Project 4 Code/Output

Christopher Ringer, Milo Baker-Durante, Sol Ben-Ishay

We affirm that we have carried out the attached academic endeavours with full academic honesty by writing our names above

Python Project

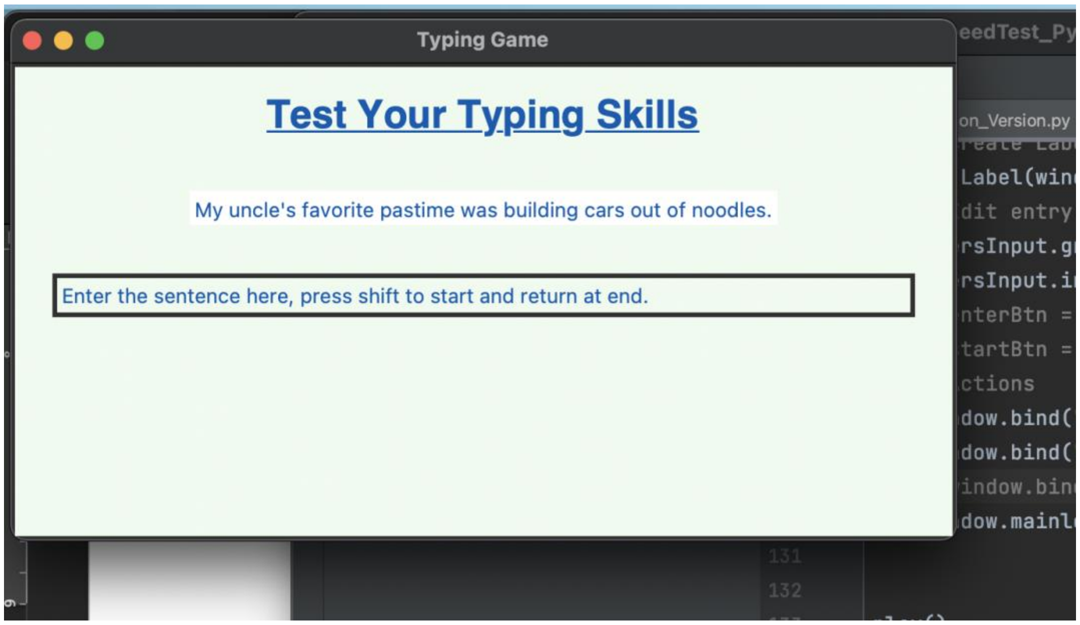
Code:

Typing_Game_Python_Version.py

```
# We affirm that we have carried out the attached academic endeavours with
full academic honesty
# Sol Ben-Ishay, Christopher Ringer, Milo Baker-Durante

# Use random for sentence get, use tkinter for gui, use timeit for stopwatch
import random
import tkinter as tk
from tkinter import messagebox
from timeit import default_timer as timer

# These global variables need to be accessed by various functions so this
initializes them
start = 0
copy_this = ""
what_typed = ""
time_took = 0
count = 0
accuracy = 0
end = 0
window = tk.Tk()
screen = tk.Canvas(window, width=600, height=300)
usersInput = tk.Entry(window, width=60, background='#EFFBEF', fg='#0059b3')
# Create Enter button
# enterBtn = tk.Button(window, text="Enter")
# # Create Start button

def reset_game():
    global window
    global start
    global copy_this
    global usersInput
    global end
    global count
    global time_took
    global what_typed
    global accuracy
    what_typed = ""
    count = 0
    accuracy = 0
    start = 0
    end = 0
    time_took = 0
    copy_this = ""
    usersInput.delete(0, "end")
    window.focus()



def get_sentences():
    # Will get a sentence for user to copy
    f = open('sentences.txt', "r")
    list_sentences = []
    for line in f:
        stripped_line = line.strip()
        list_sentences.append(stripped_line)
    f.close()
    sentence = random.choice(list_sentences)
    return sentence  # outputs a random sentence from the file of sentences


def onKey(*args):
    # When shift is pressed, timer starts
    global start
    global usersInput
    usersInput.focus()
    usersInput.delete(0, "end")
    start = timer()


def onReturn(*args):
    # When return is pressed stops timer and calculates results, updates
display to show results after
    global start
    global window
    global screen
    global copy_this
    global end
    global time_took
    global what_typed
    global count
    end = timer()
    time_took = round(end - start)
    what_typed = usersInput.get();
    # if the user enters an extra char in middle need to make two lens equal,
    # sacrifices accuracy i guess
    if len(what_typed) > len(copy_this):
        difference = len(what_typed) - len(copy_this)
        addition = what_typed[-difference:]
        copy_this = copy_this + addition
    # Checks accuracy of two sentences
    for i, c in enumerate(what_typed):
        if copy_this[i] == c:
            count += 1
    accuracy = round((count / len(copy_this)) * 100)
    # Wpm based on avg word len of 5 char
    wpm = round((len(copy_this) / 5) / (time_took / 60))



    # Create results string
    result_string = "Total time: " + str(time_took) + "sec / WPM: " +
str(wpm) + "/ Accuracy: " + str(accuracy) + "%"
    # Display results
    messagebox.showinfo("Results", result_string)
    # Continue playing until quit
    reset_game()
    play()

def play():
    # Outputs the main window for the game
    global window
    global screen
    global usersInput
    global copy_this
    global enterBtn
    global startBtn
    # Title
    window.title("Typing Game")
    screen.configure(background='#EFFBEF')
    screen.grid(columnspan=7, rowspan=7)
    # Main logo
    tk.Label(window, text="Test Your Typing Skills", font=("Helvetica", 25,
'bold', 'underline'), fg='#0059b3',
             background='#EFFBEF').grid(columnspan=7, column=0, row=0)
    # Gets a sentence to show
    copy_this = get_sentences()
    # Create Label for Sentence
    tk.Label(window, text=copy_this, background='white',
fg='#0059b3').grid(columnspan=7, column=0, row=1)
    # Edit entry box for user input
    usersInput.grid(columnspan=7, column=0, row=2)
    usersInput.insert(0, "Enter the sentence here, press shift to start and
return at end.")
    # Create enterBtn
    # enterBtn = tk.Button(window, text="Enter",command=lambda:
onReturn()).grid(columnspan=3, column=2, row=4)
    # Create startBtn
    # startBtn = tk.Button(window, text="Start",command=lambda:
onKey()).grid(columnspan=3, column=2, row=5)
    # Actions
    window.bind("<Return>", onReturn)
    window.bind("<Shift_L>", onKey)
    window.bind("<Shift_R>", onKey)
    # Run window
    window.mainloop()

# Play Game
play()
```

Output(Screenshots):

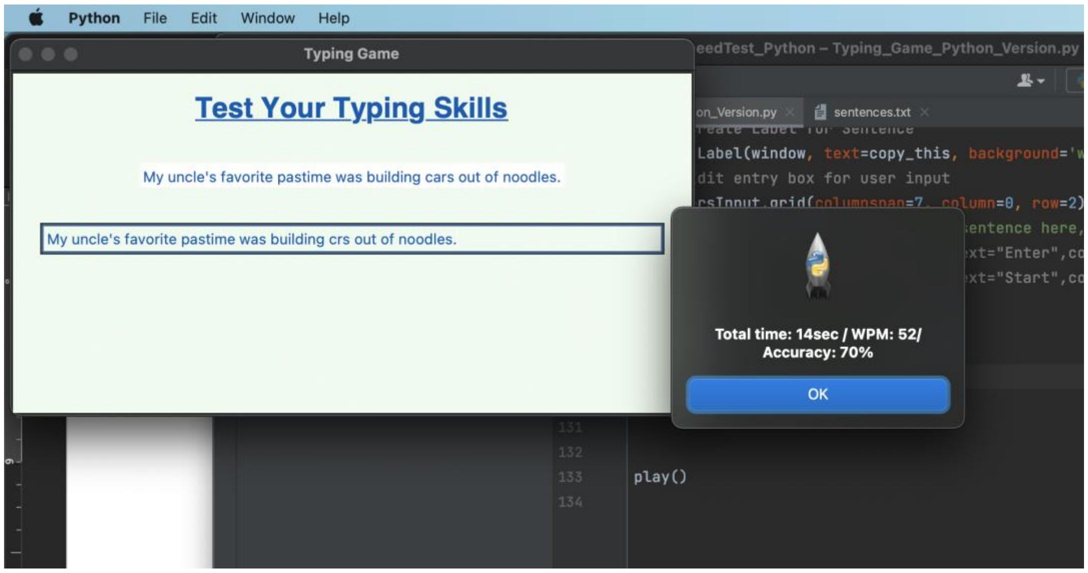
First Round:



My uncle's favorite pastime was building cars out of noodles.

Enter the sentence here, press shift to start and return at end.

Results Window:

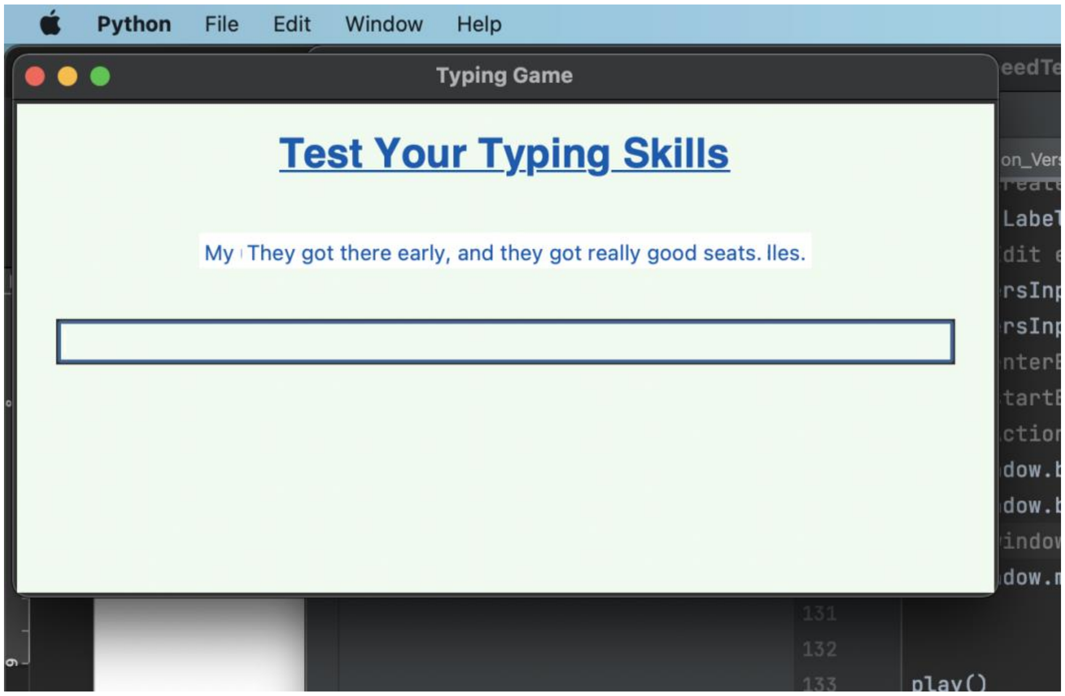


My uncle's favorite pastime was building cars out of noodles.

My uncle's favorite pastime was building crs out of noodles.

Total time: 14sec / WPM: 52/
Accuracy: 70%

Next Round:



Test Your Typing Skills

My They got there early, and they got really good seats. lles.

Keeps repeating the above until exited

Java Project

Code:

TypingSpeedTest.java

```
package TypingSpeedTest;

import javax.swing.*;

import java.awt.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;
```

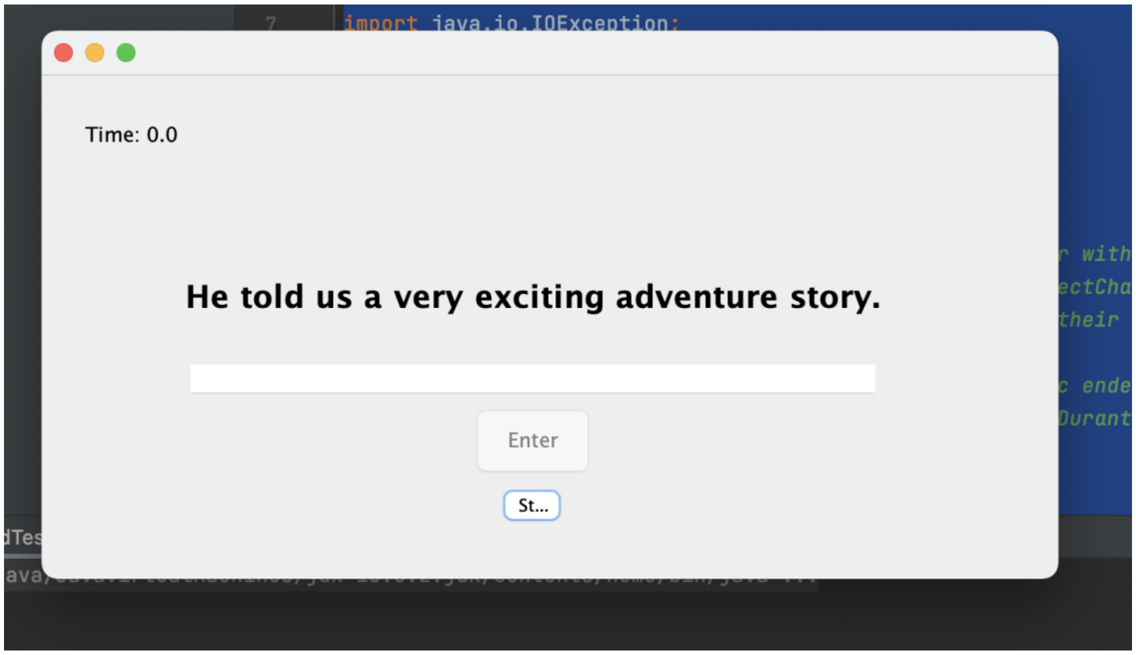
```
/**
 * This is a TypingSpeedTest program that prompts the user with a line of
text and calculates the time taken, word per
 * minute (WPM), and accuracy (as a percentage of numCorrectChars/totalChars)
*for the given line of text. The game resets with a new word after the user
*views their most recent performance until the program is quit.
 *
 *
 *We affirm that we have carried out the attached academic endeavours with
full academic honesty
 * @author Sol Ben-Ishay, Christopher Ringer, Milo Baker-Durante
 */
public class TypingSpeedTest {

    // Declarations
    // Swing Components
    final private JFrame frame;
    final private JLabel currentWord;
    final private JLabel timerLabel;
    final private JButton enterBtn;
    final private JButton startBtn;
    final private JTextField userInput;

    // Variables/Arrays/Etc
    public ArrayList<String> wordsArray = new ArrayList<>();
    double sec = 0;
    double min = 0;
    double accuracy;
    double wpm;
    int numCorrectChars = 0;
    int totalChars = 0;
    String original;
    String filePath = "Words/sentences.txt";
    Timer t;

    // Class Constructor
    public TypingSpeedTest() {

        // Create main JFrame
        frame = new JFrame();
        // Set for around center on 13 inch MacBook Pro
        frame.setBounds(250, 250, 650, 350);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        GridBagLayout gridBagLayout = new GridBagLayout();
        gridBagLayout.columnWidths = new int[]{650, 0};
        gridBagLayout.rowHeights = new int[]{322, 0};
        gridBagLayout.columnWeights = new double[]{0.0, Double.MIN_VALUE};

        gridBagLayout.rowWeights = new double[]{0.0, Double.MIN_VALUE};
        frame.getContentPane().setLayout(gridBagLayout);

        // Create ImageIcon for the Start Button
        ImageIcon play = new ImageIcon("/Users/solbenishay/Desktop/Project
4/Icons/play.png");
        Image img1 = play.getImage();
        Image newPlay = img1.getScaledInstance(20, 20,
java.awt.Image.SCALE_SMOOTH);
```

```java
        play = new ImageIcon(newPlay);

        // Create main JPanel
        JPanel panel = new JPanel();
        GridBagConstraints gbc_panel = new GridBagConstraints();
        gbc_panel.fill = GridBagConstraints.BOTH;
        gbc_panel.gridx = 0;
        gbc_panel.gridy = 0;
        frame.getContentPane().add(panel, gbc_panel);
        GridBagLayout gbl_panel = new GridBagLayout();
        gbl_panel.columnWidths = new int[]{35, 55, 361, 130, 0};
        gbl_panel.rowHeights = new int[]{30, 16, 78, 25, 22, 26, 50, 0, 0};
        gbl_panel.columnWeights = new double[]{0.0, 0.0, 0.0, 0.0,
Double.MIN_VALUE};
        gbl_panel.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, Double.MIN_VALUE};
        panel.setLayout(gbl_panel);

        // Create label for the timer
        timerLabel = new JLabel("Time: " + sec);
        GridBagConstraints gbc_timerLabel = new GridBagConstraints();
        gbc_timerLabel.anchor = GridBagConstraints.NORTH;
        gbc_timerLabel.insets = new Insets(0, 0, 5, 5);
        gbc_timerLabel.gridx = 1;
        gbc_timerLabel.gridy = 1;
        panel.add(timerLabel, gbc_timerLabel);

        // Create label for the current word/phrase
        currentWord = new JLabel("\"\"");
        currentWord.setFont(new Font("Lucida Grande", Font.BOLD, 20));
        GridBagConstraints gbc_currentWord = new GridBagConstraints();
        gbc_currentWord.anchor = GridBagConstraints.NORTH;
        gbc_currentWord.insets = new Insets(0, 0, 5, 5);
        gbc_currentWord.gridx = 2;
        gbc_currentWord.gridy = 3;
        panel.add(currentWord, gbc_currentWord);

        // Create user input field
        userInput = new JTextField();
        GridBagConstraints gbc_userInput = new GridBagConstraints();
        gbc_userInput.fill = GridBagConstraints.HORIZONTAL;
        gbc_userInput.anchor = GridBagConstraints.NORTH;
        gbc_userInput.insets = new Insets(0, 0, 5, 5);
        gbc_userInput.gridx = 2;
        gbc_userInput.gridy = 5;
        panel.add(userInput, gbc_userInput);
        userInput.setColumns(10);

        userInput.setEnabled(false);

        // Create enter button
        enterBtn = new JButton("Enter");
        enterBtn.setEnabled(false);
        // When the enter button is clicked
        enterBtn.addActionListener(e -> {
            t.stop(); // Stop the timer
            getPerformanceStats(); // Calculate the performance stats
            JOptionPane.showMessageDialog(null, "ROUND OVER!" + "\n" + "Time:
" + sec + " seconds" + "\n" + "WPM: " + wpm + "\n" + "Accuracy: " + accuracy
+ " %");
            userInput.setEnabled(false);
            enterBtn.setEnabled(false);
            sec = 0;
            resetGame();
        });
        GridBagConstraints gbc_enterBtn = new GridBagConstraints();
        gbc_enterBtn.fill = GridBagConstraints.VERTICAL;
        gbc_enterBtn.insets = new Insets(0, 0, 5, 5);
        gbc_enterBtn.gridx = 2;
        gbc_enterBtn.gridy = 6;
        panel.add(enterBtn, gbc_enterBtn);

        // Create start button
        startBtn = new JButton("Start");
        startBtn.setIcon(play); // Sets the icon to the play IconImage
        startBtn.setBackground(Color.GREEN); // Not working for some reason
        frame.getRootPane().setDefaultButton(startBtn); // Sets start button
as default if enter on keyboard is pressed
        // When the start button is clicked
        startBtn.addActionListener(e -> {
            enterBtn.setEnabled(true);
            startBtn.setEnabled(false);
            userInput.setEnabled(true);
            // Sets enter button as default if enter on keyboard is pressed
            frame.getRootPane().setDefaultButton(enterBtn);
            userInput.requestFocusInWindow();
            startBtn.setBackground(null);
            time();
        });
        GridBagConstraints gbc_startBtn = new GridBagConstraints();
        gbc_startBtn.anchor = GridBagConstraints.SOUTH;
        gbc_startBtn.insets = new Insets(0, 0, 0, 5);
        gbc_startBtn.gridx = 2;
        gbc_startBtn.gridy = 7;
        panel.add(startBtn, gbc_startBtn);

        // Sets frame visible
        frame.setVisible(true);
    }

    /**
     * This method takes the path of a txt file (to sample line of text from)
as a parameter and populates the game
     * array

     * @param textFilePath The path to the txt file to line of text from
     */
    public void getText(String textFilePath) {
        FileInputStream fileByteStream = null;
        try {
            fileByteStream = new FileInputStream(textFilePath);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

```java
        }
        assert fileByteStream != null;
        Scanner streamScnr = new Scanner(fileByteStream);
        while (streamScnr.hasNextLine()) {
            String textLine = streamScnr.nextLine();
            wordsArray.add(textLine);
        }
        streamScnr.close();
        try {
            fileByteStream.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    /**
     * This method returns a random word from the getText() populated
wordsArray
     * @return Random word from wordsArray
     */
    public String selectRandomWord() {
        Random r = new Random();
        int randomNumber = r.nextInt(wordsArray.size());
        return wordsArray.get(randomNumber);
    }

    /**
     * This method initializes the game with the words array, chooses a
random word from it, and displaying that word
     * in the currentWord JLabel
     */
    public void play() {
        getText(filePath);
        original = selectRandomWord();
        totalChars = original.length();
        currentWord.setText(original);
    }

    /**
     * This method calculates the performance statistics of the given user
input
     */
    public void getPerformanceStats() {
        String userWord = userInput.getText();
        // Accuracy
        if (userWord.length() <= original.length()) {

            for (int i=0;i<userWord.length();i++) {
                if (userWord.charAt(i) == original.charAt(i)) {
                    numCorrectChars++;
                }
            }
        }
        else {
            for (int i=0;i<original.length();i++) {
                if (userWord.charAt(i) == original.charAt(i)) {
                    numCorrectChars++;
                }
            }
        }
        accuracy = ((float) numCorrectChars/totalChars) * 100.0;
        DecimalFormat twoDec = new DecimalFormat("#.##");
        accuracy = Double.parseDouble(twoDec.format(accuracy));
        // WPM
        min = sec/60.0;
        wpm = (userWord.length() / 5.0) / min;
        wpm = Double.parseDouble(twoDec.format(wpm));
    }

    /**
     * This method initializes the timer for the game
     */
    public void time() {
        t = new Timer(100, e -> {
            sec = (float) (sec + 0.1);
            DecimalFormat oneDec = new DecimalFormat("#.#");
            sec = Double.parseDouble(oneDec.format(sec));
            timerLabel.setText("Time: " + sec);
        });
        t.start();
    }

    /**
     * This method resets the game
     */
    public void resetGame() {
        wordsArray.clear();
        getText(filePath);
        numCorrectChars = 0;
        original = "";
        if (sec != 0) {
            t.stop();
        }
        sec = 0;
        timerLabel.setText("Timer: " + sec);
        userInput.setEnabled(false);
        startBtn.setEnabled(true);
        startBtn.setBackground(Color.GREEN);
        enterBtn.setEnabled(false);
        frame.getRootPane().setDefaultButton(startBtn);
        frame.repaint();
        frame.revalidate();
        userInput.setText("");

        userInput.requestFocusInWindow();
        play();
    }

    /**
     * Main method to run the game
     * @param args
     */
    public static void main(String[] args) {
        TypingSpeedTest test = new TypingSpeedTest();
        test.play();
    }
}
```

Output (screenshots):

First Round:

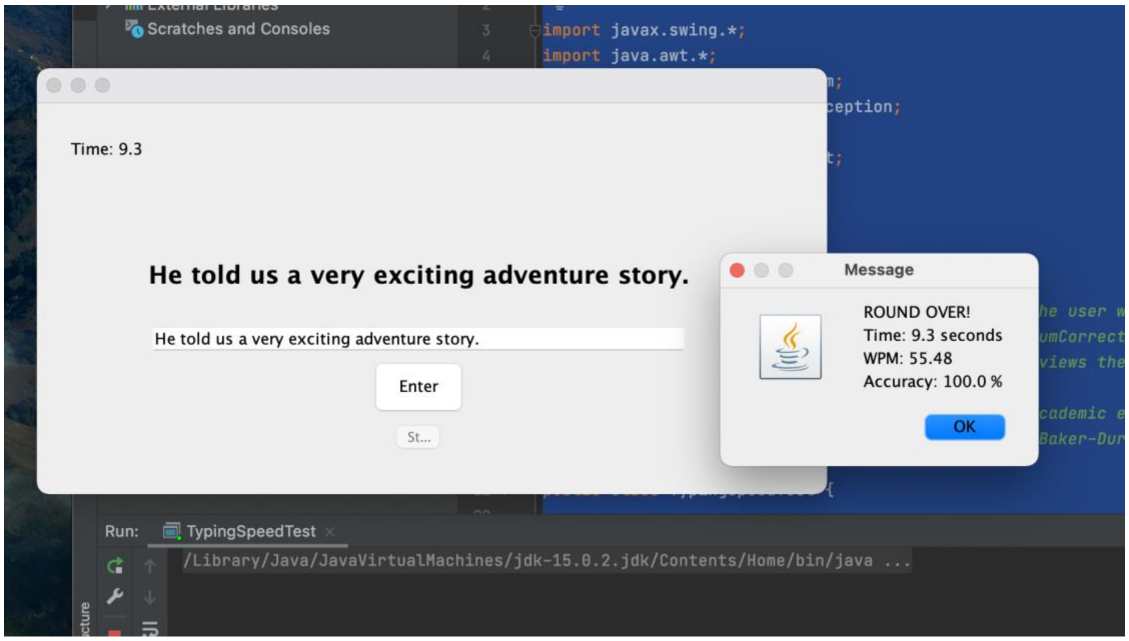


```
import java.io.IOException;
```

Time: 0.0

He told us a very exciting adventure story.

Results Window:



He told us a very exciting adventure story.

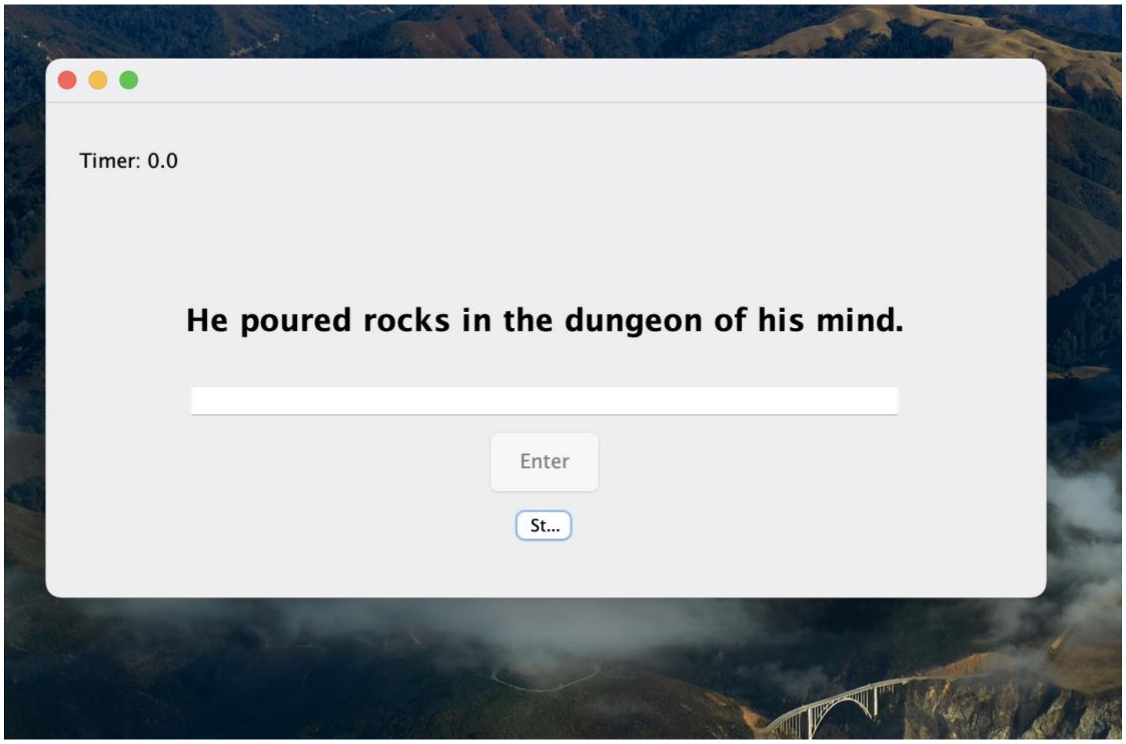
He told us a very exciting adventure story.

ROUND OVER!
Time: 9.3 seconds
WPM: 55.48
Accuracy: 100.0 %

```
import javax.swing.*;
import java.awt.*;
```

```
/Library/Java/JavaVirtualMachines/jdk-15.0.2.jdk/Contents/Home/bin/java ...
```

Next Round:



He poured rocks in the dungeon of his mind.

Keeps repeating the above until exited