Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws2021/programming-and-modelling/` zu berücksichtigen.

Abgabefrist ist der 12.11.2020 - 23:59 Uhr

Vorbereitung

Für die Abgabe der Hausaufgaben wird ein Git-Repository genutzt. Dieses wird von jedem Studierenden selbst angelegt und ist ausschließlich für den jeweiligen Studierenden sowie für die Betreuer und Korrektoren sichtbar.

Nicht oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet.

Zunächst muss ein Account auf `https://github.com/` angelegt werden (es kann auch ein bereits vorhandener Account genutzt werden). Da Git auch in der späteren Arbeitswelt ein essentielles Tool darstellt, nutzt diese Anmeldung für euch und seht von einem Spaß-Account mit Nutzernamen wie ”XxXEdg3L0rdXxX” ab.

Ohne gültigen Account kann der folgende Schritt nicht durchgeführt werden.

Meldet euch nun unter `https://classroom.github.com/a/lnJX1zvC` für diese Hausaufgabe 1 an. Folgt den dort geschilderten Anweisungen, um Zugriff auf das erstellte Repository zu erhalten. Eure Abgaben ladet ihr, wie in der Übung gezeigt, darin hoch. Es werden nur Text (.txt), Markdown (.md) oder PDF (.pdf) als Dateiformat in den jeweiligen Aufgaben verlangt. Andere Formate werden nicht akzeptiert.

Ein Abweichen von diesem Schema führt zu einer Nichtbewertung.

Aufgabe 1 - Git (30P)

In dieser Aufgabe sollen die Grundkonzepte ”Commit”, ”Branch”, ”Checkout”, ”Push” und ”Merge” des Git-Workflows kennengelernt werden. Im Verlauf der Aufgabe wird ein **Mergekonflikt** hervorgerufen, welcher aufgelöst werden soll. Führe die folgenden Schritte auf dem in der Vorbereitung erstellten Repository durch.

1. Erstelle einen Ordner mit dem Namen ”taskOne”. Erstelle innerhalb dieses Ordners eine Datei mit dem Namen ”me.txt”. Committe und pushe diese Änderung am Dateisystem.
2. Erstelle einen Branch mit dem Namen ”dev”.
3. Führe danach einen Check Out auf den ”master”-Branch durch.
4. Schreibe den folgenden Satz in die Datei:
   ”Hallo mein Name ist <Emailadresse> <Matrikelnummer> <Nachname>”. Committe und pushe die Änderung.
5. Führe einen Checkout auf den ”dev”-Branch durch.
6. Schreibe folgenden Satz in die (nun wieder leere) Datei ”me.txt”: ”Hallo mein Name ist albert@uni-kassel.de 4242 Zündorf”. Committe und pushe anschließend diese Änderung.
7. Merge nun den ”dev”-Branch in den ”master”-Branch (dies sollte einen Konflikt hervorrufen).
8. Committe die Änderung mit dem vorhandenen Konflikt in der Datei (Dies ist normalerweise nicht üblich und ein No-go, aber es verdeutlicht in diesem Fall das Problem).
9. Behebe nun den Mergekonflikt. In der Datei soll am Ende wieder folgender Satz stehen: ”Hallo mein Name ist <Emailadresse> <Matrikelnummer> <Nachname>”. Committe und pushe abschließend diese Änderung.

Nach Bearbeitung der Aufgabe sollte das Dateisystem des Repositorys wie folgt aussehen.

```
assignment-1-<GitHub-Username>/
├── .git/...
└── taskOne/
    └── me.txt
```

Der Git-Baum sollte mindestens 4 Commits beinhalten. Bei der Bewertung wird vor allem darauf geachtet, dass der Mergekonflikt unaufgelöst committet wurde und dessen Lösung erst im Nachhinein geschieht.

Zur Bearbeitung darf ein beliebiges Git-Tool genutzt werden.

Aufgabe 2 - Abstrakt vs. Konkret (30P)

Diese Aufgabe beschäftigt sich mit dem Unterschied zwischen abstrakten und konkreten Bezeichnungen sowie den Begriffen ”Abstrakt” und ”Konkret”. Hierzu wird sich mit dem in Abbildung 1 zu sehenden Wimmelbild auseinandergesetzt. Sollte das Bild zu klein sein, kann die Originaldatei aus der Quelle genutzt werden.

Quelle: `http://www.carmen-hochmann.de/Wimmelbuecher/wimmel-kassel-bugasee.jpg`

Abbildung 1: See Wimmelbild

1. Erstelle eine Tabelle mit den Spalten „Abstrakt” und „Konkret”. Trage nun 5 Beispielpaare in die Tabelle ein. Alle Paare müssen in Abbildung 1 zu sehen sein.
2. Definiere auf der Grundlage aus Aufgabenteil 1 mit eigenen Worten die Begriffe „Abstrakt” und „Konkret”.
3. Definiere den Begriff „Beispiel” mit eigenen Worten und stelle hierbei einen Bezug zu den Definitionen aus Aufgabenteil 2 her.

Lege die erstellte/n Datei/en in einem neuen Ordner mit dem Namen ”taskTwo” in deinem Repository ab. Committe und pushe die Änderungen abschließend auf den master-Branch.

Aufgabe 3 - Textuelle Szenarien (40P)

Erstelle **drei** textuelle Szenarien zu konkreten Spielsituationen des Spiels „MiniRPG“. Die Szenarien sollen in Englisch verfasst sein. Die Regeln des Spiels findest du auf der nächsten Seite vor dem Anhang.

Hinweis: Ein textuelles Szenario besteht IMMER aus einem Titel, einer Startsituation, einer Aktion sowie einer Endsituation. Diese 4 Teile sollten sichtbar (durch Farbe und/oder Absatz) voneinander getrennt sein.

Als Hilfestellung liegt ein einfaches Beispielszenario vor, das **nicht** für die Hausaufgabe verwendet/kopiert werden darf:

Title: Attack Enemy

Start:	Alice is playing ”MiniRPG”. She has 40 coins in her purse. Her Hero’s name is ”Sir Slayalot”. Her Hero has 100 out of 100 lifepoints left. The Hero’s attack is at level 3 with 15 attackpoints and his defense is at level 5 with 30 defensepoints. The Hero stands before an evil dragon with 15 attackpoints and 15 defensepoints. The dragon hisses at the hero in an attacking pose. There is no enemy after the dragon.
Action:	Alice clicks on the attack-button. The Hero strikes the dragon and the dragon strikes right back.
Result:	The Hero takes 15 LP damage. The dragon takes 15 LP damage. The dragon flinches into an defensive pose. Alice can choose her next move.

In der folgenden Hausaufgabe können die Szenarien einiger Studierenden zitiert werden. Solltest du die Verwendung deiner eigenen Szenarien nicht billigen, merke dies bitte in einer Textdatei in dieser Hausaufgabe an!

Lege die erstellte/n Datei/en in einem neuen Ordner mit dem Namen ”taskThree”in deinem Repository ab. Committe und pushe die Änderungen abschließend auf den master-Branch.

Spielregeln

This semester we organize an epic battle between man and monster. One hero faces battle and honor and wealth upon the glitt'ring quest shall be thyne.

A player can create a hero to roam dungeons filled to the brim with monsters. A Hero will start with an attack stat and a defense stat at level 1. Each dungeon contains 5 enemies that are battled in a queue. Every enemy has an attack and defense value of their own.

When an enemy is defeated, it drops a certain amount of coins the hero collects. With the coins earned, the player can level up the stats of the hero.

There will be two playing modes. The playing mode of the hero is selected when the hero is created and can not be switched afterwards.

Normal mode:

After every battle in a dungeon (aka queue of enemies), the player gets coins and is healed up to his maximum. If he dies during a battle, the stats are reset to the point before the dungeon was entered. The player gets back to the hero selection screen and can start a new dungeon.

Hard mode:

After completing a dungeon (aka queue of enemies), the hero is healed up to its maximum. If he dies during a battle, the hero is considered dead. The player gets back to the hero selection screen but can no longer start a new dungeon with the dead hero.

Combat:

The combat in this game is round based. The enemy acts after the hero. At the beginning of a combat round, the enemy switches into an attacking or defensive pose. This indicates what the enemy will do next.
The player can choose one of two options. Based on the stats of the hero, the player can attack or block.
When the player chose his action and the enemy made his response, the enemy takes a new random pose and the battle begins anew.

Anhang

Es folgt eine Auflistung hilfreicher Webseiten und weiterer Erklärungen zu den Themen dieser Hausaufgabe. Die Links sind als Startpunkt zur selbstständigen Recherche gedacht. Das Durcharbeiten der folgenden Quellen ist kein bewerteter Anteil der Hausaufgaben.

Git

- Download: `https://git-scm.com/downloads`
  - Hilfestellung für Linux:
    `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`
  - Hilfestellung für Mac:
    `https://gist.github.com/derhuerst/1b15ff4652a867391f03#file-mac-md`
- Documentation: `https://book.git-scm.com/doc`
  - What is Version Control?:
    `https://book.git-scm.com/video/what-is-version-control`
  - Pro Git: `https://book.git-scm.com/book/en/v2`
- A Visual Git Reference: `http://marklodato.github.io/visual-git-guide/index-en.html`
- Git Cheat Sheet:
  `https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf`

GitHub

- GitHub: `https://github.com/`
- Getting started with GitHub:
  `https://help.github.com/en/categories/getting-started-with-github`
- Beispiel für ein GitHub-Repository, schaut euch z. B. Dateien, Commits, Branches, Issues und Pull requests an: `https://github.com/deepfakes/faceswap`