CS 1073

FR04B

Assignment 10

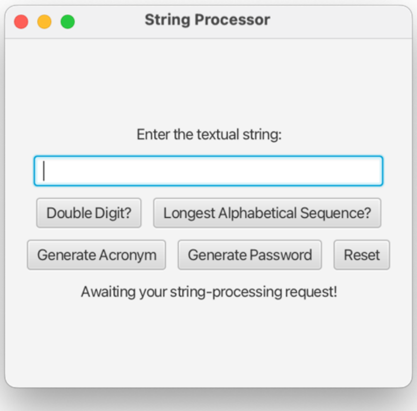
Ebrahim Arefi

3621326

1 Processing Strings: (the JavaFX GUI):

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.geometry.Pos;
import javafx.event.ActionEvent;

/**
 * StringProcessor GUI:
 *
 * Allow the user to type a textual string.
 * The program will check for Duplicates, and Longest sequence.
 * Also Generating Acronyms and Password.
 *
 * @author Ebrahim Arefi, 3621326
 */

public class StringProcessor extends Application {

    private TextField enter;
    private Label output;

    public void start(Stage primaryStage) {
        primaryStage.setTitle("String Processor");

        enter = new TextField();
        enter.setPrefWidth(300);

        Label enterLabel = new Label("Enter the textual string:");
        output = new Label("Awaiting your string-processing request!");

        Button resetButton = new Button("Reset");
        resetButton.setOnAction(this::processReset);

        Button doubleButton = new Button("Double Digit?");
        doubleButton.setOnAction(this::processDouble);

        Button longestButton = new Button("Longest Alphabetical Sequence?");
        longestButton.setOnAction(this::processLongest);

        Button acronymButton = new Button("Generate Acronym");
        acronymButton.setOnAction(this::processAcronym);

        Button passwordButton = new Button("Generate Password");
        passwordButton.setOnAction(this::processPassword);

        FlowPane pane = new FlowPane(20, 20,
                enterLabel, enter,
                doubleButton, longestButton,
                acronymButton, passwordButton,
                resetButton,
                output);

        pane.setAlignment(Pos.CENTER);
        pane.setVgap(10);
        pane.setHgap(10);
        Scene scene = new Scene(pane, 350, 300);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void processDouble(ActionEvent e) {
        String input = enter.getText();
        if (hasDoubleDigit(input)) {
            output.setText("Yes, there are adjacent double digits.");
        } else {
            output.setText("No, there are no adjacent double digits.");
        }
    }

    private void processAcronym(ActionEvent e) {
        String input = enter.getText();
        output.setText("Generated Acronym: " + generateAcronym(input));
    }

    private void processPassword(ActionEvent e) {
        String input = enter.getText();
        output.setText("Generated Password: " + generatePassword(input));
    }

    private void processReset(ActionEvent e) {
        enter.setText("");
        output.setText("Awaiting your string-processing request!");
    }

    private void processLongest(ActionEvent e) {
        String input = enter.getText();
        int result = longestSequence(input);
        output.setText("Longest alphabetical sequence: " + result);
    }

    public boolean hasDoubleDigit(String text) {

        for (int i = 0; i < text.length() - 1; i++) {
            char ch1 = text.charAt(i);
            char ch2 = text.charAt(i + 1);
            if (ch1 >= '0' && ch1 <= '9' && ch2 >= '0') {
                if (ch2 <= '9' && ch1 == ch2) {
                    return true;
                }
            }
        }
        return false;
    }

    public static int longestSequence(String text) {
        int longest = 0;
        String textLower = text.toLowerCase();

        for (int i = 0; i < textLower.length(); i++) {
            char ch = textLower.charAt(i);

            if (ch >= 'a' && ch <= 'z') {
                int current = 1;
                char last = ch;

                for (int j = i + 1; j < textLower.length(); j++) {
                    char next = textLower.charAt(j);
                if (next >= 'a' && next <= 'z') {
                    if (next >= last) {
                        current++;
                        last = next;
                    }
                }
            }
        }
    }
    return longest;
}

public static String generateAcronym(String text) {

    String acronym = "";
    String tokenDigit = "";

    for (int i = 0; i < text.length(); i++) {
        char ch = text.charAt(i);

        if (ch == ' ') {
            if (tokenDigit.length() > 0) {
                char first = tokenDigit.charAt(0);

                if (first >= 'A' && first <= 'Z') {
                    acronym += first;
                } else if (first >= '0' && first <= '9') {
                    acronym += tokenDigit;
                }
            }
            tokenDigit = "";

        } else {
            tokenDigit += ch;
        }
    }

    if (tokenDigit.length() > 0) {
        char first = tokenDigit.charAt(0);
        if (first >= 'A' && first <= 'Z') {
            acronym = acronym + first;
        } else if (first >= '0' && first <= '9') {
            acronym += tokenDigit;
        }
    }

    if (acronym.length() < 2) {
        return "Unable to generate an acronym from this input";
    }
    return acronym;
}

public static String generatePassword(String text) {
    String password = "";
    String token = "";
    int tokenCount = 0;

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

        char ch = text.charAt(i);

        if (ch != ' ') {
            token = token + ch;
        } else {
                if (token.length() > 3) {

                    tokenCount++;

                    if (tokenCount % 2 == 1) {

                        int len = token.length();

                        char c1 = token.charAt(len - 2);
                        char c2 = token.charAt(len - 1);

                        if (c1 >= 'A' && c1 <= 'Z')
                            c1 = (char) (c1 + 32);
                        if (c2 >= 'A' && c2 <= 'Z')
                            c2 = (char) (c2 + 32);

                        password = password + c1 + c2;
                    }

                }
                token = "";
            }
        }
        if (token.length() > 3) {
            password += token.charAt(0);
        }
        return password;
    }

}
```

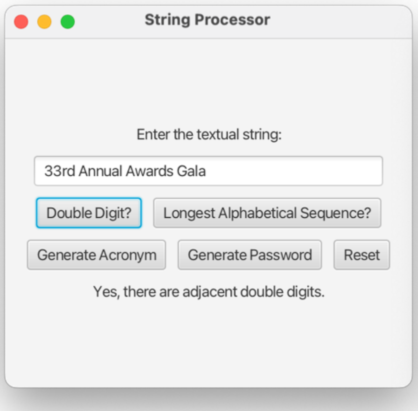
2) The sample output for Question I:


String Processor

Enter the textual string:

Awaiting your string-processing request!

Before

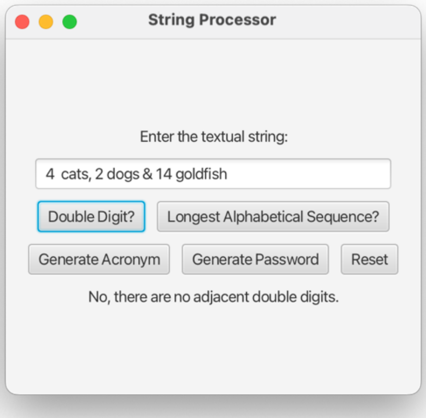

String Processor

Enter the textual string:

33rd Annual Awards Gala

Yes, there are adjacent double digits.

Double Digit test 1

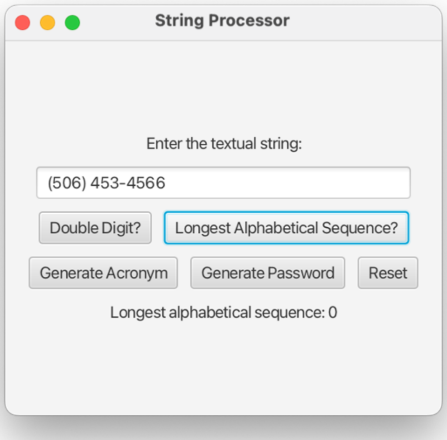

String Processor

Enter the textual string:

4 cats, 2 dogs & 14 goldfish

No, there are no adjacent double digits.

Double Digit test 2

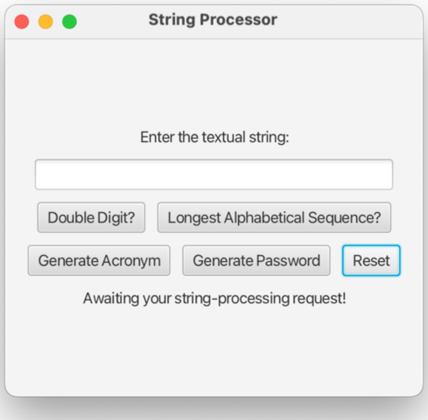

String Processor

Enter the textual string:

(506) 453-4566

Longest alphabetical sequence: 0

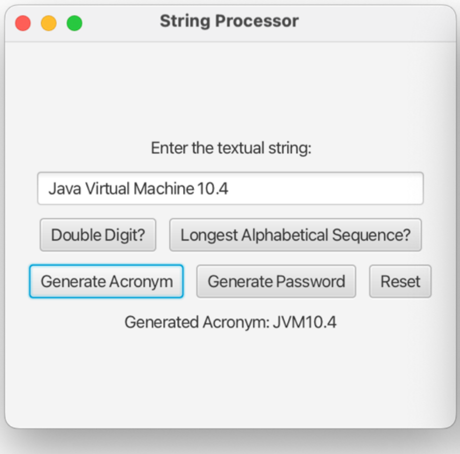
Longest test


String Processor

Enter the textual string:

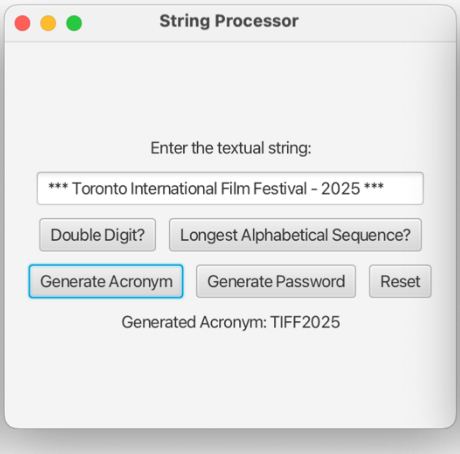
Awaiting your string-processing request!

Reset Test



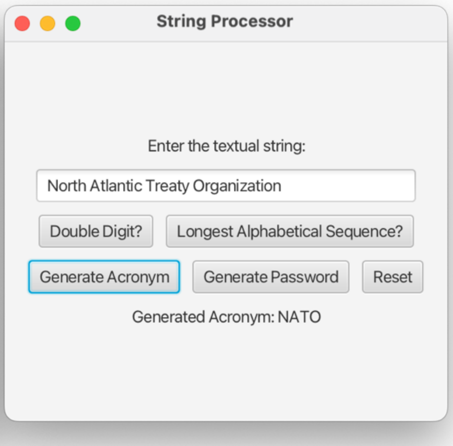
Java Virtual Machine 10.4

Generated Acronym: JVM10.4



*** Toronto International Film Festival - 2025 ***

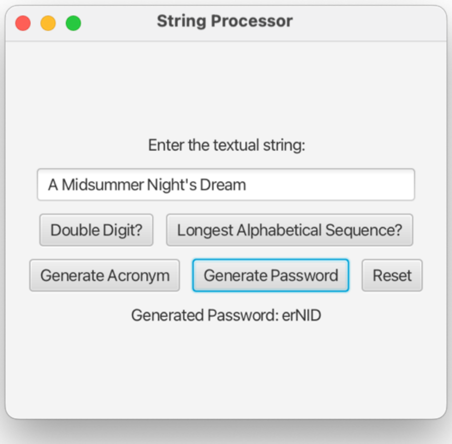
Generated Acronym: TIFF2025



North Atlantic Treaty Organization

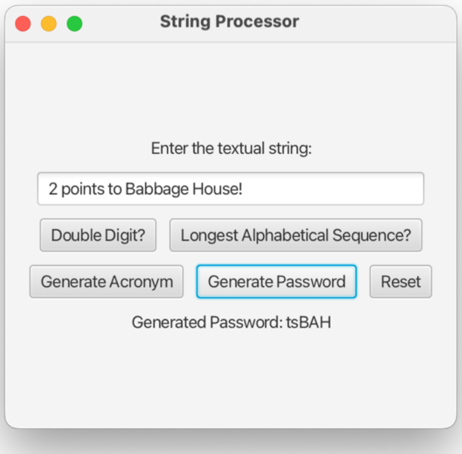
Generated Acronym: NATO

Acronyms tests



A Midsummer Night's Dream

Generated Password: erNID



2 points to Babbage House!

Generated Password: tsBAH

Passwords tests

2) Name Analyzer:

```java
/**
 * NameAnalyzer Application.
 * Analyzes user's inputs and gathers data.
 *
 * @author Ebrahim Arefi
 */

import java.util.Scanner;

public class NameAnalyzer {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();

        String surname = "";
        String givenName = "";
        String currentNick = "";

        boolean readingSurname = true;
        boolean readingGivenName = false;
        boolean readingNickName = false;

        int nickNameCount = 0;
        String longestNickName = "";
        int longestLength = 0;

        int commaCount = 0;

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

            char c = input.charAt(i);

            if (c == ',') {
                commaCount++;

                if (commaCount == 1) {
                    readingSurname = false;
                    readingGivenName = true;
                    readingNickName = false;
                }

                else if (commaCount == 2) {
                    readingSurname = false;
                    readingGivenName = false;
                    readingNickName = true;
                }

                else if (commaCount >= 3) {

                    int letters = 0;
                    for (int j = 0; j < currentNick.length(); j++) {

                        char d = currentNick.charAt(j);
                boolean upper = (d >= 'A' && d <= 'Z');
                boolean lower = (d >= 'a' && d <= 'z');

                if (upper || lower) {
                    letters++;
                }
            }

            if (letters > longestLength) {
                longestLength = letters;
                longestNickName = currentNick;
            }

            nicknameCount++;
            currentNick = "";
        }
    } else {

        if (readingSurname) {
            surname += c;
        } else if (readingGivenName) {
            givenName += c;
        } else if (readingNickName) {
            currentNick += c;
        }
    }
}

if (readingNickName && currentNick.length() > 0) {

    int letters = 0;
    for (int j = 0; j < currentNick.length(); j++) {
        char d = currentNick.charAt(j);
        boolean upper = (d >= 'A' && d <= 'Z');
        boolean lower = (d >= 'a' && d <= 'z');
        if (upper || lower) {
            letters++;
        }
    }

    if (letters > longestLength) {
        longestLength = letters;
        longestNickName = currentNick;
    }

    nicknameCount++;
}

boolean multiWord = false;
for (int i = 0; i < givenName.length(); i++) {
    char c = givenName.charAt(i);
    if (c == '-') {
        multiWord = true;
    }
}

char firstSurLetter = ' ';
for (int i = 0; i < surname.length(); i++) {
    char ch = surname.charAt(i);
```

```java
            boolean upper = (ch >= 'A' && ch <= 'Z');
            boolean lower = (ch >= 'a' && ch <= 'z');

            if (upper || lower) {

                if (upper) {
                    ch = (char) (ch + 32);
                }
                firstSurLetter = ch;
                break;
            }
        }

        char lastGivenLetter = ' ';
        for (int i = givenName.length() - 1; i >= 0; i--) {
            char ch = givenName.charAt(i);

            boolean upper = (ch >= 'A' && ch <= 'Z');
            boolean lower = (ch >= 'a' && ch <= 'z');

            if (upper || lower) {

                if (upper) {
                    ch = (char) (ch + 32);
                }
                lastGivenLetter = ch;
                break;
            }
        }

        boolean letterFlow = false;
        if (lastGivenLetter == firstSurLetter) {
            letterFlow = true;
        }

        System.out.println("-------------------------");
        System.out.println("Output:\n");

        String full = givenName.toUpperCase() + " " + surname.toUpperCase();

        if (nickNameCount > 0) {
            full += " (" + longestNickName + ")";
        }

        System.out.println(full);
        System.out.println("Multi-word Given Name: " + multiWord);
        System.out.println("Letter Flow: " + letterFlow);
        System.out.println("Number of Nicknames: " + nickNameCount);

    }
}
```

Output

```
ebi@iMac as10 % javac NameAnalyzer.java
ebi@iMac as10 % java NameAnalyzer
Zachary,Samantha,Sam,Sam I Am,Sammy,Zach
------
Output:

SAMANTHA ZACHARY (Sam I Am)
Multi-word Given Name: false
Letter Flow: false
Number of Nicknames: 4
ebi@iMac as10 % java NameAnalyzer
Ayotunde,Yuuta,ayo,tun,yu
------
Output:

YUUTA AYOTUNDE (ayo)
Multi-word Given Name: false
Letter Flow: true
Number of Nicknames: 3
ebi@iMac as10 % java NameAnalyzer
Richardson Morgan,John-Paul
------
Output:

JOHN-PAUL RICHARDSON MORGAN
Multi-word Given Name: true
Letter Flow: false
Number of Nicknames: 0
ebi@iMac as10 % java NameAnalyzer
zAya-Arseneau,micheaLa,Zay,Mikey
------
Output:

MICHEALA ZAYA-ARSENEAU (Mikey)
Multi-word Given Name: false
Letter Flow: false
Number of Nicknames: 2
ebi@iMac as10 % java NameAnalyzer
MAURY MacDonald,Nathaniel Liam,Nate,Nathan,Nat Mac
------
Output:

NATHANIEL LIAM MAURY MACDONALD (Nathan)
Multi-word Given Name: true
Letter Flow: true
Number of Nicknames: 3
ebi@iMac as10 % java NameAnalyzer
WEBBER,Bettina Alice,Tina
------
Output:

BETTINA ALICE WEBBER (Tina)
Multi-word Given Name: true
Letter Flow: false
Number of Nicknames: 1
ebi@iMac as10 %
```