Ü 5a.3: Worttrainer fertig in git und als jar

✔ **Done:** Make a submission ✔ **Done:** Receive a grade

Due: Monday, 14 November 2022, 11:59 PM

1. Schritt: git

Verwende unter folgendem Git Classroom Link deinen Github-Account (falls du noch keinen Github-Account hast, erstelle bitte einen neuen Account - dein Nachname/Login-Name sollte im Github-Accountnamen ersichtlich sein) und erstelle ein neues Git Repository:

https://classroom.github.com/a/i6iHHfxM

Du bekommst durch den Link ein neues eigenes Repository. Klone dieses Repository mittels eGit, so dass du dann in Eclipse ein mit git verwaltetes Java-Projekt hast (siehe Skriptum). Kontrolliere deinen Projektnamen und ändere ihn gegebenenfalls auf Worttrainer (siehe Skriptum 5a.3: Grundlagen Versionierung mit git Seite 8) so dass dein Eclipse-Projekt den Namen Worttrainer hat.

Pflege dann die aktuelle Version deiner Wortliste auf GitHub ein (inkl. dein **UML Diagramm** im **Subordner "design"**) und mach ein **initiales Commit (noch ohne Änderungen).**

Implementiere die folgenden Änderungen und verwende zumindest **5 Commits über den Entwicklungszeitraum verteilt** (nicht 5 Commits innerhalb der letzten 5 Minuten)! Kommentiere dein Programm dabei ausführlich und fange mögliche Fehler ab (kein Internet, Eingabe von Sonderzeichen etc.) und

2. Schritt: GUI

Erstelle die Oberfläche (noch ohne Funktionalität) für den Worttrainer (**mind. 3 Commits für die GUI**). Die Oberfläche soll dabei zunächst folgendermaßen aufgebaut sein und bereits alle Design-Prinzipien eines möglichst modularen Aufbaus (MVC) berücksichtigen:

Außerdem müssen noch Möglichkeiten (Buttons) für das Laden und Speichern der Wortliste vorgesehen werden.

Damit die GUI bereits für das Hinzufügen des Controllers vorbereitet ist, muss die Layout-Klasse zumindest folgende weitere Methoden aufweisen:

- Eine Methode, die das Wort im Textfeld zurückgibt
- Eine Methode, die aufgerufen wird, wenn das überprüfte Wort korrekt geschrieben war, die Anzahl der richtigen Wörter und die Gesamtanzahl der Wörter soll dabei als Parameter übergeben werden.
- Eine Methode, die aufgerufen wird, wenn das überprüfte Wort nicht korrekt geschrieben war, die Anzahl der richtigen Wörter und die Gesamtanzahl der Wörter soll dabei als Parameter übergeben werden. Unter Umständen kann die Methode auch mit einem zusätzlichen Parameter mit der vorherigen Methode zusammengefasst werden.

Hinweise:

- Ein Bild kann man in einer GUI am einfachsten über ein ImageIcon-Objekt darstellen, das in einer Swing-Komponente angezeigt wird. Ein Label bietet sich als rein passive Komponente zum Anzeigen des Bildes an, z.B.:

```
1       ImageIcon icon = new ImageIcon(new URL("..."));
2       Image image = icon.getImage(); // umwandeln in ein Image-Objekt
3       image = image.getScaledInstance(250, 250,  Image.SCALE_SMOOTH); // skalieren auf gewünschte Größe
4       lImage = new JLabel(new ImageIcon(image)); // anzeigen in einem JLabel
```

- Möglichst modular bedeutet größtmögliche Trennung in verantwortliche Bereiche, d.h. in diesem Schritt eine eigene Klasse, die von JPanel erbt und in der das Layout aufgebaut ist, und eine eigene Klasse, die von JFrame erbt und in der das Layout angezeigt wird.
- Um auch die View möglichst vom Controller zu trennen, müssen folgende Punkte erfüllt sein:
  - die GUI-Komponenten müssen - wie alle anderen Attribute auch - private sein
  - die gegebenen Methoden in der Layout-Klasse dürfen **auf KEINEN Fall die GUI-Komponenten als Parameter oder Rückgabetypen** aufweisen
  - auch sonst dürfen keine Methoden die GUI-Komponenten als Parameter übernehmen bzw. zurückgeben (keine setter- und getter-Methoden für die GUI-Komponenten).
- Erweitert Inhalt proportionales Skalieren: Es ist natürlich schöner und sinnvoller, ein Bild proportional zu skalieren, um Verzerrungen zu vermeiden. Die Höhe des Bildes sollte jedoch immer einen festen Wert haben.

3. Schritt: Controller

Implementiere anschließend einen Controller für dein MVC-Programm (**mind. 2 Commits für den Controller**), sodass dein Programm funktionstüchtig ist (inkl. Speichern/Laden, raten, Statistiken, etc.)!

4. Schritt: jar

Packe deine Abgabe in ein ausführbares JAR-Archiv **inklusive aller Quelldateien und der erzeugten Javadoc-Dateien** und lade es hier hoch.

Hinweis:

- Github hat vor kürzerem die Verwendung der Account-Passwörter zur Authentifizierung deaktiviert. Anstelle des Passwortes muss zum einloggen ein sgn. Personal Access Token verwendet werden (einfach im Passwortfeld anstelle eures Passworts eintragen). Eine Anleitung zum Erstellen eines PATs findet ihr hier: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Abgabe:

- Ein Link zu deinem git-Repository direkt als Text eingegeben
- Ein Screenshot deiner commit-History als png
- Dein jar-Archiv
- Zeig deine Abgabe deinem Lehrer bzw. deiner Lehrerin. Demonstriere dabei das starten der jar-Datei von der Konsole (Terminal, Eingabeaufforderung ...) aus.

Submission status

Submission status	Submitted for grading
Grading status	Graded
Time remaining	Assignment was submitted 62 days 9 hours late
Last modified	Monday, 16 January 2023, 9:52 AM
File submissions	Screenshot 2023-01-16 095201.png 16 January 2023, 9:52 AM Worttrainer.jar 16 January 2023, 9:52 AM
Submission comments	Comments (0)

Grading criteria

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *3 points*	Funktion vollständig *6 points*
MVC	MVC nicht umgesetzt *0 points*	MVC nur teilweise umgesetzt. *1 points*	MVC mit sauberer Trennung *2 points*

Commits	Zu wenig Commits und/oder keine sinnvollen Messages. *0 points*	Commit-Anzahl gerade ausreichend und/oder kaum sinnvolle Messages. *1 points*	Ausreichende Anzahl an Commits mit sinnvollen Messages *2 points*
JAR	JAR-Datei nicht vorhanden. *0 points*	JAR-Datei ohne Quellcode vorhanden bzw. nicht ausführbar. *1 points*	JAR-Datei inkl. Quellcode vorhanden und ausführbar. *2 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*

Feedback

Grade	16.00 / 18.00
Graded on	Thursday, 19 January 2023, 8:27 AM
Graded by	HD Haselberger David
Feedback comments	Gut gemacht!

Grade breakdown

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *3 points*	Funktion vollständig *6 points*
MVC	MVC nicht umgesetzt *0 points*	MVC nur teilweise umgesetzt. *1 points*	MVC mit sauberer Trennung *2 points*
Commits	Zu wenig Commits und/oder keine sinnvollen Messages. *0 points*	Commit-Anzahl gerade ausreichend und/oder kaum sinnvolle Messages. *1 points*	Ausreichende Anzahl an Commits mit sinnvollen Messages *2 points*
JAR	JAR-Datei nicht vorhanden. *0 points*	JAR-Datei ohne Quellcode vorhanden bzw. nicht ausführbar. *1 points*	JAR-Datei inkl. Quellcode vorhanden und ausführbar. *2 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*