1. **需求分析**
   1. 概述

打字训练测试软件是一个专门设计来提升用户打字效率和准确性的交互式学习工具。它采用面向对象的设计原则，提供了一个分布式的平台，允许用户在各种操作系统上无缝运行。该软件是解释型的，意味着它不需要编译即可执行，从而为用户提供了一个灵活且易于使用的环境。

软件的核心特点包括健壮性和安全性，确保用户在练习时的数据保护和隐私。它还具有结构中立性，使得用户可以在不同的硬件和操作系统上获得一致的体验。此外，软件的多线程设计允许同时处理多个任务，从而提高性能。

软件的用户界面设计简洁直观，易于导航，使用户能够轻松地开始练习、跟踪进度和查看统计信息。软件还提供了一个自定义练习模式，允许用户上传或输入自己的文本，从而进行个性化的打字训练。

综上所述，打字训练测试软件是一个功能全面、用户友好的工具，旨在通过系统性的练习和反馈，帮助用户有效提高打字技能，同时提供了一个安全、高效的学习环境。

以下就给出这次课题设计的题目：打字训练测试软件。

* 1. 课题分析

从这个题目来分析，要做的主要有三大点，两大模块。

打字练习模块：

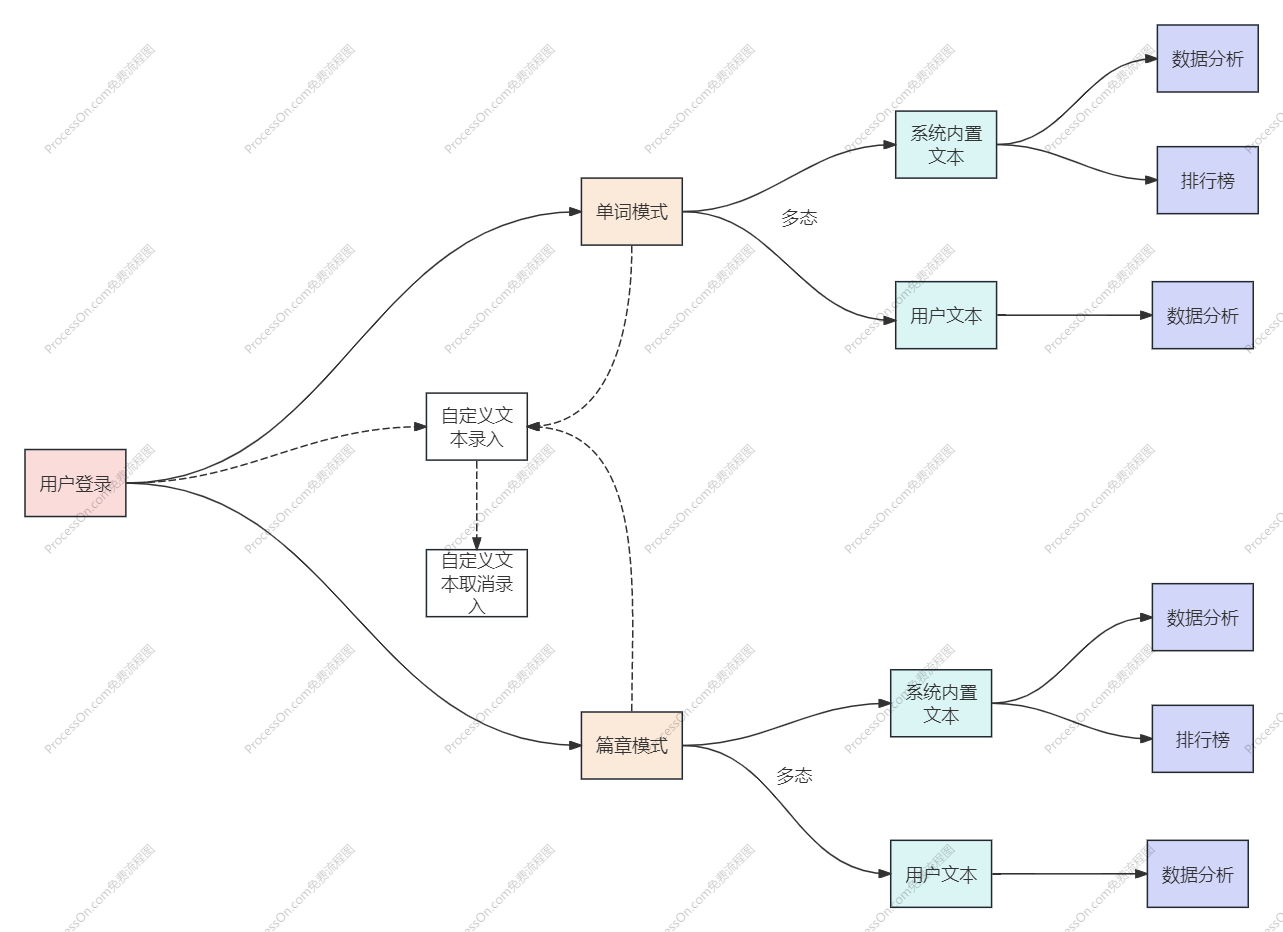
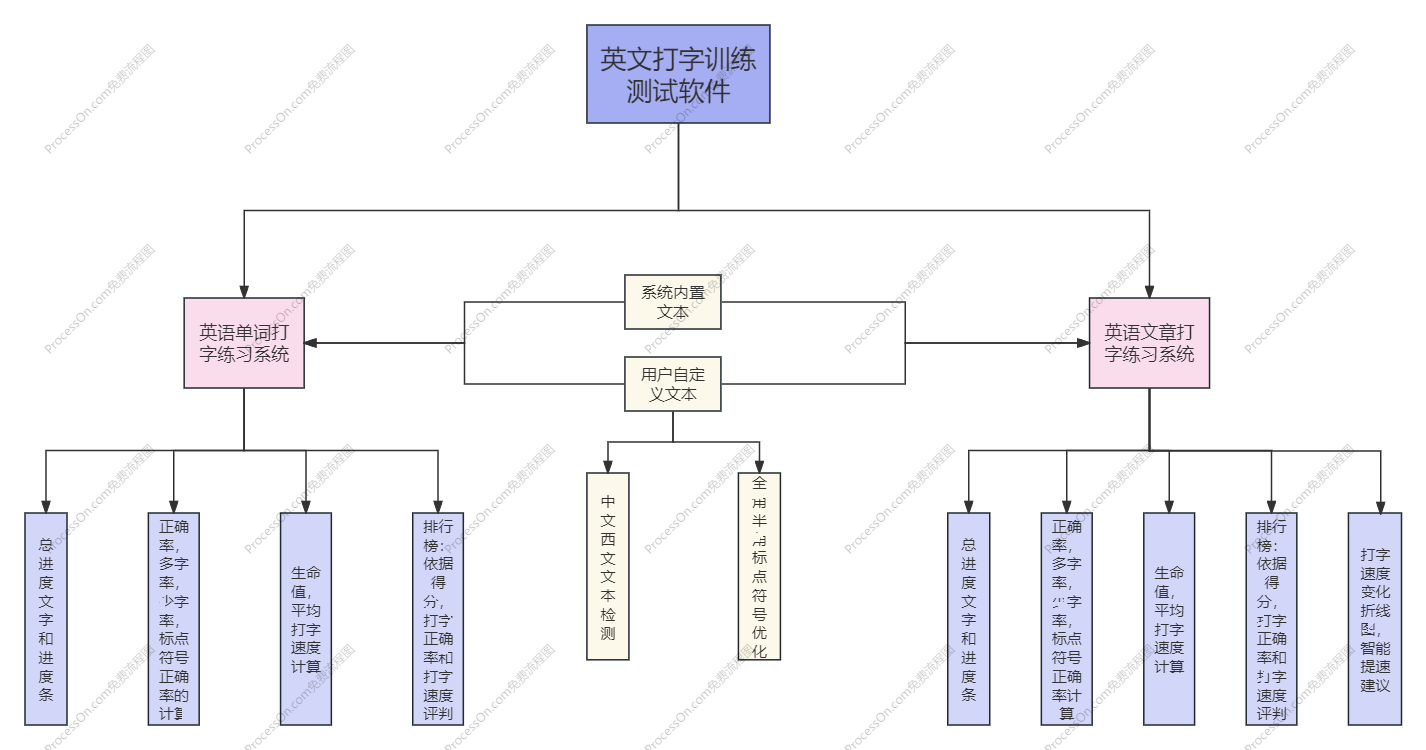
提供多种难度级别的打字练习，包括基础单词、短语、句子到文章级别。支持用户自定义练习文本，允许用户上传或输入自己的文本进行练习。练习过程中实时反馈打字速度和准确率。练习结束后提供详细的统计信息，包括总字数、准确率、速度等。

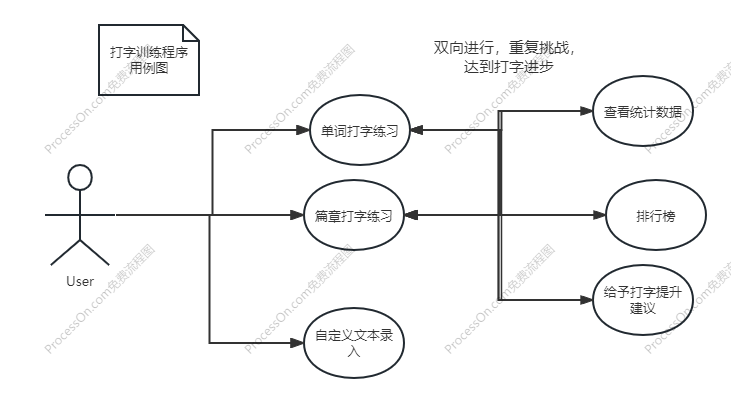
统计与报告模块：

提供用户练习历史记录和统计分析。生成练习报告，包括速度、准确率、进步趋势等。另外在结束一轮系统内置文本的打字后，提供排行榜排名功能，提高用户使用积极性。

自定义文章文字处理模块：

用户自定义输入的文章格式很可能具有问题，比如空格过多，全角半角符号不匹配；针对以上问题，我们要将用户输入的文本做一次处理，再方便代码的逐字比对，优化用户体验。

* 1. ****系统功能结构图和用例图

****

* 1. 具体目标

1.从文件中调入英文录入材料，可以显示/隐藏原始材料。

2.依次提示材料中出现的所有英文单词（可以重复），一次练习一个，回车进入下一个单词练习。

3.给出总进度

4.实时给出当前错误率，平均录入速度。

5.根据中文，西文优化特殊符号全角半角格式。

6.绘制打字速度变化折线图，并根据结果分析给出提升打字的相关建议。

* 1. 实现功能步骤

打开程序，先显示用户登录界面，输入用户名点击按钮后，在数据库中检测是否存在该用户名，有则进入功能选择页面，没有则在数据库中新建，并弹出提示框，然后进入系统。进入系统后，能选择系统自带文本进行练习或者导入用户自定义文本进行练习。在导入用户自定义文本时，根据文本是中文还是西文，对标点符号的全角半角做统一化处理。在打字页面，用户不需要点击输入框进行输入，只要用户聚焦在这个窗体，用户的输入都是有效的，与现在的主流打字软件接轨，提高用户体验。

* 1. 对开发遇到的问题的预判

多线程技术掌握不牢固：无法顺利展示多个英文字母的动画下降效果。

渲染卡顿问题：当载入大量单词时，单词移动的动画是否会顺畅移动，程序的操作是否会受到影响。

计时器的使用：何时该启动计时器，何时改结束计时器，计时器刷新的频率设计为多少才不会影响性能。

数据刷新问题：刷新时间应该设计为多少才能既能让用户取得最新的打字速度等相关信息，又不会因为信息的频繁更新而闪动，影响用户体验。

* 1. 运行环境

系统软件：Window 11 23H2 22635.4145 Windows Feature Experience Pack 1000.22700.1039.0

所需文件： login.java, textProcess.java, WordsMode.java,背景图片，系统内置文本等资源文件

语言选择：Java SE（Java Platform，Standard Edition）

工具软件：IntelliJ IDEA Utimate Edition 2023.3.6

* 1. 小结

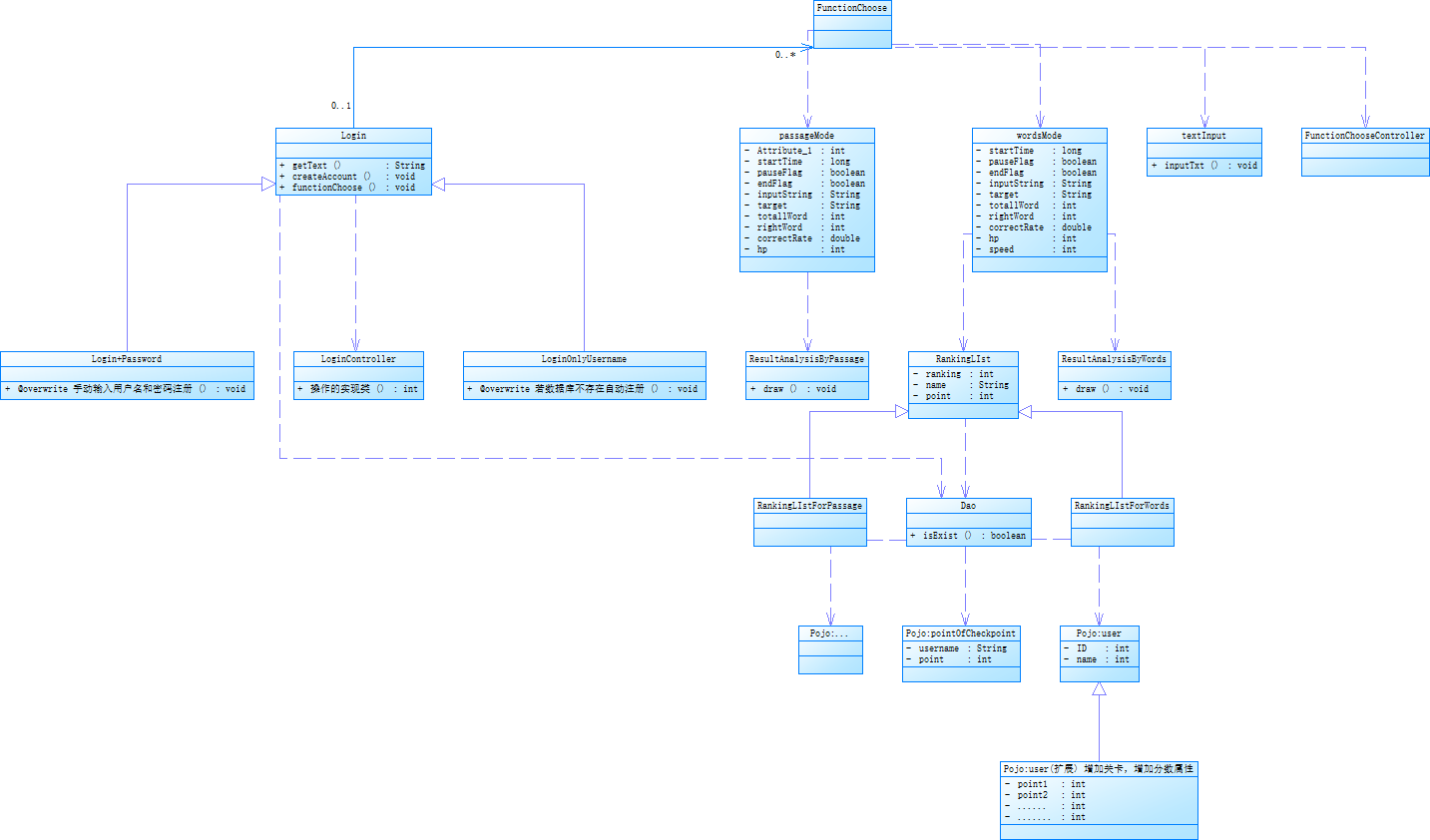
本软件旨在为用户提供一个全面的英语打字练习平台，以提高用户的打字速度和准确性。该软件将通过一系列功能和目标，帮助用户在练习中提高英语打字技能，同时提供详尽的反馈和建议，以促进用户持续进步。软件的主要功能模块包括：

1.打字练习模块：提供不同难度级别的练习，从基础单词到文章级别，支持用户自定义文本练习，允许上传或输入个人文本。练习过程中实时显示打字速度和准确率，并在练习结束后提供详细的统计信息。

2.统计与报告模块：记录用户的练习历史，提供统计分析和练习报告，包括速度、准确率、进步趋势等。此外，提供排行榜功能，以增强用户积极性。

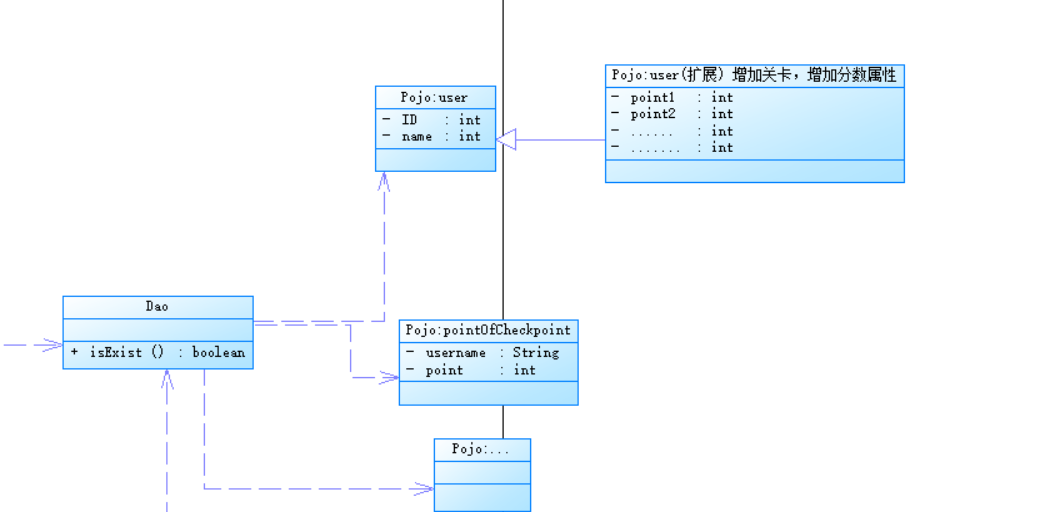
3.自定义文章文字处理模块：优化用户输入的文本格式，处理空格和全角半角符号不匹配问题，以方便代码逐字比对，提升用户体验。

1. **概要设计**

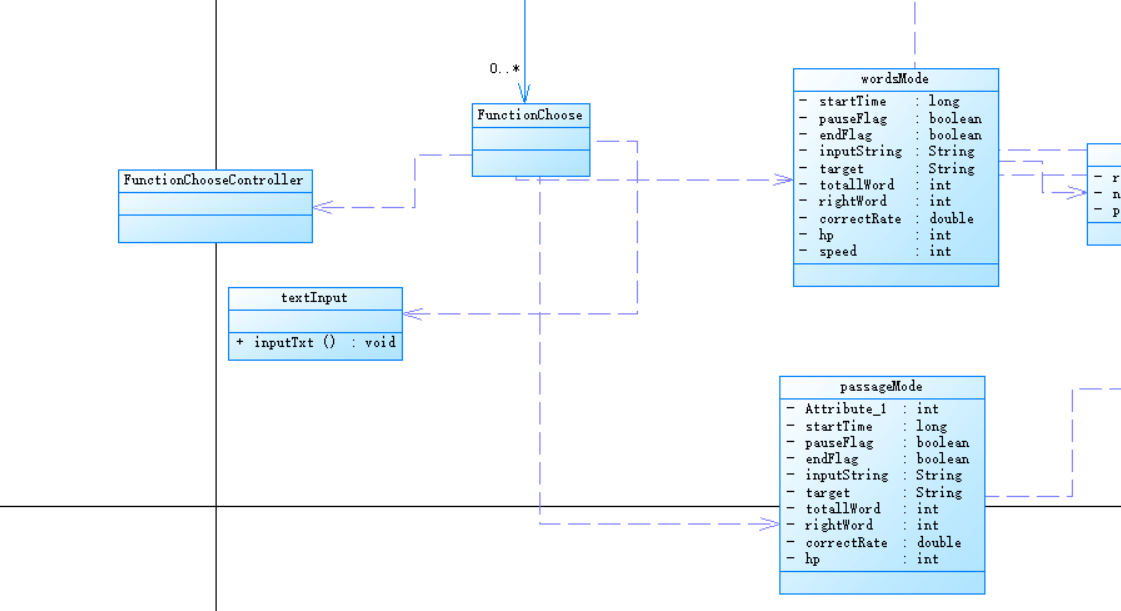
2.1 系统类图

图表 1总体类图概览

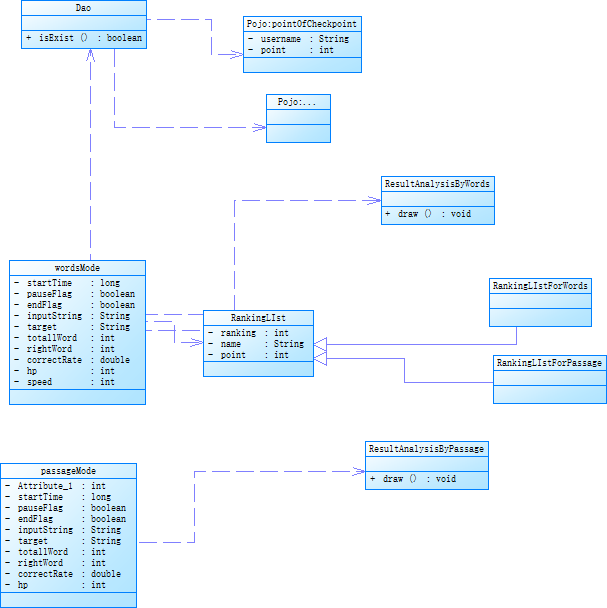
登录注册页面和数据库交互页面：

数据库及pojo类设计：

自定义文本输入和功能选择，选择全文打字还是篇章中的单词打字

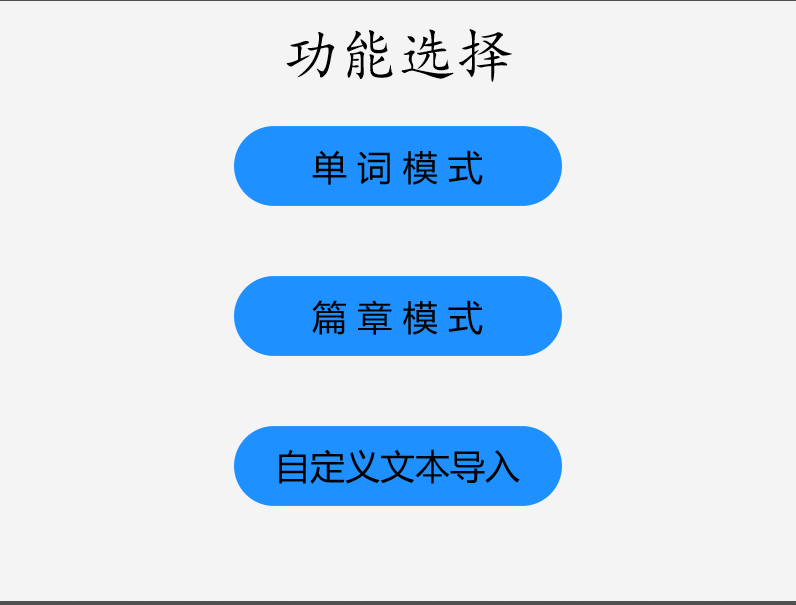
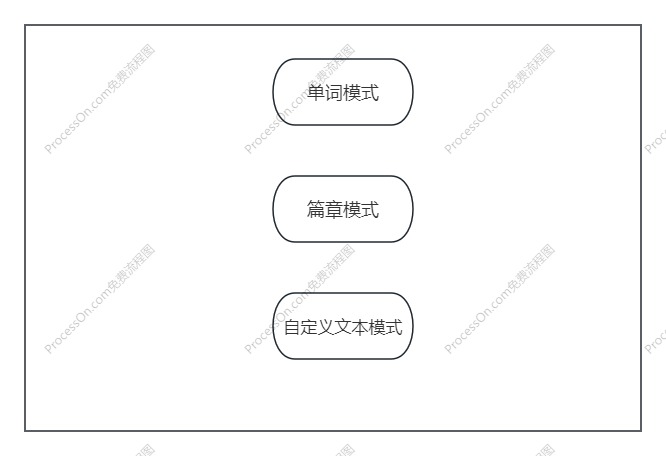


每个模式以及扩展复用，以及相关的打字速度分析及系统内置的文本挑战分数排行榜如下图所示。



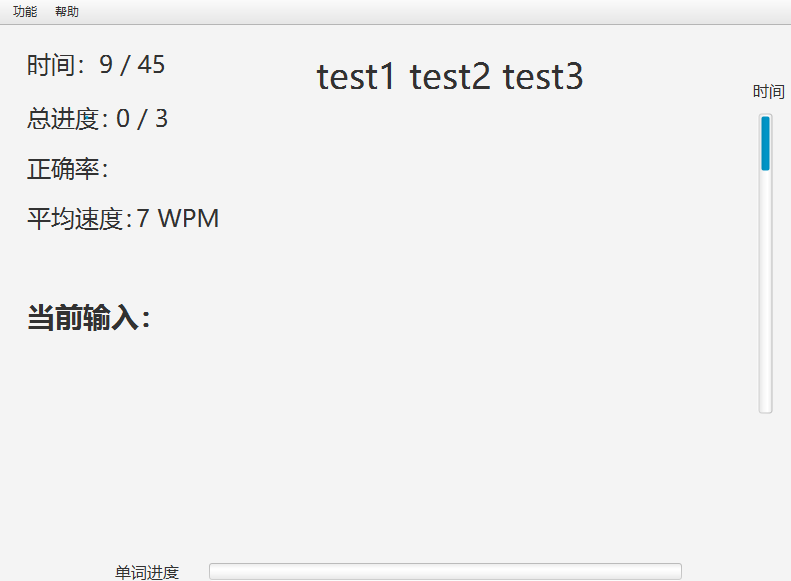
* 1. 界面设计图

程序功能选择窗口：



单词模式：仿照下图进行设计，系统内置常用的单词，进行打字游戏。





篇章模式：仿照目前主流网页打字网站进行设计

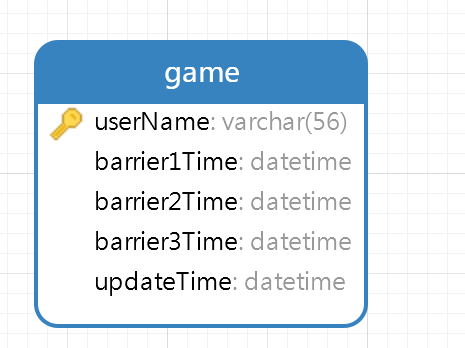


在执行完一段时间（60s）的打字任务后，给出相应的报告以及折线图分析。如下图所示。



* 1. 文件储存

数据库设计：储存用户名，和用户相关的数据。



程序内数据储存设计

创建了两个 ArrayList 对象，words 和 copyOfwords，都以初始容量为 8 进行实例化。前者储存需要显示的数据，根据用户的操作对该ArrayList进行增加和删除操作；而后者在于，玩家进入自由练习模式后，练习的词汇不会和自己已经进行增加，删除后的数组冲突，保证了数据的原始性。至于使用ArrayList 的原因。ArrayList 是一个动态数组，能够存储可变数量的元素。选择 ArrayList 的原因在于其提供了随机访问功能，能通过索引快速获取元素。此外，ArrayList 自动调整其容量，以适应元素的增加或删除，避免了固定数组的大小限制。通过初始化容量为 8，可以提高性能，减少内存的重复分配和复制操作，特别是在预计会存储较多单词，比如读取到用户自定义的长txt文档的情况下。

此外，本程序的难点主要体现在使用 JavaFX 进行前端显示，同时确保前端展示的内容与后端操作的数据保持一致性。在设计时要充分考虑用户交互体验，及时更新界面以反映数据变化。在这个过程中，选择的数据结构相对较少，旨在简化数据管理和提高性能。例如，使用简单的列表和映射可以有效地组织和存储信息，从而使得界面更新更加高效。这种设计不仅减少了复杂性，也提高了程序的可维护性，使得打字游戏的体验更加流畅和直观。通过合理的结构设计，玩家可以更加专注于游戏本身，而不是受到技术细节的干扰。

1. **详细设计**

3.1详细模块划分

总体分为三个模块，登录注册和功能选择模块，文件导入和文字全角半角处理模块，核心功能模块：打字与实时运算模块。

登录注册与功能选择模块：

用户可以注册新账户、登录现有账户，以及选择进入不同的打字训练或者选择录入自己的txt文档进行练习测试选项。功能包括：用户输入验证、错误提示、功能菜单显示等。

文件导入与文字全角半角处理模块

用户导入文本文件，并处理其中的文字格式。具体包括全角与半角字符的转换，确保用户在训练时能使用统一格式的文本。

功能包括：文件读取、字符格式检测与转换、错误处理等。

核心功能模块：打字与实时运算模块

这是软件的核心部分，负责实时监测用户的打字速度、准确性等指标。该模块会在用户打字时进行动态计算，并提供实时反馈。功能包括：计时器、正确率计算、速度统计、错误提示显示等。

具体小模块划分如下：

1.英文打字训练测试软件：

系统内置文本：提供预设的英文文本供用户练习。

用户自定义文本：允许用户上传或输入自己的文本进行练习。

2.

英语单词打字练习系统：

系统内置文本：提供预设的英语单词列表供用户练习。

用户自定义文本：用户可以添加或编辑单词列表。

3.

中文文本检测：

系统内置文本：提供预设的中文文本供用户练习。

用户自定义文本：允许用户上传或输入自己的中文文本进行练习。

4.

英语文章打字练习系统：

系统内置文本：提供预设的英文文章供用户练习。

用户自定义文本：用户可以上传或输入自己的英文文章进行练习。

5.

全用半标点符号优化：

系统内置文本：提供预设的文本，优化标点符号的输入。

用户自定义文本：用户可以上传或输入自己的文本，系统进行标点符号优化。

6.

总进度文字和进度条：

系统内置文本：显示练习的总进度。

用户自定义文本：允许用户自定义进度显示的文本。

7.

正确率，多字，少字，标点符号正确的计算：

系统内置文本：提供预设的文本，用于计算打字的正确率等指标。

用户自定义文本：用户可以上传或输入自己的文本，系统计算相关指标。

8.

生命值，平均打字速度计算：

系统内置文本：提供预设的文本，用于计算用户的“生命值”和平均打字速度。

用户自定义文本：用户可以上传或输入自己的文本，系统进行相关计算。

9.

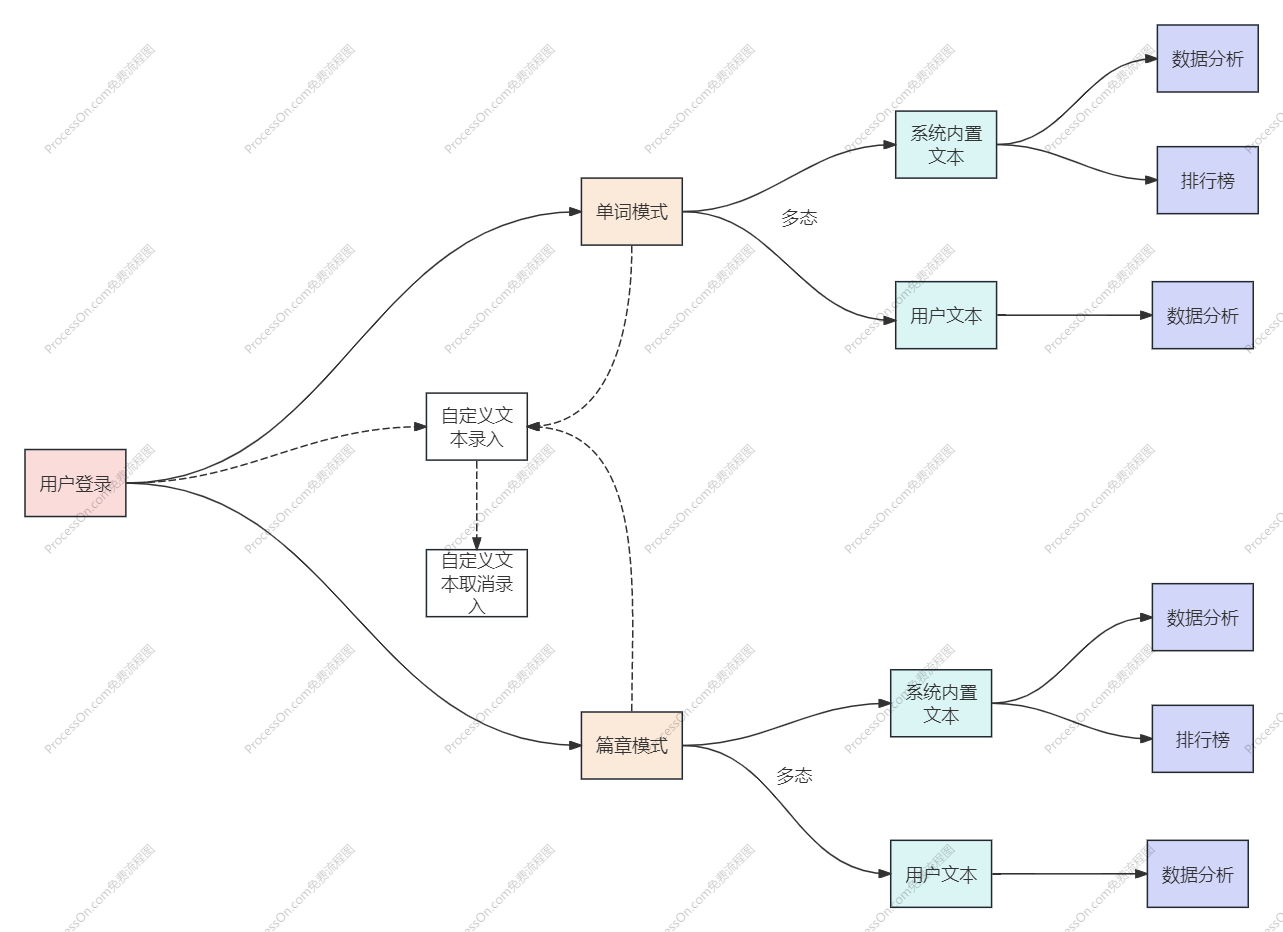
排行榜：依据得分，打字正确率和打字速度评判：

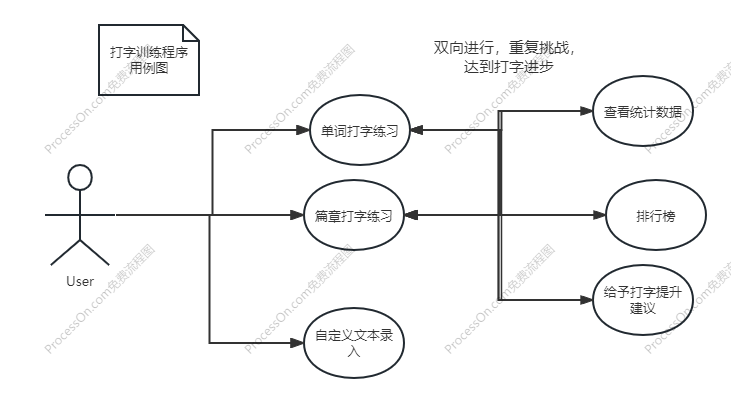
系统内置文本：提供预设的文本，用于生成排行榜。

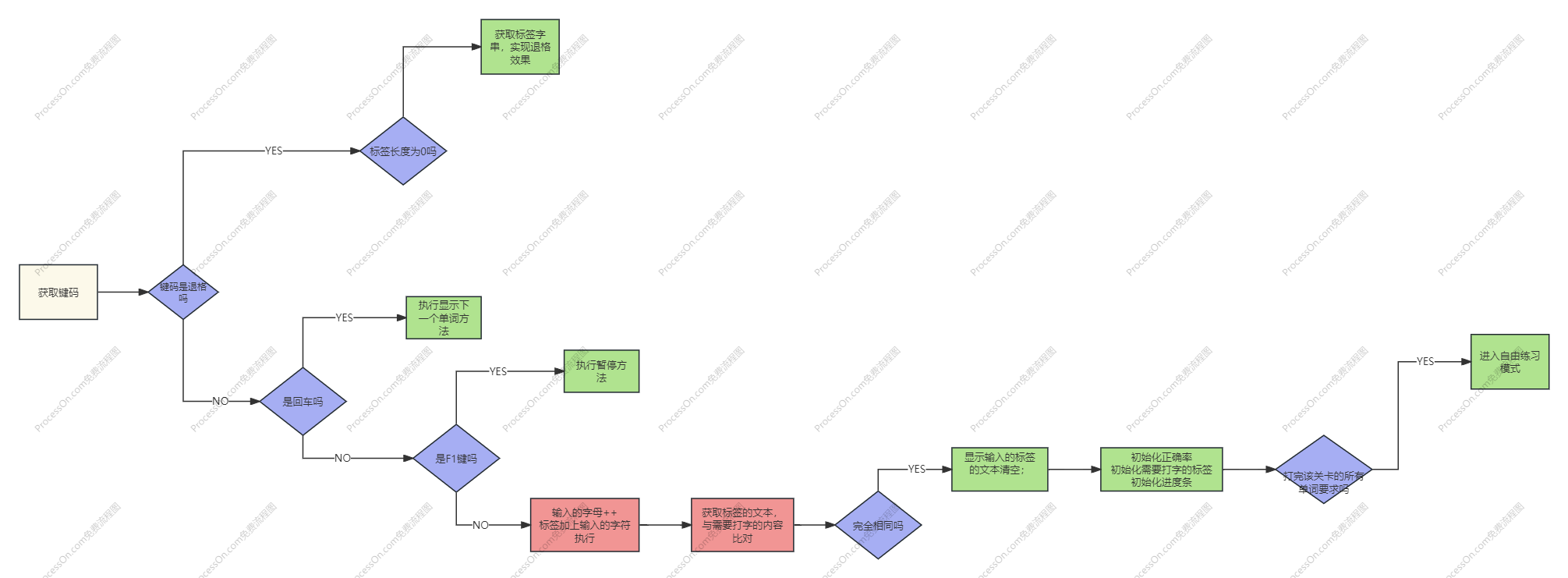
用户自定义文本：用户可以上传或输入自己的文本，系统根据得分和打字表现生成排行榜。

10.

打字速度变化折线智能提示建议：



****

3.2 程序逻辑

1.获取标签：用户开始练习时，系统首先提供一个标签，即一个单词供用户输入。

2.标签长度为0吗？：系统检查当前提供的标签长度是否为0。如果长度为0，不进行操作，防止null空指针异常。

3.是回车吗？：如果标签长度不为0，系统询问用户是否按下了回车键。回车键通常表示用户完成了对当前标签的输入。

4.是F1键吗？：如果用户按下了回车键，系统进一步检查用户是否按下了F1键。F1键可能是一个特定的快捷键，用于触发某暂停操作。

5.输入的字母+标签加上输入的字符执行：如果用户按下了F1键，系统将执行一暂停。

6.获取标签的文本，与需要打字的文本内容比对：系统获取标签文本，并将其与用户需要输入的文本进行比对，以验证用户的输入是否正确。

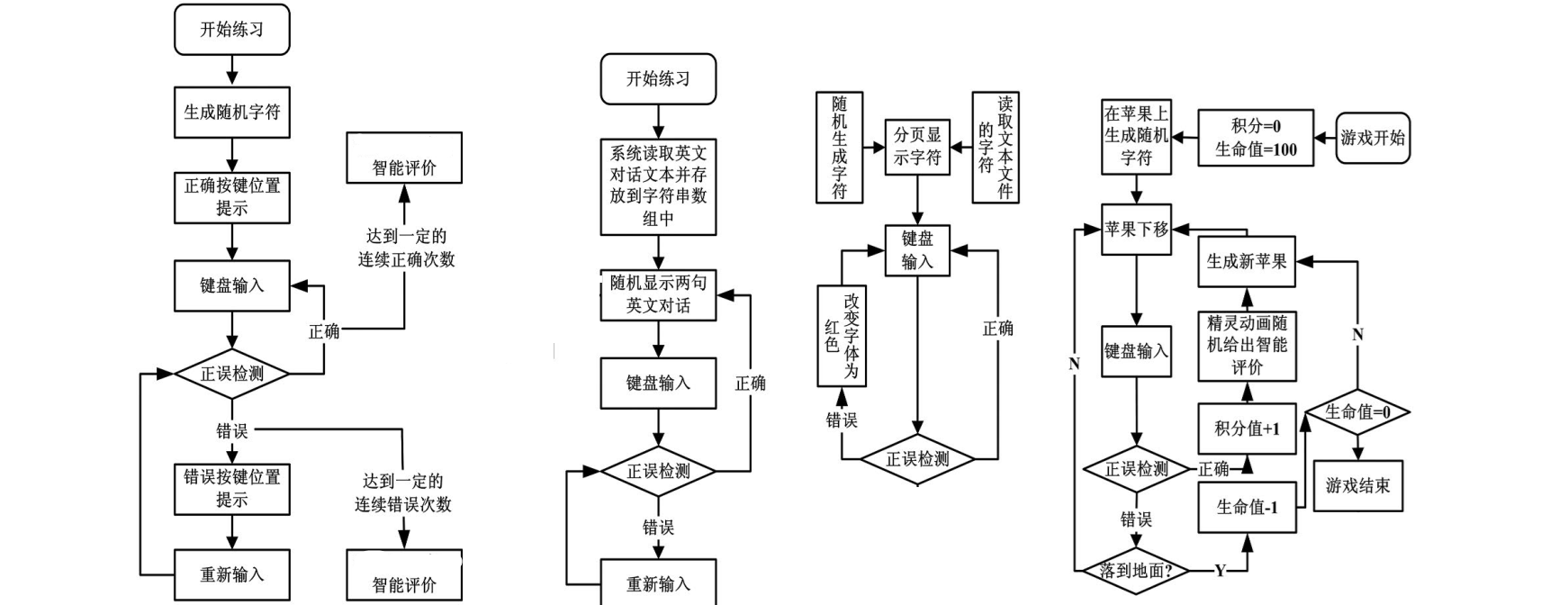
7.显示输入的标签的文本清空：一旦用户完成输入，系统将清空显示的标签文本，为下一轮练习做准备。

8.初始化正确率、需要打字的标签、进度条：系统初始化用户的正确率、需要打字的标签和进度条，为用户提供练习的准备。

9.打完该卡的所有单词要求吗？：如果用户已经完成，则流程继续下一步；如果未完成，则流程返回到获取标签的步骤，继续练习。

10.显示输入的标签的文本清空：在用户完成输入后，系统将清空显示的标签文本，为下一轮练习做准备。

11.进入自由练习模式：用户选择进入自由练习模式，允许用户自由输入文本进行练习。



3.3 聚焦失焦缩放加高斯模糊的算法实现

public void zoomIn(){  
 if (!isZoom){  
 GaussianBlur gaussianBlur = new GaussianBlur(14);  
 backPane.setEffect(gaussianBlur);  
  
 *// 获取 Pane 的宽度和高度* double centerX = backPane.getWidth() / 2;  
 double centerY = backPane.getHeight() / 2;  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*seconds*(0.3), backPane);  
 rate\*=1.08;  
 scaleTransition.setToX(rate); *// 设置放大目标X倍数* scaleTransition.setToY(rate); *// 设置放大目标Y倍数* scaleTransition.setCycleCount(1); *// 设置为播放一次* isZoom=!isZoom;  
 scaleTransition.play();  
 }  
}

public void zoomOut(){  
 if (isZoom){  
 backPane.setEffect(null);  
  
 double centerX = backPane.getWidth() / 2;  
 double centerY = backPane.getHeight() / 2;  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*seconds*(0.3), backPane);  
 rate=rate/1.08;  
 scaleTransition.setToX(rate); *// 设置放大目标X倍数* scaleTransition.setToY(rate); *// 设置放大目标Y倍数* scaleTransition.setCycleCount(1); *// 设置为播放一次* isZoom=!isZoom;  
 scaleTransition.play();  
 }  
}

该段代码实现了一个简单的缩放功能，允许用户对一个界面元素（Pane）进行放大和缩小的操作。

zoomIn() 方法用于放大界面元素，而 zoomOut() 方法则用于缩小界面元素。

两个方法都使用了 isZoom 变量来跟踪当前的缩放状态，确保缩放操作不会重复执行。

代码结构

缩放操作：使用 ScaleTransition 类来实现平滑的缩放动画效果。ScaleTransition 是 JavaFX 中用于创建缩放动画的类。

模糊效果：在放大时，通过 GaussianBlur 类为界面元素添加模糊效果，以突出放大效果。

动画设置：设置动画的持续时间为0.3秒，并且动画只播放一次（setCycleCount(1)）。

缩放倍数：缩放倍数通过变量 rate 控制，初始值为1，每次放大时乘以1.08，缩小时除以1.08。

缩放目标设置：scaleTransition.setToX(rate) 和 scaleTransition.setToY(rate) 分别设置X轴和Y轴的缩放目标倍数。

动画播放：调用 scaleTransition.play() 开始执行缩放动画。

状态切换：在每次缩放操作后，isZoom 变量的值会反转，以确保下一次调用时执行相反的操作。

命名规范：方法名 zoomIn 和 zoomOut 清晰地表达了它们的功能。

总结

该段代码展示了如何使用 JavaFX 的动画类来实现界面元素的缩放效果。通过合理地使用动画类和属性设置，可以创建出流畅且用户友好的界面交互效果。代码的结构和逻辑清晰，但未实现代码复用，可能会导致维护上的不便。

3.4 全半角标点符号统一化算法实现

public static boolean isMostlyChinese(String text) {  
 int chineseCount = 0;  
 int totalCount = 0;  
 for (char c : text.toCharArray()) {  
 if (Character.*toString*(c).matches("[\\p{IsHan}]")) {  
 chineseCount++;  
 }  
 if (!Character.*isWhitespace*(c)) {  
 totalCount++;  
 }  
 }  
  
 if (totalCount>0&&((double) chineseCount / totalCount) > 0.5){  
 return true;  
 }  
 return false;  
}

if (*isMostlyChinese*(text)) {  
 text=text.replaceAll(",", "，") *// 替换中文逗号* .replaceAll("//.", "。") *// 替换中文句号* .replaceAll("!", "！");  
 char[] c=text.toCharArray();  
 for (int i = 0; i < c.length; i++)  
 {  
 if (c[i]==32)  
 {  
 c[i]=(char)12288;  
 continue;  
 }  
 if (c[i]<127)  
 c[i]=(char)(c[i]+65248);  
 }  
 return new String(c);  
}

*//全角转半角 英文转纯英文*else {  
 text=text.replaceAll("，", ",") *// 替换中文逗号* .replaceAll("。", ".") *// 替换中文句号* .replaceAll("！", "!")  
 .replaceAll("’","'")  
 .replaceAll("‘","'");  
 char[] c = text.toCharArray();  
 for (int i = 0; i < c.length; i++) {  
 if (c[i] == 12288) {  
 *//全角空格为12288，半角空格为32* c[i] = (char) 32;  
 continue;  
 }  
 if (c[i] > 65280 && c[i] < 65375)  
 *//其他字符半角(33-126)与全角(65281-65374)的对应关系是：均相差65248* c[i] = (char) (c[i] - 65248);  
 }  
 return new String(c);

这模块代码包含两个主要功能：isMostlyChinese 方法用于检测字符串是否主要由中文字符组成，而另一个未命名的方法则用于将字符串中的全角字符转换为半角字符。

isMostlyChinese 方法分析

该方法接收一个字符串参数 text，并返回一个布尔值，表示该字符串是否主要由中文字符组成。方法的实现逻辑如下：

1.初始化两个计数器：chineseCount 用于统计中文字符的数量，totalCount 用于统计字符串中的总字符数（包括中文字符和非中文字符）。

2.遍历字符串中的每个字符，使用 Character.toString(c).matches("[\\p{IsHan}]") 判断字符是否为中文字符。如果是，chineseCount 自增。

3.同时，检查字符是否为空白字符（如空格），如果是，则 totalCount 自增。

4.最后，通过比较 chineseCount 与 totalCount 的比例是否大于0.5来决定返回值。如果大于0.5，说明中文字符占多数，返回 true；否则返回 false。

全角转半角方法分析：该方法接收一个字符串参数 text，并返回一个新的字符串，其中所有的全角字符被转换为对应的半角字符。方法的实现逻辑如下：

1.使用 text.replaceAll 方法替换字符串中的中文标点符号（逗号、句号、感叹号）为英文标点符号。

2.将字符串转换为字符数组 c，遍历数组中的每个字符。

3.对于每个字符，如果它是全角空格（Unicode值为12288），则将其转换为半角空格（Unicode值为32）。

4.对于其他全角字符（Unicode值范围为65281-65374），通过减去65248将其转换为对应的半角字符（Unicode值范围为33-126）。

5.最后，将修改后的字符数组转换回字符串并返回。

但是，对于大字符串，replaceAll 方法可能会导致性能问题，可以考虑使用更高效的字符串处理方法。

总结：这段代码展示了如何检测字符串中的中文字符比例以及如何将全角字符转换为半角字符。代码实现简洁，逻辑清晰，但在异常处理和性能优化方面还有改进空间。

3.5 单词正误判断算法实现

@FXML  
public void keyboardHit(KeyEvent event){  
 *//获取键码* KeyCode code = event.getCode();  
 String text = event.getText();  
  
 if (code==KeyCode.*BACK\_SPACE*){  
 *//判断label的长度* if (!userTextLabel.getText().isEmpty()){  
 userTextLabel.setText(userTextLabel.getText().substring(0,userTextLabel.getText().length()-1));  
 }  
 }  
 else if(code==KeyCode.*ENTER*){  
 nextWord();  
 }  
 *//点击空格 执行暂停* else if (code==KeyCode.*F1*) {  
 pause= !pause;  
 pauseAction();  
 }  
 *//输入的是文本* else {  
 letterNumber++;  
 userTextLabel.setText(userTextLabel.getText()+text);  
 *//执行正误判断* String userText=userTextLabel.getText();*//用户的文本* for (String each:wordToTypeLabel.getText().split(" ")){  
 if(userText.equals(each)){  
 *//随机变换一个颜色* Random random = new Random();  
 Color randomColor = new Color(random.nextDouble(), random.nextDouble(), random.nextDouble(), 1.0);  
 userTextLabel.setTextFill(randomColor);  
  
 currentCorrectWords++;  
 correctLetter+=userText.length();  
 *//标签面板减少1个* words.remove(userText);  
 *//输入框清空* userTextLabel.setText("");  
 *//更新正确率* initializeCorrectRate();  
 initializeWordLabel();  
 initializeProcess();  
 if (words.isEmpty()){  
 if (!endProcess){  
 endProcess=!endProcess;  
 achieveAll();  
 }  
 String next=copyOfwords.get(random.nextInt(copyOfwords.size()));  
 wordToTypeLabel.setText(next);  
 }  
 break;  
 }  
 }  
 }  
}

这模块代码是一个JavaFX应用程序中的键盘事件处理函数，主要功能是响应用户的键盘输入，并根据输入执行不同的操作。下面是对这段代码的详细分析：

1.方法定义和事件参数：该方法使用@FXML注解，表明它可能与JavaFX的FXML文件相关联。它接受一个KeyEvent参数，这允许程序访问触发事件的键的信息。

2.删除操作：当用户按下 BACK\_SPACE 键时，如果文本标签userTextLabel不为空，则删除最后一个字符。

3.确认操作：按下 ENTER 键时，调用nextWord()方法。

4.暂停操作：按下 F1 键切换pause变量的状态，并调用pauseAction()方法。

5.文本输入处理：对于其他按键（文本输入），程序将输入的文本追加到userTextLabel。然后，它遍历wordToTypeLabel中的单词，检查用户输入的文本是否与某个单词匹配。如果匹配，执行以下操作：

随机改变文本颜色。

增加正确单词计数器。

更新已输入的字符数。

从待输入的单词列表中移除匹配的单词。

清空用户文本标签。

更新正确率、单词标签和处理过程。

6.完成所有单词输入：如果所有单词都被正确输入，程序将显示所有单词已达成的提示，并随机选择一个新的单词显示。

总结来说，这段代码通过监听键盘事件来实现一个文本输入练习程序的逻辑。它处理了文本的输入、删除、确认和暂停操作，并在用户输入正确时提供反馈，如颜色变化和统计信息更新。代码中还包含了对程序流程的控制，如跳转到下一个单词、暂停操作以及在完成所有单词输入后的处理。

3.6 帮助页面算法实现

使用Java新功能字符块

public void help(){  
 pause=true;  
 pauseAction();  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("提示");  
 alert.setHeaderText(null); *// 可以设置为null，避免显示默认标题* alert.setContentText("""  
 你可以选择不同的难度  
  
 按F1表示开始或暂停  
  
 按backspace(退格)删除一个字符  
  
 在结束一轮打字练习后可按Enter(回车)进入下一个单词  
  
 选择功能菜单栏内的关闭游戏按钮可关闭游戏""");  
  
 alert.showAndWait(); *// 显示对话框并等待用户关闭* pause=false;  
 pauseAction();  
}

这段代码定义了一个名为help的方法，其主要目的是向用户展示一个包含游戏操作指南的对话框，并在展示期间暂停游戏，等待用户关闭对话框后继续游戏。以下是该方法的详细分析：

1.暂停游戏：方法开始时，首先将pause变量设置为true，这表明游戏将进入暂停状态。随后调用pauseAction()方法。

2.创建对话框：接着，代码创建了一个Alert对象，这是一个JavaFX组件，用于显示信息、警告、错误和确认对话框。在这里，它被设置为INFORMATION类型，用于显示信息性的提示。

3.配置对话框内容：通过setTitle方法设置了对话框的标题为“提示”。setHeaderText方法被调用并传入null值，这可能用于移除对话框头部的默认文本。setContentText方法包含了多行文本，详细说明了游戏操作指南，包括如何开始或暂停游戏、删除字符、进入下一个单词以及关闭游戏的方法。

4.显示对话框并等待：通过调用showAndWait()方法，对话框被显示出来，并且程序会等待用户关闭对话框。在此期间，游戏处于暂停状态，用户可以阅读对话框中的信息。

5.恢复游戏状态：用户关闭对话框后，代码将pause变量重新设置为false，并再次调用pauseAction()方法。这可能用于恢复游戏的正常运行状态。

help方法通过一个信息对话框向用户提供游戏操作指南，同时在展示期间暂停游戏，确保用户在需要帮助时能够获得及时的反馈，而不会错过游戏进程。代码中使用了JavaFX的Alert类来创建和管理对话框，这是JavaFX中用于显示信息、警告、错误和确认对话框的标准方式。

3.7 计时算法实现（不使用多线程或者Timer）

public void initializeTimer(){  
 *// 每秒执行一次* if (timeline==null){  
 timeline = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimer()));  
 }  
 if (timelineForSpeed==null){  
 timelineForSpeed = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimerForSpeed()));  
 }  
 timeline.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环* timeline.play(); *// 启动计时器* timelineForSpeed.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环* timelineForSpeed.play(); *// 启动计时器*}

这段代码定义了一个名为initializeTimer的方法，用于初始化和启动两个计时器，这两个计时器分别用于不同的计时任务。方法中没有使用传统的多线程或Timer类，而是利用了JavaFX的Timeline类来实现定时功能。下面是对这段代码的详细分析：

1.初始化计时器变量：方法首先检查两个Timeline类型的变量timeline和timelineForSpeed是否为null。这两个变量分别用于控制不同的计时任务。如果它们为null，则分别创建新的Timeline实例。

2.设置计时器任务：为每个Timeline实例创建了一个KeyFrame，设置时间间隔为1秒（Duration.seconds(1)），并关联了一个lambda表达式作为事件处理器（e -> updateTimer()和e -> updateTimerForSpeed()）。

3.设置循环次数：通过调用setCycleCount(Timeline.INDEFINITE)方法，将每个计时器的循环次数设置为无限次。这意味着一旦启动，这些计时器将不断重复执行，直到显式停止。

4.启动计时器：调用play()方法启动两个计时器。这将使得每个KeyFrame定义的事件处理器（更新方法）每秒被调用一次，从而实现定时任务的持续执行。

initializeTimer方法通过JavaFX的Timeline类实现了一个简单的计时器功能，而无需依赖于传统的多线程或Timer类。这种方法的优点在于它利用了JavaFX的动画框架，使得定时任务的执行与图形用户界面（GUI）的更新可以更加顺畅地集成。通过设置KeyFrame和无限循环，代码能够以一种简洁的方式实现定时任务的持续执行。需要注意的是，这种方法依赖于JavaFX的事件循环，因此适用于JavaFX应用程序中。

1. **调试分析**
   1. 问题以及修复情况总览

|  |  |  |  |
| --- | --- | --- | --- |
| 问题编号 | 问题内容 | 优先级别 | 解决情况 |
| 1 | 动画异常：实现聚焦动画时，只能放大一次，如果后续持续缩小，背景将离开舞台 | 重要(major) | 解决(resolved) |
| 2 | 面板异常：无法达到想要的不同pane面板具有不同透明度，以区分重点，引导用户视线 | 次要(minor) | 解决(resolved) |
| 3 | 监听事件异常：不同pane下的按钮无法点击 | 重要(major) | 解决(resolved) |
| 4 | 功能异常：暂停游戏时，虽然正确率，完成度的实时计算得到了暂停，但是用户仍然能够输入，实现作弊。 | 紧急(critical) | 减轻(alleviated) |
| 5 | 功能异常：重新开始游戏，游戏虽然数据得到初始化，但是需要打字的内容仍然延续上局游戏的内容 | 次要(minor) | 解决(resolved) |
| 6 | 功能异常：更换难度后，会再次新建一个计时器，导致计时速度翻倍 | 紧急(critical) | 解决(resolved) |
| 7 | 工具栏异常：菜单点击无效果 | 重要(major) | 解决(resolved) |
| 8 | 用户体验异常：正确率更新速度快，数据闪动，用户体验不佳 | 重要(major) | 解决(resolved) |
| 9 | 用户体验异常：玩家在打字过程中，无法返回主菜单进行自定义文本导入或者重新选择关卡 | 重要(major) | 解决(resolved) |
| 10 | IO异常:如果导入非txt文件，并不会在一开始报错，到最后进入游戏后会造成没有需要打字练习的字符异常 | 重要(major) | 解决(resolved) |
|  |  |  |  |
|  |  |  | |

* 1. 详细调试修复过程

1.动画异常：实现聚焦动画时，只能放大一次，如果后续持续缩小，背景将离开舞台

Javafx类库方法中，scaleTransition.setToX(rate); 这行代码设置了在X轴上的放大倍数，rate的值决定了放大的程度。具体来说：

初始值：当rate为1时，表示没有放大，元素保持原始大小。

放大倍数：每次调用zoomIn()方法时，rate会乘以1.08。这意味着如果你调用zoomIn()两次，rate会变成1.08（第一次放大）和1.1664（第二次放大）。这样，每次放大的比例都是基于当前状态，而不是从原始大小重新计算。

效果：因此，setToX(rate)设置的值会逐步增加，使得每次放大后的元素大小是累积的，形成总的放大效果。

简单来说，rate控制了元素在X轴的缩放程度，随着每次放大，元素会越来越大。这样就可以和缩小的方法zoomOut相配合，实现聚焦放大，失焦缩小的效果。

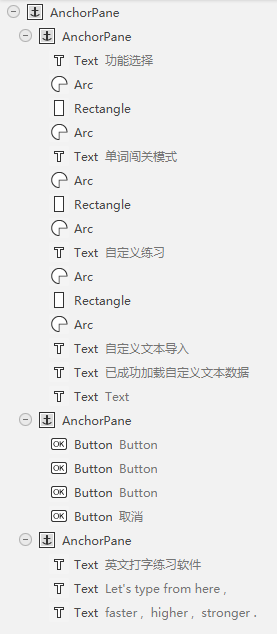
Bug产生原因：对Java类库方法使用不熟悉，没注意看API类库说明文档。

2. 面板异常：无法达到想要的不同pane面板具有不同透明度，以区分重点，引导用户视线

在JavaFX中，每个Pane类似于PhotoShop中的图层。对当前Pane属性的更改仅会影响该Pane及其下方组成部分的属性。这种设计限制了对Pane内容的修改，无法像三原色叠加那样影响所在像素区域的所有组成部分。所以，为了实现预期的效果，合理地设计每个Pane的层级关系至关重要。通过精确规划每个Pane的附属关系，开发者可以有效控制界面的表现和交互效果，从而实现复杂的UI设计和视觉效果。

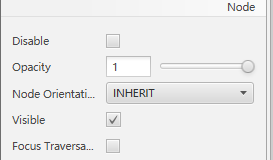
以下为重新设计的Javafx从属关系和实现的效果

合理的搭配Slogan层和选择关卡层的pane，合理的改变每一层的透明度，颜色搭配关系，模糊效果和阴影效果，以实现吸睛的视觉效果。





3. 监听事件异常：不同pane下的按钮无法点击  
不同的组件虽然被设置了visible的属性，以改变其可见度，但是即便这个按钮不可见，但它仍然是可点击的，可实现其效果；如果在一个pane的按钮被覆盖在它上面的pane所遮挡，即使按钮能够显示出来，也是无法点击触发相应的事件的。因此，需要合理的设置可见性，可操作性和不同pane间的关系。



4. 功能异常：暂停游戏时，虽然正确率，完成度的实时计算得到了暂停，但是用户仍然能够输入，实现作弊。

pausePane.setVisible(true);  
pausePane.setDisable(false);  
  
pauseLabel.setVisible(true);  
timeline.stop(); *// 停止计时器*userTextLabel.setVisible(false);  
*//高斯模糊*GaussianBlur gaussianBlur = new GaussianBlur(18);  
pane.setEffect(gaussianBlur);

暂停游戏时，虽然相关计时器得到了关闭，导致相关的数据不会得到更新，但是键盘输入的检测仍然在进行，因此玩家仍然可以进行输入；最开始尝试使用禁用显示用户输入的标签以实现禁止输入效果（因为程序设计是从前端显示的标签来获取用户字符串的），但实验验证该方法不可行。到最后发现，这其实是程序设计问题，无法解决。

所以，我在此采用一种取巧的方法，在暂停后，取消显示需要打字的字符串和用户已经打字的字符，并且对背景进行重度的高斯模糊处理，防止用户偷看。

5.功能异常：重新开始游戏，游戏虽然数据得到初始化，但是需要打字的内容仍然延续上局游戏的内容

在打字练习界面添加一个属性：保存原始的需要打字的内容，在重新开始游戏后，再次进行打字内容的初始化。

private ArrayList <String>words=new ArrayList<>(8);  
private ArrayList <String>copyOfwords=new ArrayList<>(8);

private String longText="";

6. 功能异常：更换难度后，会再次新建一个计时器，导致计时速度翻倍

if (timeline==null){  
 timeline = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimer()));  
}  
if (timelineForSpeed==null){  
 timelineForSpeed = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimerForSpeed()));  
}  
timeline.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环*timeline.play(); *// 启动计时器*timelineForSpeed.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环*timelineForSpeed.play(); *// 启动计时器*

在JavaFX中，Timeline类用于创建动画和定时任务，而在上面的代码中，对timeline和timelineForSpeed的判断主要是为了确保在每次调用时不会重复初始化这些对象。首先，timeline和timelineForSpeed在未被创建时为null，通过检查它们的状态，可以防止在每次更新时创建新的Timeline实例，从而避免不必要的资源消耗和潜在的内存泄漏。

若timeline或timelineForSpeed已被实例化，再次尝试创建它们将导致多个计时器同时运行。这不仅会导致更新逻辑的重复执行，还可能造成界面响应的混乱，影响应用程序的整体性能。因此，通过仅在它们为null时创建新的Timeline，确保了每种定时器在应用程序生命周期内仅存在一个实例。此外，setCycleCount(Timeline.INDEFINITE)方法的使用确保了计时器的无限循环，这意味着一旦启动，它们会持续运行，直到手动停止。

7. 工具栏异常：菜单点击无效果

如果创建了一个 Menu，但没有添加任何 MenuItem。这样做是合法的，但用户无法选择或进行任何操作，因此通常建议至少添加一个 MenuItem 以增强用户体验。虽然可以创建空菜单，但建议根据实际需要设计用户界面，以便让用户能够清楚地理解如何进行操作。



8. 用户体验异常：正确率更新速度快，数据闪动，用户体验不佳  
 更新数据刷新的逻辑，将正确率更新的方法移到检测到输入正确的单词后执行，这样既能降低性能消耗，又能提高用户体验。

String userText=userTextLabel.getText();*//用户的文本*for (String each:wordToTypeLabel.getText().split(" ")){  
 if(userText.equals(each)){  
 *//随机变换一个颜色* Random random = new Random();  
 Color randomColor = new Color(random.nextDouble(), random.nextDouble(), random.nextDouble(), 1.0);  
 userTextLabel.setTextFill(randomColor);  
  
 currentCorrectWords++;  
 correctLetter+=userText.length();  
 *//标签面板减少1个* words.remove(userText);  
 *//输入框清空* userTextLabel.setText("");  
 *//更新正确率* initializeCorrectRate();  
 initializeWordLabel();  
 initializeProcess();  
 if (words.isEmpty()){  
 if (!endProcess){  
 endProcess=!endProcess;  
 achieveAll();  
 }  
 String next=copyOfwords.get(random.nextInt(copyOfwords.size()));  
 wordToTypeLabel.setText(next);  
 }  
 break;  
 }  
}

9. 用户体验异常：玩家在打字过程中，无法返回主菜单进行自定义文本导入或者重新选择关卡

在每一个打字窗口类里添加主菜单的scene，使其能够自由切换。因为一定是在功能选择之后才能进入游戏，所以可以不用担心主菜单的scene为null的情况。但是，如果用户在主菜单选择的是从关卡选择进入游戏，那么在游戏中返回主菜单后也会显示关卡选择的pane，但这样比重新new一个主菜单出来相比，大大减少了内存消耗。

10.IO异常:如果导入非txt文件，并不会在一开始报错，到最后进入游戏后会造成没有需要打字练习的字符异常  
 添加相应的提示语，并且加入判断语句，保证用户输入的是合法的txt文本文件。具体代码如下

private void inputClick () {  
 *//输入txt文本* FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("选择需要练习打字的txt文件");  
 fileChooser.getExtensionFilters().add(  
 new FileChooser.ExtensionFilter("选择txt文本文件", "\*.txt")  
 );  
 File file = fileChooser.showOpenDialog(stage);  
 *//尝试读取文件* if (file != null) {  
 try {…} catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
}

* 1. 调试分析

1.动画异常：问题出现在实现聚焦动画时，放大动画只能执行一次，后续缩小动画导致背景离开舞台。修复方法是调整scaleTransition的setToX(rate)方法，确保每次放大都是基于当前状态而非原始大小重新计算。

2.面板异常：不同面板无法实现不同的透明度，以区分重点并引导用户视线。问题在于对JavaFX的Pane层级关系理解不足。修复方法是通过合理设计Pane的层级关系，实现预期的视觉效果。

3.监听事件异常：不同面板下的按钮无法点击。问题在于即使按钮不可见，它仍然是可点击的。修复方法是合理设置可见性、可操作性以及不同面板间的关系。

4.功能异常：暂停游戏时，用户仍能输入，导致作弊。修复方法是在暂停时取消显示需要打字的字符串和用户已输入的字符，并对背景进行高斯模糊处理。

5.功能异常：重新开始游戏时，需要打字的内容延续上局游戏的内容。修复方法是在打字练习界面添加属性保存原始的需要打字内容，并在重新开始游戏后进行初始化。

6.功能异常：更换难度后，会新建一个计时器，导致计时速度翻倍。修复方法是确保在每次调用时不会重复初始化Timeline对象，避免多个计时器同时运行。

7.工具栏异常：菜单点击无效果。修复方法是至少添加一个MenuItem以增强用户体验。

8.用户体验异常：正确率更新速度快，数据闪动。修复方法是将正确率更新的逻辑移到检测到输入正确的单词后执行。

9.用户体验异常：玩家无法在打字过程中返回主菜单。修复方法是在每个打字窗口类中添加主菜单的scene，允许自由切换。

10.IO异常：导入非txt文件时不会报错。修复方法是添加提示语和判断语句，确保用户输入的是合法的txt文本文件。

* 1. 调试总结与展望

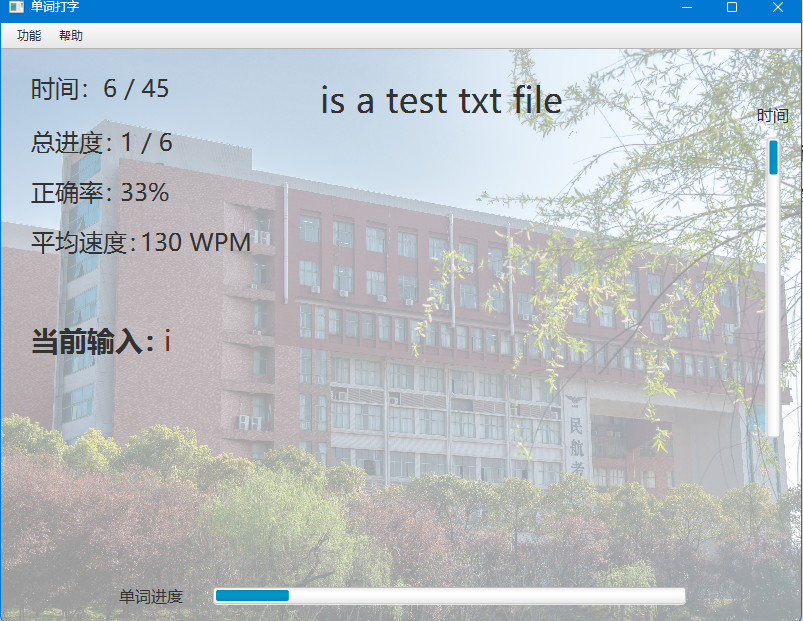
对于将来程序扩展可能存在的bug：用户输入的自定义的txt文本过长，javafx界面可能无法显示完全，但并不会影响程序的功能。以及需要添加对非txt文件的其他文本文件的导入支持。

展望未来，我们计划继续优化软件的性能和用户体验。可能会引入更复杂的动画效果，以提供更丰富的视觉反馈；改进用户界面设计，使其更加直观和易于使用；增加更多自定义选项，如字体大小、颜色主题等，以满足不同用户的需求。

**五 、 测试结果**

在login.java中调用主方法，运行程序,输入用户名，进入系统，进入功能选择页面。在导入自定义文档后，自定义练习按钮会变成蓝色，并且变得可以点击，进入自由练习，如图所示。

进入游戏后，就可以自由练习，详情演示可见演示视频。



**六 、 参考文献**

1.Monkeytype. (n.d.). **Monkeytype**. Retrieved September 9, 2024, from [Monkeytype](https://monkeytype.com/)

该网站是一个在线英语打字平台，它仅显示需要打字的文本，而不显示用户实际输入的内容。输入的正确性和错误通过红色和绿色进行区分。计时器会在用户输入第一个字符后自动启动，而无需用户手动开始计时，这体现了平台的智能化设计。此外，在用户完成30秒短篇文章的打字后，系统会自动探测用户是否暂时离开，并提供详细的反馈，包括正确字符数、不正确字符数、超出字符数、遗漏字符数以及平均打字速度。平台还具备绘制单位时间内打字速度变化的功能，并以折线图的形式呈现。在使用JavaFX实现方面，可以尝试使用文字叠加上去，输入每一个字符后判断，然后逐字判断正误设置颜色。平均速度变化亦可通过计算实现。

2.51CTO. (2021, August 25). **Java打字游戏代码**. Retrieved September 9, 2024, from [51CTO](https://blog.51cto.com/u_16175517/10909410)

具有较多BUG，使用JavaFX的控件较多，在出现100以上个字母时界面会出现明显的掉帧卡顿，需要对性能进行优化。

3.GitHub. (n.d.). **JAVA打字小游戏**. Retrieved September 9, 2024, from [GitHub](https://github.com/)

该程序只能手打字母，但在正确输入后会出现向四周发射弹幕的动画效果，且更换了鼠标的光标样式，较为美观，值得采纳。

4.GitHub. (n.d.). **利用GUI界面布局实现一个小的打字得分游戏**. Retrieved September 9, 2024, from [GitHub](https://github.com/)

该程序只能手打随机的字母，无法实现单词或者长篇篇章的打字练习，且需要手动点击输入框，手动敲击回车验证，体验效果较差。在程序设计中，应该只要鼠标聚焦在程序上，无论在哪里打字都应实现正误的验证。

5.张芸, & 王继东. (n.d.). **基于Microsoft Agent的智能打字软件设计与实现**.

使用AI助手提示内容，方式新颖，但与我课设的方向不一致，但值得借鉴。

6.冯素梅. (n.d.). **基于C#的打字测试软件研究与开发**.

基于C#语言开发，具有面向对象的特点，值得借鉴类设计。界面简捷，组织测试方便，评分精确合理，成绩回收容易，统计成绩方便，能进行水平分析。但是需要手动交卷进行确认取消等操作，相比现代的主流打字程序不太讨人喜欢。

**七 、 附录**

public class LoginController {  
  
 public Stage stage;  
 @FXML  
 private AnchorPane mainPane;  
 @FXML  
 private AnchorPane photoPane;  
 @FXML  
 private AnchorPane bodyPane;  
 @FXML  
 private AnchorPane backPane;  
 @FXML  
 private TextField textField;  
  
 private boolean isZoom=false;  
 private double rate=1;  
  
  
  
 public void zoomIn(){  
 if (!isZoom){  
 GaussianBlur gaussianBlur = new GaussianBlur(14);  
 backPane.setEffect(gaussianBlur);  
  
 *// 获取 Pane 的宽度和高度* double centerX = backPane.getWidth() / 2;  
 double centerY = backPane.getHeight() / 2;  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*seconds*(0.3), backPane);  
 rate\*=1.08;  
 scaleTransition.setToX(rate); *// 设置放大目标X倍数* scaleTransition.setToY(rate); *// 设置放大目标Y倍数* scaleTransition.setCycleCount(1); *// 设置为播放一次* isZoom=!isZoom;  
 scaleTransition.play();  
 }  
 }  
  
 public void zoomOut(){  
 if (isZoom){  
 backPane.setEffect(null);  
  
 double centerX = backPane.getWidth() / 2;  
 double centerY = backPane.getHeight() / 2;  
 ScaleTransition scaleTransition = new ScaleTransition(Duration.*seconds*(0.3), backPane);  
 rate=rate/1.08;  
 scaleTransition.setToX(rate); *// 设置放大目标X倍数* scaleTransition.setToY(rate); *// 设置放大目标Y倍数* scaleTransition.setCycleCount(1); *// 设置为播放一次* isZoom=!isZoom;  
 scaleTransition.play();  
 }  
 }  
  
  
 public void initialize () {  
 initialField();  
 initialMainGraph();  
 }  
  
 public void initialField(){  
  
 textField.focusedProperty().addListener((observable, oldValue, newValue) -> {  
 if (newValue) {  
 *// 聚焦时执行的操作* zoomIn();  
 }  
 });  
 }  
 public void initialMainGraph () {  
 Image image = new Image("photo1.jpg");  
 BackgroundImage backgroundImage = new BackgroundImage(  
 image, null, null, null,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*,  
 false, false, true, false));  
  
 photoPane.setBackground(new Background(backgroundImage));  
 }  
  
 public void zoomOutAndLostFocus(){  
 zoomOut();  
 bodyPane.requestFocus();  
 }  
  
  
 public void gotoFunction() throws IOException {  
 String s=textField.getText();  
 if (!s.isEmpty()){  
 stage.setTitle("");  
 Scene fun=HelloApplication.*getScene*(stage);  
 stage.setScene(fun);  
 stage.show();  
 }  
 else {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("提示");  
 alert.setHeaderText(null);  
 alert.setContentText(" 哥们，你还没输入账户名");  
 alert.showAndWait();  
 }  
  
 }  
  
}

public class FunctionController {  
 public Scene myself = null;  
 private Stage stage;  
 @FXML  
 public Text prompt;  
 @FXML  
 public Text fileName;  
 @FXML  
 public Arc arc1;  
 @FXML  
 public Arc arc2;  
 @FXML  
 public Rectangle rec;  
 @FXML  
 public Text textCustomized;  
 @FXML  
 public AnchorPane mainPane;  
 @FXML  
 public AnchorPane barrierPane;  
 @FXML  
 private AnchorPane sloganPane;  
  
 private String s = "";  
  
  
 public Stage getStage () {  
 return stage;  
 }  
  
 public void setStage (Stage stage) {  
 this.stage = stage;  
 }  
  
  
 @FXML  
 private void inputClick () {  
 *//输入txt文本* FileChooser fileChooser = new FileChooser();  
 fileChooser.setTitle("选择需要练习打字的txt文件");  
 fileChooser.getExtensionFilters().add(  
 new FileChooser.ExtensionFilter("选择txt文本文件", "\*.txt")  
 );  
 File file = fileChooser.showOpenDialog(stage);  
  
 *//尝试读取文件* if (file != null) {  
 try {  
 s = new String(Files.*readAllBytes*(file.toPath()));  
 prompt.setVisible(true);  
 *//读取文件名字* String fileNameText = file.getName();  
 fileName.setText(fileNameText);  
 fileName.setVisible(true);  
 arc1.setFill(Color.*DODGERBLUE*);  
 arc2.setFill(Color.*DODGERBLUE*);  
 rec.setFill(Color.*DODGERBLUE*);  
 textCustomized.setDisable(false);  
 arc1.setDisable(false);  
 arc2.setDisable(false);  
 rec.setDisable(false);  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 }  
  
 @FXML  
 private void customizedGame () throws IOException {  
 *//自定义文本开始游戏  
 //先对 文本进行半角全角西文中文的识别转换* s = textProcess.*formatPunctuation*(s);  
 WordsMode.*wordsMode*(stage, s, myself);  
 }  
  
 @FXML  
 public void initialize () {  
 initialMainGraph();  
 initializeGraph();  
 }  
  
 public void initialMainGraph () {  
 Image image = new Image("photo1.jpg");  
 BackgroundImage backgroundImage = new BackgroundImage(  
 image, null, null, null,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*,  
 false, false, true, false));  
  
 mainPane.setBackground(new Background(backgroundImage));  
 }  
  
 public void initializeGraph () {  
 Image image = new Image("levelSelect.png");  
 BackgroundImage backgroundImage = new BackgroundImage(  
 image, BackgroundRepeat.*NO\_REPEAT*,  
 BackgroundRepeat.*NO\_REPEAT*, null,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*,  
 false, false, true, false));  
  
 barrierPane.setBackground(new Background(backgroundImage));  
 }  
  
 public void normalGame () {  
 barrierPane.setVisible(true);  
 barrierPane.setDisable(false);  
 GaussianBlur gaussianBlur = new GaussianBlur(15);  
 mainPane.setEffect(gaussianBlur);  
 sloganPane.setVisible(false);  
 }  
  
 public void cancel () {  
 barrierPane.setVisible(false);  
 barrierPane.setDisable(true);  
 mainPane.setEffect(null);  
 sloganPane.setVisible(true);  
 }  
  
 public void b1 () throws IOException {  
 String s = getIOTxt("b1");  
 WordsMode.*wordsMode*(stage, s, myself);  
 }  
  
 public void b2 () throws IOException {  
 String s = getIOTxt("b2");  
 WordsMode.*wordsMode*(stage, s, myself);  
 }  
  
 public void b3 () throws IOException {  
 String s = getIOTxt("b3");  
 WordsMode.*wordsMode*(stage, s, myself);  
 }  
  
 public String getIOTxt (String barrier) {  
 String s = "";  
 String filePath = "src/main/resources/" + barrier + ".txt";  
 try {  
 s = Files.*readString*(Paths.*get*(filePath));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return s;  
 }  
}

public class wordsModeController {  
 public Stage stage=null;  
 public Scene menuScene=null;  
 private boolean pause=false;  
 private ArrayList <String>words=new ArrayList<>(8);  
 private ArrayList <String>copyOfwords=new ArrayList<>(8);  
 private int currentCorrectWords=0;  
 private int totalWords=0;  
 private int totalTime=45;  
 private Timeline timeline=null;  
 private Timeline timelineForSpeed=null;  
 private int timeForSpeed=0;  
 private int durationTime=0;  
 private int letterNumber=0;  
 private int correctLetter=0;  
 private boolean showText=true;  
 private boolean end=false;  
 private boolean endProcess =false;  
 private String longText="";  
  
  
  
 @FXML  
 private AnchorPane pane;  
 @FXML  
 private AnchorPane endPane;  
 @FXML  
 private Label userTextLabel;  
 @FXML  
 private Label timeLabel;  
 @FXML  
 private Label processLabel;  
 @FXML  
 private Label speedLabel;  
 @FXML  
 private Label wordToTypeLabel;  
 @FXML  
 private ProgressBar progressBar;  
 @FXML  
 private ProgressBar timeBar;  
 @FXML  
 private Label pauseLabel;  
 @FXML  
 private Label correctRatetLabel;  
 @FXML  
 private AnchorPane pausePane;  
  
 @FXML  
 public void startHit(ActionEvent event){  
 }  
  
  
 @FXML  
 public void keyboardHit(KeyEvent event){  
 *//获取键码* KeyCode code = event.getCode();  
 String text = event.getText();  
  
 if (code==KeyCode.*BACK\_SPACE*){  
 *//判断label的长度* if (!userTextLabel.getText().isEmpty()){  
 userTextLabel.setText(userTextLabel.getText().substring(0,userTextLabel.getText().length()-1));  
 }  
 }  
 else if(code==KeyCode.*ENTER*){  
 nextWord();  
 }  
 *//点击空格 执行暂停* else if (code==KeyCode.*F1*) {  
 pause= !pause;  
 pauseAction();  
 }  
 *//输入的是文本* else {  
 letterNumber++;  
 userTextLabel.setText(userTextLabel.getText()+text);  
 *//执行正误判断* String userText=userTextLabel.getText();*//用户的文本* for (String each:wordToTypeLabel.getText().split(" ")){  
 if(userText.equals(each)){  
 *//随机变换一个颜色* Random random = new Random();  
 Color randomColor = new Color(random.nextDouble(), random.nextDouble(), random.nextDouble(), 1.0);  
 userTextLabel.setTextFill(randomColor);  
  
 currentCorrectWords++;  
 correctLetter+=userText.length();  
 *//标签面板减少1个* words.remove(userText);  
 *//输入框清空* userTextLabel.setText("");  
 *//更新正确率* initializeCorrectRate();  
 initializeWordLabel();  
 initializeProcess();  
 if (words.isEmpty()){  
 if (!endProcess){  
 endProcess=!endProcess;  
 achieveAll();  
 }  
 String next=copyOfwords.get(random.nextInt(copyOfwords.size()));  
 wordToTypeLabel.setText(next);  
 }  
 break;  
 }  
 }  
  
 }  
 }  
  
 public void resetWordToTypeLabel(){  
 StringBuilder content= new StringBuilder();  
 for (String each:copyOfwords){  
 content.append(each).append(" ");  
 }  
 wordToTypeLabel.setText(content.toString());  
 }  
  
  
 public void level1 () {  
 totalTime=45;  
 durationTime=0;  
 currentCorrectWords=0;  
 letterNumber=0;  
 correctLetter=0;  
  
 resetWordToTypeLabel();  
 userTextLabel.setText("");  
 initializeProcess();  
 initializeCorrectRate();  
 initializeSpeed();  
 }  
 public void level2 (){  
 totalTime=30;  
 durationTime=0;  
 currentCorrectWords=0;  
 letterNumber=0;  
 correctLetter=0;  
  
 resetWordToTypeLabel();  
 userTextLabel.setText("");  
 initializeProcess();  
 initializeCorrectRate();  
 initializeSpeed();  
 }  
 public void level3 (){  
 totalTime=20;  
 durationTime=0;  
 currentCorrectWords=0;  
 letterNumber=0;  
 correctLetter=0;  
  
 resetWordToTypeLabel();  
 userTextLabel.setText("");  
 initializeProcess();  
 initializeCorrectRate();  
 initializeSpeed();  
 }  
 public void level4 (){  
 totalTime=15;  
 durationTime=0;  
 currentCorrectWords=0;  
 letterNumber=0;  
 correctLetter=0;  
  
 resetWordToTypeLabel();  
 userTextLabel.setText("");  
 initializeProcess();  
 initializeCorrectRate();  
 initializeSpeed();  
 }  
 public void changeTextVisibility(){  
 *//现在是展示文本，改为不展示* if (showText){  
 showText=!showText;  
 wordToTypeLabel.setVisible(false);  
 }  
 *//现在是不展示文本，改为展示* else{  
 showText=!showText;  
 wordToTypeLabel.setVisible(true);  
 }  
 }  
  
 @FXML  
 public void initialize (){  
 *//words.clear();* initializeGraph();  
 initializeTimer();  
 }  
  
 *//设置标签的字* public void initializeWordLabel(){  
 StringBuilder content= new StringBuilder();  
 for (String each:words){  
 content.append(each).append(" ");  
 }  
 wordToTypeLabel.setText(content.toString());  
 }  
 public void initializeProcess(){  
 if (!end){  
 String s=currentCorrectWords+" / "+ totalWords;  
 processLabel.setText(s);  
 progressBar.setProgress(currentCorrectWords/(double)totalWords);  
 }  
 }  
 public void initializeTimer(){  
 *// 每秒执行一次* if (timeline==null){  
 timeline = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimer()));  
 }  
 if (timelineForSpeed==null){  
 timelineForSpeed = new Timeline(new KeyFrame(Duration.*seconds*(1), e -> updateTimerForSpeed()));  
 }  
 timeline.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环* timeline.play(); *// 启动计时器* timelineForSpeed.setCycleCount(Timeline.*INDEFINITE*); *// 无限循环* timelineForSpeed.play(); *// 启动计时器* }  
 private void updateTimerForSpeed(){  
 timeForSpeed++;  
 initializeSpeed();  
 }  
 private void updateTimer() {  
 durationTime++;  
  
 if (!end){  
 timeLabel.setText(durationTime+" / "+totalTime);  
 *//更新进度条* timeBar.setProgress(durationTime/(double)totalTime);  
 *//更新进度条* }  
  
 *//判断是否超时，结束* if (durationTime/(double)totalTime>=1){  
 */\*pauseLabel.setVisible(false);  
 timeline.stop(); // 停止计时器  
 userTextLabel.setVisible(true);\*/* timeline.stop();  
 endPane.setVisible(true);  
 *//装饰性内容* GaussianBlur gaussianBlur = new GaussianBlur(10);  
 pane.setEffect(gaussianBlur);  
  
 }  
 }  
  
 public void initializeSpeed(){  
 double speed=letterNumber/((double)timeForSpeed/60);  
 String temp=String.*valueOf*(Math.*round*(speed))+" WPM";  
 speedLabel.setText(temp);  
 }  
  
 public void initializeCorrectRate(){  
 if (!end){  
 double rate=correctLetter/(double) letterNumber\*100;  
 correctRatetLabel.setText(String.*valueOf*(Math.*round*(rate)) + "%" );  
 }  
  
 }  
  
 public void achieveAll(){  
 *//打完所有字数，仍然继续练习* end=true;  
 timeLabel.setDisable(true);  
 processLabel.setDisable(true);  
 correctRatetLabel.setDisable(true);  
 timeline.stop();  
 }  
  
 public void nextWord(){  
 if (end){  
 Random random = new Random();  
 String next=copyOfwords.get(random.nextInt(copyOfwords.size()));  
 wordToTypeLabel.setText(next);  
 }  
 }  
  
 public void pauseAction(){  
 if (pause){  
 *//执行暂停* pausePane.setVisible(true);  
 pausePane.setDisable(false);  
  
 pauseLabel.setVisible(true);  
 timeline.stop(); *// 停止计时器* userTextLabel.setVisible(false);  
 *//高斯模糊* GaussianBlur gaussianBlur = new GaussianBlur(18);  
 pane.setEffect(gaussianBlur);  
 }  
 else {  
 *//终止暂停* pauseLabel.setVisible(false);  
 timeline.play(); *// 停止计时器* userTextLabel.setVisible(true);  
 pane.setEffect(null);  
  
 pausePane.setVisible(false);  
 pausePane.setDisable(true);  
 }  
 }  
  
 public void exitGame(){  
 stage.close();  
 }  
 public void help(){  
 pause=true;  
 pauseAction();  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("提示");  
 alert.setHeaderText(null); *// 可以设置为null，避免显示默认标题* alert.setContentText("""  
 你可以选择不同的难度  
  
 按F1表示开始或暂停  
  
 按backspace(退格)删除一个字符  
  
 在结束一轮打字练习后可按Enter(回车)进入下一个单词  
  
 选择功能菜单栏内的关闭游戏按钮可关闭游戏""");  
  
 alert.showAndWait(); *// 显示对话框并等待用户关闭* pause=false;  
 pauseAction();  
 }  
 public void initializeGraph(){  
 Image image = new Image("photo2.jpg");  
 BackgroundImage backgroundImage = new BackgroundImage(  
 image, null, null, null,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*,  
 false, false, true, false));  
  
 pane.setBackground(new Background(backgroundImage));  
 }  
 public void restart() throws IOException {  
 WordsMode.*wordsMode*(stage,longText,menuScene);  
 }  
  
  
 public void initializeText(String s){  
 s = s.replace(",", " ").replace(".", " ");  
 String[] words = s.split(" ");  
 this.longText=s;  
 for (String each:words){  
 this.words.add(each);  
 this.copyOfwords.add(each);  
 }  
 totalWords=this.words.size();  
 initializeWordLabel();  
 initializeProcess();  
 }  
  
 public void backToMenu(){  
 stage.setTitle("主菜单");  
 stage.setScene(menuScene);  
 stage.show();  
 }  
}