Донецький Національний Технічний Університет

Лабораторна робота № 5

«Дискреційна модель»

Виконав:

ст. групи ІПЗІм -17

Лисенко А. С.

Перевірила:

доцент каф. ПМІ

Маслова Н. О.

Покровськ 2017

|  |  |  |
| --- | --- | --- |
| Варіант | Кількість суб’єктів  доступу (користувачів) | Кількість об’єктів  доступу |
| 9 | 6 | 4 |

Під політикою безпеки розуміють набір норм, правил і практичних прийомів, що регулюють управління, захист і розподіл цінної інформації. Політика безпеки задає механізми управління доступу до об'єкту, визначає як дозволені, так і заборонені доступи.

Політика безпеки реалізується через адміністративно−організаційні заходи, фізичні і програмно−технічні засоби та визначає архітектуру системи захисту. Для конкретної організації політика безпеки повинна носити індивідуальний характер і залежати від конкретної технології обробки інформації і тих програмних і технічних засобів, що використовуються.

Політика безпеки визначається способом управління доступом, який задає порядок доступу до об'єктів системи. Розрізняють два основних види політики безпеки: виборчу і повноважну.

Алгоритм формування матриці доступу користувачів до об'єктів комп'ютерної системи.

Алгоритм формування матриці виконується за допомогою функції **public static** String randomCharacter(String characters)яка повертає випадковий символ r – read, w – write. Нижче приведено опис функції

**public static** String randomCharacter(String characters)  
{  
 **int** n = characters.length();  
 **int** r = (**int**) (n \* Math.*random*());  
 **return** characters.substring(r, r + 1);  
}

Заповнення матриці доступу.

Матриця доступу заповнюється за допомогою **private** ObservableList<Person> **personList** = FXCollections.*observableArrayList*(); **personList** включає в себе метод **personList**.add(person); за допомогою якого виконується додавання об'єктів в матрицю. Нижче приведено пример опису заповнення матриці

**personList**.clear();

**personList**.add(**new** Person(**"Administrator"**,**"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**));  
**personList**.add(**new** Person(**"User"**, **password**,**password1**,**password2**,**password3**));  
**personList**.add(**new** Person(**"User1"**, **password2**,**password2**,**password3**,**password2**));  
**personList**.add(**new** Person(**"User2"**, **password3**,**password1**,**password3**,**password1**));  
**personList**.add(**new** Person(**"User3"**, **password2**,**password3**,**password1**,**password1**));  
**personList**.add(**new** Person(**"User4"**, **password2**,**password**,**password2**,**password3**));

Заповнення та читання матриці від User, виконується через запис та читання з файлу в матрицю. Нижче приведено пример запису та читання

**file**.createNewFile();  
FileWriter fileWriter = **new** FileWriter(**file**);  
BufferedWriter bufferedWriter = **new** BufferedWriter(fileWriter);  
**pass** = **password**;  
bufferedWriter.write(**pass**);  
bufferedWriter.newLine();  
bufferedWriter.flush();  
bufferedWriter.close();

FileReader fileReader = **new** FileReader(**file**);  
BufferedReader bufferedReader = **new** BufferedReader(fileReader);  
**while** (bufferedReader.ready())  
{  
 **pass** = bufferedReader.readLine();  
}

Робота модуля з дискреційною моделлю політики безпеки.

Програма працює таким чином, admin – входить у свій кабінет, генерую значення, тобто саму дискреційну модель політики безпеки, потім user – входить у свій кабінет, та перевіряє свої операції над об'єктами комп'ютерної системи.

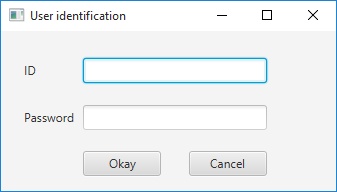


Рис. 1 – Вікно ідентифікації користувача

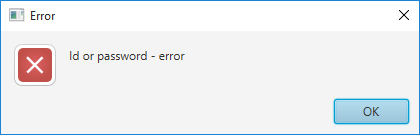


Рис. 2 – Вікно неправильно ідентифікованого користувача

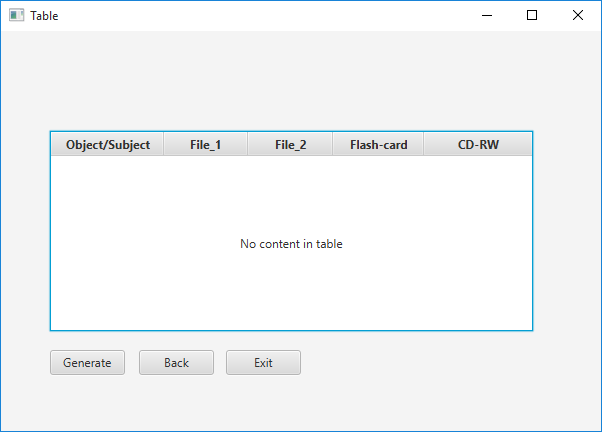


Рис. 3 – Вікно адміністратора

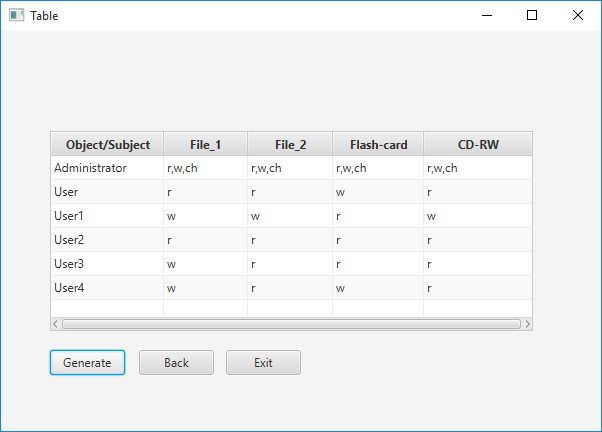


Рис. 4 – Пример виконання генерації матриці

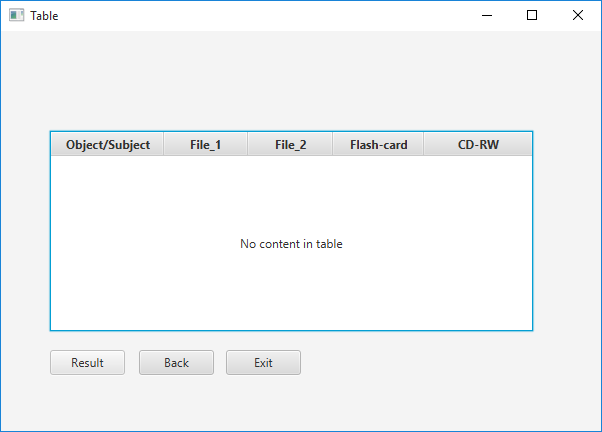


Рис. 5 – Вікно користувача

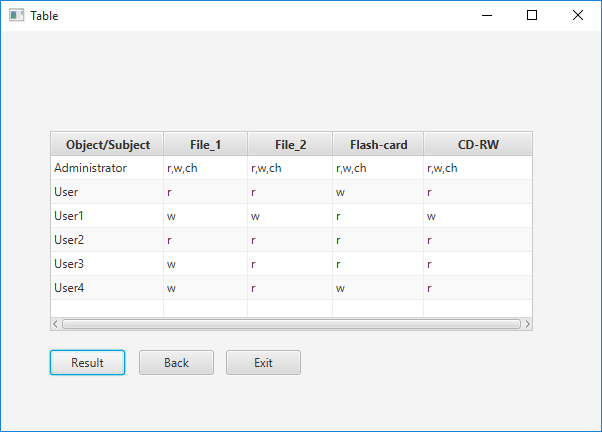


Рис. 6 – Пример огляду моделі користувачем

Лістинг програми

Код на Java

Main.java

**package** sample;  
  
**import** javafx.application.Application;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
  
**public class** Main **extends** Application {  
   
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception{  
  
 Stage primaryStage1 = **new** Stage();  
 Parent root1 = FXMLLoader.*load*(getClass().getResource(**"sample.fxml"**));  
 primaryStage1.setTitle(**"User identification"**);  
 Scene scene1 = **new** Scene(root1, 335, 160);  
 primaryStage1.setScene(scene1);  
 primaryStage1.show();  
 }  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

Controller.java

**package** sample;  
  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Node;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.stage.Stage;  
  
**import** java.io.IOException;  
  
**public class** Controller {  
  
 **private** CollectionTableModel **tableModel** = **new** CollectionTableModel();  
 **public** TextField **txtID**;  
 **public** Button **btnOkay**;  
 **public** Button **btnCancel**;  
 **public** PasswordField **txtPassword**;  
 **public** Button **btnGenerate**;  
 **public** Button **btnExit**;  
 **public** Button **btnBack**;  
 **public** TableColumn<Person, String> **columnObject**;  
 **public** TableView **tabModel**;  
 **public** TableColumn<Person, String> **columnFile1**;  
 **public** TableColumn<Person, String> **columnFile2**;  
 **public** TableColumn<Person, String> **columnFlashCard**;  
 **public** TableColumn<Person, String> **columnCD\_RW**;  
   
 **public void** accessButtonAction(ActionEvent actionEvent) **throws** IOException {  
 **if**(**txtID**.getText().toLowerCase().equals(**"admin"**) && **txtPassword**.getText().toLowerCase().equals(**"admin"**)) {  
 Stage primaryStage = **new** Stage();  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"DModel.fxml"**));  
 primaryStage.setTitle(**"Table"**);  
 Scene scene = **new** Scene(root, 600, 400);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
 **else if**(**txtID**.getText().toLowerCase().equals(**"user"**) && **txtPassword**.getText().toLowerCase().equals(**"user"**)) {  
  
 Stage primaryStage = **new** Stage();  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"DModelUser.fxml"**));  
 primaryStage.setTitle(**"Table"**);  
 Scene scene = **new** Scene(root, 600, 400);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
 **else** {  
 *showInfoDialog*(**"Error"**, **"Id or password - error"**);  
 Stage primaryStage1 = **new** Stage();  
 Parent root1 = FXMLLoader.*load*(getClass().getResource(**"sample.fxml"**));  
 primaryStage1.setTitle(**"User identification"**);  
 Scene scene1 = **new** Scene(root1, 335, 160);  
 primaryStage1.setScene(scene1);  
 primaryStage1.show();  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
  
 }  
 **public static void** showInfoDialog(String title, String text)  
 {  
 Alert alert = **new** Alert(Alert.AlertType.***ERROR***);  
 alert.setTitle(title);  
 alert.setContentText(text);  
 alert.setHeaderText(**""**);  
 alert.showAndWait();  
 }  
  
  
 **public void** cancelButtonAction(ActionEvent actionEvent) {  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
  
 **public void** generateOnAction(ActionEvent actionEvent) **throws** IOException {  
 **columnObject**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"ID"**));  
 **columnFile1**.setCellValueFactory(**new** PropertyValueFactory< Person, String>(**"File\_1"**));  
 **columnFile2**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"File\_2"**));  
 **columnFlashCard**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"Flash\_card"**));  
 **columnCD\_RW**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"CD\_RW"**));  
  
  
 **tableModel**.testGenerateData(1);  
 **tabModel**.setItems(**tableModel**.getPersonList());  
  
 }  
  
 **public void** exitOnAction(ActionEvent actionEvent) {  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
  
 **public void** backOnAction(ActionEvent actionEvent) **throws** IOException {  
 Stage primaryStage = **new** Stage();  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"sample.fxml"**));  
 primaryStage.setTitle(**"User identification"**);  
 Scene scene = **new** Scene(root, 335, 160);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 Node source = (Node) actionEvent.getSource();  
 Stage stage = (Stage) source.getScene().getWindow();  
 stage.hide();  
 }  
  
 **public void** resultOnAction(ActionEvent actionEvent) **throws** IOException {  
 **columnObject**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"ID"**));  
 **columnFile1**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"File\_1"**));  
 **columnFile2**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"File\_2"**));  
 **columnFlashCard**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"Flash\_card"**));  
 **columnCD\_RW**.setCellValueFactory(**new** PropertyValueFactory<Person, String>(**"CD\_RW"**));  
  
  
 **tableModel**.testData(0);  
 **tabModel**.setItems(**tableModel**.getPersonList());  
 }  
}

Person.java

**package** sample;  
  
**public class** Person {  
 **private** String **ID**;  
 **private** String **File\_1**;  
 **private** String **File\_2**;  
 **private** String **Flash\_card**;  
 **private** String **CD\_RW**;  
  
 **public** Person(String ID, String file\_1, String file\_2, String flash\_card, String CD\_RW) {  
 **this**.**ID** = ID;  
 **this**.**File\_1** = file\_1;  
 **this**.**File\_2** = file\_2;  
 **this**.**Flash\_card** = flash\_card;  
 **this**.**CD\_RW** = CD\_RW;  
 }  
  
 **public** String getID() {  
 **return ID**;  
 }  
  
 **public void** setID(String ID) {  
 **this**.**ID** = ID;  
 }  
  
 **public** String getFile\_1() {  
 **return File\_1**;  
 }  
  
 **public void** setFile\_1(String file\_1) {  
 **File\_1** = file\_1;  
 }  
  
 **public** String getFile\_2() {  
 **return File\_2**;  
 }  
  
 **public void** setFile\_2(String file\_2) {  
 **File\_2** = file\_2;  
 }  
  
 **public** String getFlash\_card() {  
 **return Flash\_card**;  
 }  
  
 **public void** setFlash\_card(String flash\_card) {  
 **Flash\_card** = flash\_card;  
 }  
  
 **public** String getCD\_RW() {  
 **return CD\_RW**;  
 }  
  
 **public void** setCD\_RW(String CD\_RW) {  
 **this**.**CD\_RW** = CD\_RW;  
 }  
}

CollectionTableModel.java

**package** sample;  
  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
  
**import** java.io.\*;  
**import** java.lang.reflect.Array;  
**import** java.util.ArrayList;  
  
**public class** CollectionTableModel **implements** TableModel  
{  
 **private** ObservableList<Person> **personList** = FXCollections.*observableArrayList*();  
  
 @Override  
 **public void** generateTable(Person person) {  
 **personList**.remove(person);  
 **personList**.add(person);  
 }  
  
 **public** ObservableList<Person> getPersonList() {  
 **return personList**;  
 }  
  
 String **password** = **""**;  
 String **password1** = **""**;  
 String **password2** = **""**;  
 String **password3** = **""**;  
 String **pass** = **""**;  
 String **pass1** = **""**;  
 String **pass2** = **""**;  
 String **pass3** = **""**;  
 **int i**;  
 File **file** = **new** File(**"pass.txt"**);  
 File **file1** = **new** File(**"pass1.txt"**);  
 File **file2** = **new** File(**"pass2.txt"**);  
 File **file3** = **new** File(**"pass3.txt"**);  
 **public void** testGenerateData(**int** length) **throws** IOException {  
  
  
 **for**(**i**=0; **i** < length; **i**++)  
 **password** = *randomCharacter*(**"rw"**);  
 **for**(**i**=0; **i** < length; **i**++)  
 **password1** = *randomCharacter*(**"rw"**);  
 **for**(**i**=0; **i** < length; **i**++)  
 **password2** = *randomCharacter*(**"rw"**);  
 **for**(**i**=0; **i** < length; **i**++)  
 **password3** = *randomCharacter*(**"rw"**);  
 **personList**.clear();  
 **personList**.add(**new** Person(**"Administrator"**, **"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**));  
 **personList**.add(**new** Person(**"User"**, **password**,**password1**,**password2**,**password3**));  
 **personList**.add(**new** Person(**"User1"**, **password2**,**password2**,**password3**,**password2**));  
 **personList**.add(**new** Person(**"User2"**, **password3**,**password1**,**password3**,**password1**));  
 **personList**.add(**new** Person(**"User3"**, **password2**,**password3**,**password1**,**password1**));  
 **personList**.add(**new** Person(**"User4"**, **password2**,**password**,**password2**,**password3**));  
 **file**.createNewFile();  
 FileWriter fileWriter = **new** FileWriter(**file**);  
 BufferedWriter bufferedWriter = **new** BufferedWriter(fileWriter);  
 **pass** = **password**;  
 bufferedWriter.write(**pass**);  
 bufferedWriter.newLine();  
 bufferedWriter.flush();  
 bufferedWriter.close();  
 **file1**.createNewFile();  
 fileWriter = **new** FileWriter(**file1**);  
 bufferedWriter = **new** BufferedWriter(fileWriter);  
 **pass1** = **password1**;  
 bufferedWriter.write(**pass1**);  
 bufferedWriter.newLine();  
 bufferedWriter.flush();  
 bufferedWriter.close();  
 **file2**.createNewFile();  
 fileWriter = **new** FileWriter(**file2**);  
 bufferedWriter = **new** BufferedWriter(fileWriter);  
 **pass2** = **password2**;  
 bufferedWriter.write(**pass2**);  
 bufferedWriter.newLine();  
 bufferedWriter.flush();  
 bufferedWriter.close();  
 **file3**.createNewFile();  
 fileWriter = **new** FileWriter(**file3**);  
 bufferedWriter = **new** BufferedWriter(fileWriter);  
 **pass3** = **password3**;  
 bufferedWriter.write(**pass3**);  
 bufferedWriter.newLine();  
 bufferedWriter.flush();  
 bufferedWriter.close();  
 }  
  
  
  
 **public void** testData(**int** length) **throws** IOException {  
 FileReader fileReader = **new** FileReader(**file**);  
 BufferedReader bufferedReader = **new** BufferedReader(fileReader);  
 **while** (bufferedReader.ready())  
 {  
 **pass** = bufferedReader.readLine();  
 }  
 fileReader = **new** FileReader(**file1**);  
 bufferedReader = **new** BufferedReader(fileReader);  
 **while** (bufferedReader.ready())  
 {  
 **pass1** = bufferedReader.readLine();  
 }  
 fileReader = **new** FileReader(**file2**);  
 bufferedReader = **new** BufferedReader(fileReader);  
 **while** (bufferedReader.ready())  
 {  
 **pass2** = bufferedReader.readLine();  
 }  
 fileReader = **new** FileReader(**file3**);  
 bufferedReader = **new** BufferedReader(fileReader);  
 **while** (bufferedReader.ready())  
 {  
 **pass3** = bufferedReader.readLine();  
 }  
 **personList**.clear();  
 **personList**.add(**new** Person(**"Administrator"**, **"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**,**"r,w,ch"**));  
 **personList**.add(**new** Person(**"User"**, **pass**,**pass1**,**pass2**,**pass3**));  
 **personList**.add(**new** Person(**"User1"**, **pass2**,**pass2**,**pass3**,**pass2**));  
 **personList**.add(**new** Person(**"User2"**, **pass3**,**pass1**,**pass3**,**pass1**));  
 **personList**.add(**new** Person(**"User3"**, **pass2**,**pass3**,**pass1**,**pass1**));  
 **personList**.add(**new** Person(**"User4"**, **pass2**,**pass**,**pass2**,**pass3**));  
 }  
 **public static** String randomCharacter(String characters)  
 {  
 **int** n = characters.length();  
 **int** r = (**int**) (n \* Math.*random*());  
 **return** characters.substring(r, r + 1);  
 }  
}

TableModel.java

**package** sample;  
  
**public interface** TableModel {  
 **void** generateTable(Person person);  
  
}

DModel.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.TableColumn***?>  
<?***import javafx.scene.control.TableView***?>  
<?***import javafx.scene.layout.Pane***?>*<**Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/9.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**TableView fx:id="tabModel" layoutX="49.0" layoutY="100.0" prefHeight="200.0" prefWidth="483.0"**>  
 <**columns**>  
 <**TableColumn fx:id="columnObject" prefWidth="113.0" text="Object/Subject"** />  
 <**TableColumn fx:id="columnFile1" prefWidth="84.0" text="File\_1"** />  
 <**TableColumn fx:id="columnFile2" prefWidth="85.0" text="File\_2"** />  
 <**TableColumn fx:id="columnFlashCard" prefWidth="91.0" text="Flash-card"** />  
 <**TableColumn fx:id="columnCD\_RW" prefWidth="109.0" text="CD-RW"** />  
 </**columns**>  
 </**TableView**>  
 <**Button fx:id="btnResult" layoutX="49.0" layoutY="319.0" mnemonicParsing="false" onAction="#resultOnAction" prefWidth="75.0" text="Result"** />  
 <**Button fx:id="btnExit" layoutX="225.0" layoutY="319.0" mnemonicParsing="false" onAction="#exitOnAction" prefWidth="75.0" text="Exit"** />  
 <**Button fx:id="btnBack" layoutX="138.0" layoutY="319.0" mnemonicParsing="false" onAction="#backOnAction" prefWidth="75.0" text="Back"** />  
</**Pane**>

DModelUser.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.TableColumn***?>  
<?***import javafx.scene.control.TableView***?>  
<?***import javafx.scene.layout.Pane***?>*<**Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/9.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**TableView fx:id="tabModel" layoutX="49.0" layoutY="100.0" prefHeight="200.0" prefWidth="483.0"**>  
 <**columns**>  
 <**TableColumn fx:id="columnObject" prefWidth="113.0" text="Object/Subject"** />  
 <**TableColumn fx:id="columnFile1" prefWidth="84.0" text="File\_1"** />  
 <**TableColumn fx:id="columnFile2" prefWidth="85.0" text="File\_2"** />  
 <**TableColumn fx:id="columnFlashCard" prefWidth="91.0" text="Flash-card"** />  
 <**TableColumn fx:id="columnCD\_RW" prefWidth="109.0" text="CD-RW"** />  
 </**columns**>  
 </**TableView**>  
 <**Button fx:id="btnGenerate" layoutX="49.0" layoutY="319.0" mnemonicParsing="false" onAction="#generateOnAction" prefWidth="75.0" text="Generate"** />  
 <**Button fx:id="btnExit" layoutX="225.0" layoutY="319.0" mnemonicParsing="false" onAction="#exitOnAction" prefWidth="75.0" text="Exit"** />  
 <**Button fx:id="btnBack" layoutX="138.0" layoutY="319.0" mnemonicParsing="false" onAction="#backOnAction" prefWidth="75.0" text="Back"** />  
</**Pane**>

sample.fxml

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.PasswordField***?>  
<?***import javafx.scene.control.TextField***?>  
<?***import javafx.scene.layout.ColumnConstraints***?>  
<?***import javafx.scene.layout.GridPane***?>  
<?***import javafx.scene.layout.Pane***?>  
<?***import javafx.scene.layout.RowConstraints***?>*<**GridPane alignment="center" hgap="10" vgap="10" xmlns="http://javafx.com/javafx/9.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**rowConstraints**>  
 <**RowConstraints** />  
 <**RowConstraints** />  
 </**rowConstraints**>  
 <**columnConstraints**>  
 <**ColumnConstraints** />  
 <**ColumnConstraints** />  
 </**columnConstraints**>  
 <**Pane prefHeight="172.0" prefWidth="308.0"**>  
 <**Label layoutX="14.0" layoutY="31.0" text="ID"** />  
 <**Label layoutX="14.0" layoutY="78.0" text="Password"** />  
 <**TextField fx:id="txtID" layoutX="73.0" layoutY="27.0" prefHeight="25.0" prefWidth="184.0"** />  
 <**Button fx:id="btnOkay" layoutX="73.0" layoutY="120.0" mnemonicParsing="false" onAction="#accessButtonAction" prefHeight="25.0" prefWidth="78.0" text="Okay"** />  
 <**Button fx:id="btnCancel" layoutX="179.0" layoutY="120.0" mnemonicParsing="false" onAction="#cancelButtonAction" prefWidth="78.0" text="Cancel"** />  
 <**PasswordField fx:id="txtPassword" layoutX="73.0" layoutY="74.0" prefHeight="25.0" prefWidth="184.0"** />  
 </**Pane**>  
</**GridPane**>