programme.....	3
1.1.	Maquette.....	3
1.2.	Modèle	3
1.3.	Contrôleur	8
1.4.	View	9
1.5.	View.fxml	10
3.	Conclusion	10

1. Introduction

Pour ce travail pratique nous allons devoir créer un générateur de mots de passe qui prendra en compte plusieurs critères selon l'envie de l'utilisateur.

2. Programme

1.1. Maquette

Voici la maquette de l'application telle qu'elle a été imaginée au début du travail pratique.

La maquette de l'application 'Pastis' est présentée dans un cadre gris. Le titre 'Pastis' est en haut au centre. En dessous, à gauche, se trouvent huit paramètres de configuration : 'Longueur du mot de passe', 'Lettres majuscules', 'Lettres minuscules', 'Chiffres', 'Caractères spéciaux', 'Prononçable', 'Caractères ambigus' et 'Niveau de sécurité'. À droite de ces paramètres sont des contrôles : un champ de texte pour la longueur (contenant '12'), des cases à cocher pour les lettres, chiffres et caractères spéciaux (toutes cochées), des cases à cocher pour 'Prononçable' et 'Caractères ambigus' (toutes cochées), et un slider pour le 'Niveau de sécurité'. En bas à gauche, il y a un bouton 'Générer' en bleu. En bas à droite, le mot 'password' est écrit.

1.2. Modèle

```
package s03;
import java.util.Random;
import javafx.scene.input.Clipboard;
import javafx.scene.input.ClipboardContent;
public class PastisModel implements IPastisModel {
    private Random r = new Random();
    private String mdp = new String();
    private char[] chars;
    // Longueur par défaut
    private int length = 12;
    private boolean ucLetters;
    private boolean lcLetters;
    private boolean digits;
```

```
private boolean symbols;
private boolean pronounceable;
private boolean unAmbiguous;
// Constructeur
public PastisModel() {
}
public String getNewPassword() {
    int count = 0;
    // recoit la longueur du mdp
    length = pwLength();
    chars = new char[length * 4];
    // si minuscules
    if (withLcLetters()) {
        chars = getLcLetters().toCharArray();
        count = 26;
    }
    // si majuscules
    if (withUcLetters()) {
        String strtemp = getUcLetters();
        for (int i = 0; i < chars.length; i++) {
            strtemp += chars[i];
            count++;
        }
        char[] temp = strtemp.toCharArray();
        chars = temp;
    }
    // si digits
    if (withDigits()) {
        String strtemp = getDigits();
        for (int i = 0; i < chars.length; i++) {
            strtemp += chars[i];
            count++;
        }
        char[] temp = strtemp.toCharArray();
        chars = temp;
    }
}
```

```
// si symbols
if (withSymbols()) {
    String strtemp = getSymbols();
    for (int i = 0; i < chars.length; i++) {
        strtemp += chars[i];
        count++;
    }
    char[] temp = strtemp.toCharArray();
    chars = temp;
}

// creation du mdp
StringBuilder sb = new StringBuilder();
for (int i = 0; i < length; i++) {
    char x = '\0';
    while (x == '\0') {
        x = chars[r.nextInt(chars.length)];
    }
    char c = chars[r.nextInt(length)];
    sb.append(x);
}
mdp = sb.toString();

// si caracteres ambigus
if (isUnambiguous()) {
    if (mdp.contains("i") || mdp.contains("l") || mdp.contains("1")
        || mdp.contains("0") || mdp.contains("O")) {
        mdp = getNewPassword();
    }
}

// insertion dans le clipboard
final Clipboard clipboard = Clipboard.getSystemClipboard();
final ClipboardContent cbContent = new ClipboardContent();
cbContent.putString(mdp);
clipboard.setContent(cbContent);

return mdp;
```

```
}

// -----
// Getters
// -----

public int pwLength() {
    return length;
}

@Override
public boolean withUcLetters() {
    return ucLetters;
}

@Override
public boolean withLcLetters() {
    return lcLetters;
}

@Override
public boolean withDigits() {
    return digits;
}

@Override
public boolean withSymbols() {
    return symbols;
}

@Override
public boolean isPronounceable() {
    return pronounceable;
}

@Override
public boolean isUnambiguous() {
    return unAmbiguous;
}
```

```
// -----  
// Setters  
// -----  
  
public void setLength(int length) {  
    this.length = length;  
}  
  
public void setUcLetters(boolean ucLetters) {  
    this.ucLetters = ucLetters;  
}  
  
public void setLcLetters(boolean lcLetters) {  
    this.lcLetters = lcLetters;  
}  
  
public void setDigits(boolean digits) {  
    this.digits = digits;  
}  
  
public void setSymbols(boolean symbols) {  
    this.symbols = symbols;  
}  
  
public void setPronounceable(boolean pronounceable) {  
    this.pronounceable = pronounceable;  
}  
  
public void setUnAmbiguous(boolean unAmbiguous) {  
    this.unAmbiguous = unAmbiguous;  
}  
  
}
```

1.3. Contrôleur

```
package s03;

import javafx.fxml.FXML;
import javafx.scene.control.*;

public class PastisControler {

    PastisModel mdl = new PastisModel();

    @FXML
    private TextField longueur;
    @FXML
    private CheckBox majuscules;
    @FXML
    private CheckBox minuscules;
    @FXML
    private CheckBox chiffres;
    @FXML
    private CheckBox charSpec;
    @FXML
    private CheckBox pronon;
    @FXML
    private CheckBox charAmbigus;
    @FXML
    private Button generate;
    @FXML
    private TextField password;
    @FXML
    private ProgressBar robust;
    @FXML
    private Tooltip tooltip;
```



```
@FXML
public void handleGenerateButton() {
    mdl.setLength(Integer.valueOf(longueur.getText()));
    mdl.setLcLetters(minusculs.isSelected());
    mdl.setUcLetters(majusculs.isSelected());
    mdl.setDigits(chiffres.isSelected());
    mdl.setSymbols(charSpec.isSelected());
    mdl.setPronounceable(pronon.isSelected());
    mdl.setUnAmbiguous(charAmbigus.isSelected());
    password.setText(mdl.getNewPassword());
    robuste(password.getText().length());
}

public void robuste(int longueur){
    if (longueur < 5){
        robust.setProgress(0.25);
    }else if(longueur >=5 && longueur < 8){
        robust.setProgress(0.5);
    }else if(longueur >=8 && longueur < 10){
        robust.setProgress(0.75);
    }else if(longueur >10 ){
        robust.setProgress(1.0);
    }
}
}
```

1.4. View

```
package s03;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
```

```
public class PastisView extends Application {  
    PastisController ctrl = new PastisController();  
    IPastisModel mdl = new PastisModel();  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
  
        FXMLLoader loader = new  
FXMLLoader(getClass().getResource("PastisView.fxml"));  
        BorderPane root = loader.load();  
        loader.setController(ctrl);  
        Scene scene = new Scene(root);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
        primaryStage.setTitle("Password Generator");  
    }  
  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
}
```

1.5. View.fxml

Voici la vue de l'application.



The screenshot shows the user interface of the 'Pastis' password generator. At the top, there is a logo with the word 'PASTIS' in a stylized blue font. Below the logo, the interface is organized into several sections. The first section is labeled 'Longueur du mot de passe' (Password length) and features a text input field containing the number '12'. The second section contains five checkboxes for password requirements: 'Lettres majuscules' (Uppercase letters), 'Lettres minuscules' (Lowercase letters), 'Chiffres' (Numbers), 'Caractères spéciaux' (Special characters), and 'Prononçable' (Pronounceable). The third section is labeled 'Robustesse' (Robustness) and includes a horizontal slider control. At the bottom right, there is a blue button labeled 'Générer' (Generate). A large, empty rectangular text box is positioned at the bottom center of the interface, likely for displaying the generated password.

3. Conclusion

Le travail pratique s'est bien déroulé en général. Nous avons rencontré des difficultés lors de l'implémentation de la méthode qui génère les mots de passe car nous n'avions pas imaginé correctement la façon dont celle-ci allait fonctionner.