Бруцкая Анастасия, 251001

package com.example.encryption;

import com.example.lapa12.PluginImplementation;

import com.example.lapa12.heros.Hilichurl;

import javafx.collections.FXCollections;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import static com.example.lapa12.Main.\*;

public class Encryption implements PluginImplementation {

int p;

int k;

int x;

int y;

int g;

byte[] inputText;

ArrayList<Integer> encryptText = new ArrayList<>();

byte[] decryptText;

Random random = new Random();

Logic logic = new Logic();

File encryptFile = new File("../Encryption/encryptFile.txt");

@Override

public void doSmth() {

System.out.println("encryption");

addNewInterface();

}

public void addNewInterface(){

if(mSettings==null){

mSettings = new Menu("Settings");

MenuBar mbSettings = new MenuBar(mSettings);

controls.getChildren().add(mbSettings);

}

MenuItem miEncryption = new MenuItem("Encryption");

mSettings.getItems().add(miEncryption);

miEncryption.setOnAction(event -> {

createEncryptionWindow();

});

}

public void createEncryptionWindow(){

Stage encryptionStage = new Stage();

FileChooser fcChooseFile = new FileChooser();

Label lP = new Label(" p: ");

TextField tfP = new TextField();

HBox hbP = new HBox(5, lP, tfP);

Label lX = new Label(" x: ");

TextField tfX = new TextField();

HBox hbX = new HBox(5, lX, tfX);

Label lK = new Label(" k: ");

TextField tfK = new TextField();

HBox hbK = new HBox(5, lK, tfK);

Button btnCalculatePublicKey = new Button("Calculate public key");

Button btnEnterPrivateKey = new Button("Enter private key");

btnEnterPrivateKey.setDisable(true);

Button btnEnterSecretNumber = new Button("Enter secret number");

btnEnterSecretNumber.setDisable(true);

Button btnEncrypt = new Button("Encrypt");

btnEncrypt.setDisable(true);

Button btnDecrypt = new Button("Decrypt");

btnDecrypt.setDisable(true);

HBox buttons = new HBox(10, btnEncrypt, btnDecrypt);

buttons.setAlignment(Pos.BOTTOM\_CENTER);

Label lPrimitiveRoots = new Label(" primitive roots: ");

ComboBox<Integer> cbPrimitiveRoots = new ComboBox<>();

cbPrimitiveRoots.setDisable(true);

HBox hbPrimitiveRoots = new HBox(5, lPrimitiveRoots, cbPrimitiveRoots);

btnCalculatePublicKey.setOnAction(event -> {

int tempP;

try {

tempP = Integer.parseInt(tfP.getText());

if (logic.isPrimeNumber(tempP)) {

p = tempP;

cbPrimitiveRoots.setItems(FXCollections.observableList(logic.findAllPrimitiveRoots(p, logic.findAllPrimeDivisors(p - 1))));

cbPrimitiveRoots.setValue(cbPrimitiveRoots.getItems().get(0));

cbPrimitiveRoots.setDisable(false);

btnEnterPrivateKey.setDisable(false);

btnEnterSecretNumber.setDisable(false);

}

else {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("p is not prime number");

alert.showAndWait();

}

} catch (NumberFormatException | IndexOutOfBoundsException e ) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("invalid p");

alert.showAndWait();

}

});

btnEnterPrivateKey.setOnAction(event -> {

int tempX;

try {

tempX = Integer.parseInt(tfX.getText());

if (tempX >= p - 1 || tempX == 1) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("invalid x");

alert.showAndWait();

} else {

x = tempX;

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("x is set");

alert.showAndWait();

}

} catch (NumberFormatException e) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("invalid x");

alert.showAndWait();

}

});

btnEnterSecretNumber.setOnAction(event -> {

int tempK;

try {

tempK = Integer.parseInt(tfK.getText());

if ((tempK>0 && tempK<p) && !logic.isRelativelyPrime(tempK, p-1)) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("invalid k");

alert.showAndWait();

} else {

btnEncrypt.setDisable(false);

btnDecrypt.setDisable(false);

k = tempK;

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("k is set");

alert.showAndWait();

}

} catch (NumberFormatException e) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("invalid k");

alert.showAndWait();

}

});

btnEncrypt.setOnAction(event -> {

try {

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);

objectOutputStream.writeObject(hilichurls.hilichurls);

inputText = byteArrayOutputStream.toByteArray();

objectOutputStream.close();

byteArrayOutputStream.close();

} catch (IOException e){

e.printStackTrace();

}

g = cbPrimitiveRoots.getValue();

y = logic.powerMod(g, x, p);

int randomNumber=0;

for (byte value : inputText) {

int a = logic.powerMod(g, k + randomNumber, p);

int b = logic.powerModWithMultiply(y, k+randomNumber, value, p);

encryptText.add(a);

encryptText.add(b);

do {

randomNumber = random.nextInt(100);

} while ((k+randomNumber>0 && k+randomNumber<p) && !logic.isRelativelyPrime(k+randomNumber,p-1 ));

}

try {

encryptFile = fcChooseFile.showSaveDialog(encryptionStage);

FileOutputStream fileOutputStream = new FileOutputStream(encryptFile);

for (Integer integer :

encryptText) {

fileOutputStream.write(integer);

}

} catch (IOException e) {

throw new RuntimeException(e);

}

});

btnDecrypt.setOnAction(event -> {

encryptText= new ArrayList<>();

try{

encryptFile=fcChooseFile.showOpenDialog(encryptionStage);

FileInputStream fileInputStream = new FileInputStream(encryptFile);

int temp = fileInputStream.read();

while (temp>=0){

encryptText.add(temp);

temp = fileInputStream.read();

}

fileInputStream.close();

} catch (IOException e){

e.printStackTrace();

}

decryptText = new byte[encryptText.size()/2];

for (int i = 0; i < encryptText.size(); i+=2) {

byte temp = (byte) logic.powerModWithMultiply(encryptText.get(i),-x, encryptText.get(i+1),p);

if (temp<0) {

decryptText[i / 2] = (byte) (temp-1);

} else {

decryptText[i / 2] = temp;

}

}

try {

ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decryptText);

ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);

hilichurls.hilichurls.addAll((List<Hilichurl>) objectInputStream.readObject());

objectInputStream.close();

byteArrayInputStream.close();

for (Hilichurl character :

hilichurls.hilichurls) {

movement.keyPress(scene, heroes, character, character.getX(),

character.getY());

movement.stopTimer();

}

scene.getRoot().requestFocus();

} catch (IOException | ClassNotFoundException e) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("The file could not be decrypted, check that public and private keys are entered" +

" correctly");

alert.showAndWait();

}

});

VBox root = new VBox(10, hbP, btnCalculatePublicKey, hbPrimitiveRoots, hbX, btnEnterPrivateKey,

hbK,btnEnterSecretNumber, buttons);

Scene optionsScene = new Scene(root, 300, 350);

encryptionStage.setScene(optionsScene);

encryptionStage.setTitle("Encryption");

encryptionStage.show();

}

}

package com.example.encryption;

import java.math.BigInteger;

import java.util.ArrayList;

public class Logic {

public boolean isPrimeNumber(int number) {

BigInteger bigInteger = BigInteger.valueOf(number);

return (bigInteger.isProbablePrime((int) Math.log(number)))&&(number!=1);

}

public boolean isRelativelyPrime(int num1, int num2){

BigInteger bigInteger1 = BigInteger.valueOf(num1);

BigInteger bigInteger2 = BigInteger.valueOf(num2);

return bigInteger1.gcd(bigInteger2).intValue() == 1;

}

public ArrayList<Integer> findAllPrimeDivisors(int p) {

int tempP = p;

ArrayList<Integer> primeDivisors = new ArrayList<>();

for (int i = 2; i \* i <= tempP; i++) {

while (tempP % i == 0) {

primeDivisors.add(i);

tempP /= i;

}

}

if (tempP > 1) {

primeDivisors.add(tempP);

}

return primeDivisors;

}

public int powerMod(int base, int exponent, int modulus) {

BigInteger bigIntegerBase = BigInteger.valueOf(base);

BigInteger bigIntegerExponent = BigInteger.valueOf(exponent);

BigInteger bigIntegerModulus = BigInteger.valueOf(modulus);

return bigIntegerBase.modPow(bigIntegerExponent, bigIntegerModulus).intValue();

}

public int powerModWithMultiply(int base, int exponent, int multiply, int modulus) {

BigInteger bigIntegerBase = BigInteger.valueOf(base);

BigInteger bigIntegerMultiply = BigInteger.valueOf(multiply);

BigInteger bigIntegerModulus = BigInteger.valueOf(modulus);

BigInteger power;

if (exponent>=0) {

power = bigIntegerBase.pow(exponent);

} else {

exponent = Math.abs(exponent);

power = bigIntegerBase.pow(modulus - 1 -exponent);

}

return power.multiply(bigIntegerMultiply).mod(bigIntegerModulus).intValue();

}

public ArrayList<Integer> findAllPrimitiveRoots(int p, ArrayList<Integer> primeDivisors) {

ArrayList<Integer> primitiveRoots = new ArrayList<>();

for (int i = 2; i < p; i++) {

int g = i;

boolean flag = true;

for (Integer primeDivisor : primeDivisors) {

if (powerMod(g,(p-1)/primeDivisor, p)==1){

flag = false;

}

}

if (flag){

primitiveRoots.add(g);

}

}

return primitiveRoots;

}

}

package com.example.transformingxmlintojson;

import com.example.lapa12.PluginImplementation;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.dataformat.xml.XmlMapper;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import org.json.JSONObject;

import org.json.XML;

import java.io.\*;

import static com.example.lapa12.Main.controls;

import static com.example.lapa12.Main.mSettings;

public class TransformingXMLintoJSON implements PluginImplementation {

File openFile;

File transformedFile;

String xmlString;

String input="";

String output="";

String jsonString;

@Override

public void doSmth() {

System.out.println("transforming");

addNewInterface();

}

public void addNewInterface(){

if(mSettings==null){

mSettings = new Menu("Settings");

MenuBar mbSettings = new MenuBar(mSettings);

controls.getChildren().add(mbSettings);

}

MenuItem miEncryption = new MenuItem("Transforming XML into JSON");

mSettings.getItems().add(miEncryption);

miEncryption.setOnAction(event -> {

createTransformingWindow();

});

}

public void createTransformingWindow(){

Stage transformingStage = new Stage();

FileChooser fcChooseFile = new FileChooser();

Button btnSaveFile = new Button("Save file");

Button btnOpenFile = new Button("Choose file");

Button btnTransformFile = new Button("Transform file");

VBox vbButtons = new VBox(5, btnOpenFile, btnTransformFile);

RadioButton rbFromXMLintoJSON = new RadioButton("from XML into JSON");

rbFromXMLintoJSON.setSelected(true);

RadioButton rbFromJSONintoXML = new RadioButton("from JSON into XML");

VBox vbRadioButtons = new VBox(5, rbFromXMLintoJSON, rbFromJSONintoXML);

HBox hbControls = new HBox(10, vbButtons, vbRadioButtons);

Label lOpenFile = new Label("Open file: ");

TextArea taOpenFile = new TextArea();

taOpenFile.setWrapText(true);

HBox hbOpenFile = new HBox(10, lOpenFile, taOpenFile);

Label lTransformedFile = new Label("Transformed file: ");

TextArea taTransformedFile = new TextArea();

taTransformedFile.setWrapText(true);

HBox hbTransformedFile = new HBox(10, lTransformedFile, taTransformedFile);

rbFromJSONintoXML.setOnAction(event -> {

if (rbFromXMLintoJSON.isSelected()){

rbFromXMLintoJSON.setSelected(false);

}

if (!rbFromJSONintoXML.isSelected()){

rbFromJSONintoXML.setSelected(true);

}

});

rbFromXMLintoJSON.setOnAction(event -> {

if (rbFromJSONintoXML.isSelected()){

rbFromJSONintoXML.setSelected(false);

}

if (!rbFromXMLintoJSON.isSelected()){

rbFromXMLintoJSON.setSelected(true);

}

});

btnOpenFile.setOnAction(event -> {

try {

openFile = fcChooseFile.showOpenDialog(transformingStage);

BufferedReader bufferedReader= new BufferedReader(new FileReader(openFile));

String line = bufferedReader.readLine();

while (line!=null){

input+=line;

line = bufferedReader.readLine();

}

taOpenFile.setText(input);

bufferedReader.close();

} catch (IOException e) {

throw new RuntimeException(e);

}

});

btnTransformFile.setOnAction(event -> {

if (rbFromXMLintoJSON.isSelected()){

xmlString=input;

JSONObject jsonObject = XML.toJSONObject(xmlString);

jsonString = jsonObject.toString();

taTransformedFile.setText(jsonString);

} else {

try {

jsonString = input;

ObjectMapper objectMapper = new ObjectMapper();

JsonNode jsonNode = objectMapper.readTree(jsonString);

xmlString=new XmlMapper().writeValueAsString(jsonNode);

taTransformedFile.setText(xmlString);

} catch (JsonProcessingException e) {

System.out.println("wrong data format");

}

}

output=taTransformedFile.getText();

});

btnSaveFile.setOnAction(event -> {

try {

transformedFile = fcChooseFile.showSaveDialog(transformingStage);

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(transformedFile));

bufferedWriter.write(output);

bufferedWriter.close();

} catch (IOException e){

e.printStackTrace();

}

});

VBox root = new VBox(10, hbControls, hbOpenFile, hbTransformedFile, btnSaveFile);

Scene optionsScene = new Scene(root, 600, 500);

transformingStage.setScene(optionsScene);

transformingStage.setTitle("Transforming");

transformingStage.show();

}

}