import java.awt.\*;

import java.awt.event.\*;

import java.security.SecureRandom;

import java.util.ArrayList;

public class PasswordGeneratorAWT extends Frame implements ActionListener {

Label labelLength, labelOptions, labelResult;

TextField textLength, textResult;

Checkbox checkboxUppercase, checkboxLowercase, checkboxNumbers, checkboxSymbols;

Button generateButton, closeButton;

SecureRandom random = new SecureRandom();

public PasswordGeneratorAWT() {

// Frame settings

setTitle("Password Generator");

setSize(500, 300);

setLayout(new FlowLayout());

setVisible(true);

// Input fields and labels

labelLength = new Label("Password Length:");

textLength = new TextField(10);

labelOptions = new Label("Include:");

checkboxUppercase = new Checkbox("Uppercase Letters");

checkboxLowercase = new Checkbox("Lowercase Letters");

checkboxNumbers = new Checkbox("Numbers");

checkboxSymbols = new Checkbox("Symbols");

labelResult = new Label("Generated Password:");

textResult = new TextField(30);

textResult.setEditable(false);

// Buttons

generateButton = new Button("Generate");

closeButton = new Button("Close");

// Adding components to Frame

add(labelLength);

add(textLength);

add(labelOptions);

add(checkboxUppercase);

add(checkboxLowercase);

add(checkboxNumbers);

add(checkboxSymbols);

add(labelResult);

add(textResult);

add(generateButton);

add(closeButton);

// Action Listeners

generateButton.addActionListener(this);

closeButton.addActionListener(this);

// Closing the window

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

dispose();

}

});

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == generateButton) {

try {

// Step 1: Get length input

int length = Integer.parseInt(textLength.getText());

// Step 2: Validate length

if (length < 8) {

textResult.setText("Length must be at least 8.");

return;

}

// Step 3: Get character pool options

boolean[] options = {

checkboxUppercase.getState(),

checkboxLowercase.getState(),

checkboxNumbers.getState(),

checkboxSymbols.getState()

};

ArrayList<Character> pool = createCharacterPool(options);

if (pool.isEmpty()) {

textResult.setText("Select at least one character type.");

return;

}

// Step 4: Generate password

String password = generatePassword(length, pool);

// Step 5: Display password

textResult.setText(password);

} catch (NumberFormatException ex) {

textResult.setText("Enter a valid number for length.");

}

} else if (e.getSource() == closeButton) {

dispose();

}

}

// Module 1: Character Pool Creation

public ArrayList<Character> createCharacterPool(boolean[] options) {

ArrayList<Character> pool = new ArrayList<>();

if (options[0]) {

for (char c = 'A'; c <= 'Z'; c++) pool.add(c);

}

if (options[1]) {

for (char c = 'a'; c <= 'z'; c++) pool.add(c);

}

if (options[2]) {

for (char c = '0'; c <= '9'; c++) pool.add(c);

}

if (options[3]) {

String symbols = "!@#$%^&\*()-\_=+[]{}|;:'\",.<>?/`~";

for (char c : symbols.toCharArray()) pool.add(c);

}

return pool;

}

// Module 2: Password Generator

public String generatePassword(int length, ArrayList<Character> pool) {

StringBuilder password = new StringBuilder(length);

for (int i = 0; i < length; i++) {

password.append(pool.get(random.nextInt(pool.size())));

}

return password.toString();

}

public static void main(String[] args) {

new PasswordGeneratorAWT();

}

}