

Hausaufgabe 1

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/ws2223/programming-and-modelling/> zu berücksichtigen.

Abgabefrist ist der 03.11.2022 - 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 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, nutze diese Anmeldung für dich und sieh von einem Spaß-Account mit Nutzernamen wie „XxXEdg3L0rdXxX“ ab.

Ohne gültigen Account kann der folgende Schritt nicht durchgeführt werden.

Melde dich nun unter <https://classroom.github.com/a/qJIzpHaC> für die Hausaufgabe 1 an. Folge den dort geschilderten Anweisungen, um Zugriff auf das erstellte Repository zu erhalten. Deine Abgaben lädst du, 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 und folglich mit 0 Punkten gewertet.

Eine Registrierung für die Hausaufgaben ist zwingend notwendig.

Falls du dich noch nicht in dem Google Form (<https://forms.gle/wfWWuV2vmTsrDN2U7>) angemeldet hast, hole dies jetzt unbedingt nach. Deine Hausaufgaben können ohne Anmeldung nicht bewertet werden!

Ein Abweichen von diesem Schema führt zu einer Nichtbewertung.

Hausaufgabe 1

Aufgabe 1 - Git (10P)

In dieser Aufgabe sollen die Grundkonzepte „Commit“, „Branch“, „Checkout“, „Push“ und „Merge“ des Git-Workflows erlernt werden. Im Verlauf der Aufgabe wird ein **Mergekonflikt** verursacht, welcher aufgelöst werden soll. Führe die folgenden Schritte auf dem Repository, welches du in der **Vorbereitung** erstellt hast, aus.

1. Erstelle einen Ordner mit dem Namen „task1“. Erstelle innerhalb dieses Ordners eine Datei mit dem Namen „me.txt“. Committe und pushe diese Änderung.
2. Erstelle einen Branch mit dem Namen „dev“.
3. Führe danach einen Checkout auf den „main“-Branch durch (damit wechselst du zurück auf den „main“-Branch).
4. Schreibe den folgenden Satz in die Datei:
„Hallo mein Name ist <Name> und ich befinde mich auf dem main-Branch!“. 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 Zündorf und ich befinde mich auf dem dev-Branch!“. Committe und pushe anschließend diese Änderung.
7. Führe einen Checkout auf den „main“-Branch durch.
8. Merge nun den „dev“-Branch in den „main“-Branch (dies sollte einen Konflikt hervorrufen).
9. 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).
10. Behebe nun den Mergekonflikt. In der Datei soll am Ende wieder folgender Satz stehen: „Hallo mein Name ist <Name> und ich befinde mich auf dem main-Branch!“. Committe und pushe abschließend diese Änderung.

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

```
pmws2223-assignment-1-<GitHub-Username>/
├── .git/...
├── task1/
│   └── 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 (8P)

Diese Aufgabe beschäftigt sich mit dem Unterschied zwischen abstrakten und konkreten Bezeichnungen sowie den Begriffen „Abstrakt“ und „Konkret“. Betrachte dafür das Wimmelbild aus Abbildung 1. Sollte das Bild zu klein sein, kann die Originaldatei aus der Quelle genutzt werden.

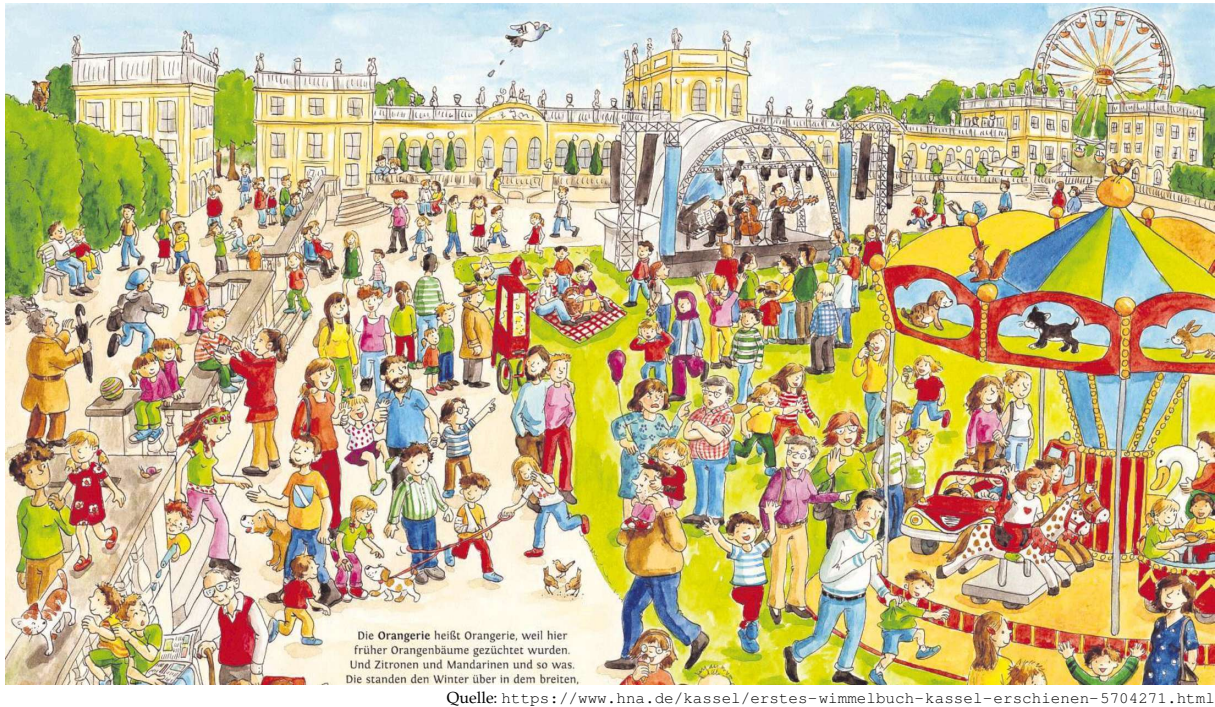


Abbildung 1: Orangerie Wimmelbild

1. Erstelle eine Tabelle mit den Spalten „Abstrakt“ und „Konkret“. Trage nun **fünf** 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 einen neuen Ordner mit dem Namen „task2“ in deinem Repository ab. Committe und pushe die Änderungen abschließend auf den [main](#)-Branch.

Aufgabe 3 - Textuelle Szenarien (12P)

Erstelle **drei** textuelle Szenarien zu konkreten Spielsituationen des Spiels „PMon“. 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 vier 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: Attacking the enemy

Start: Karli is playing „PMon“.

Karli's current monster is Firebear of type fire.

The Firebear has a Fireball ability of type fire.

The enemy has the type ice and has 15 health points left.

Action: Karli tells her monster to attack the enemy with Fireball.

Result: The attack is strong against the enemy and its health drops to zero. Firebear gains experience. The encounter is over.

Lege die erstellte(n) Datei(en) in einem neuen Ordner mit dem Namen „task3“ in deinem Repository ab. Committe und pushe die Änderungen abschließend auf den [main](#)-Branch.

Spielregeln

This semester we take a stroll down memory lane, back to simpler times where the most important knowledge was how to level up your monsters and if you could defeat your enemies.

We will call the game PMon.

Encounter

A player can enter an encounter with up to three monsters fighting against up to three other monsters. The combat in this game is round based and the monsters always fight 1 vs 1. The player always acts prior to the enemy. The player can either attack the enemy with an ability of the current monster or the player can change his/her current monster. The enemy attacks with a random ability of its current monster.

If a player defeats all the enemies in an encounter the game can be stopped or a new encounter can be started. Either way all monsters of the player get healed to their maximum health points after clearing an encounter.

Monsters

A monster always starts with level 1 and 0 experience. When the player defeats a monster the player's current monster gains experience. If a monster's experience reaches a threshold the monster gains a level and the experience is reset to 0. With the level up the monster can unlock new abilities and it gets stronger via more maximum health points. A monster can only have a maximum of 3 abilities.

Ability and Monster Types

Monsters and abilities have an elemental type. Each type is strong or weak against other types, except the „normal“ type, which has no special weakness or strength. The strengths and weaknesses are displayed in figure 2. An arrow from type A to another type B means A is strong against B and B is weak against A. For example: Fire is strong against plant, but weak against earth. Meanwhile water is strong against earth and weak against ice. Normal is neither weak nor strong against other types. Strong attacks deal more damage and weak attacks deal less damage.

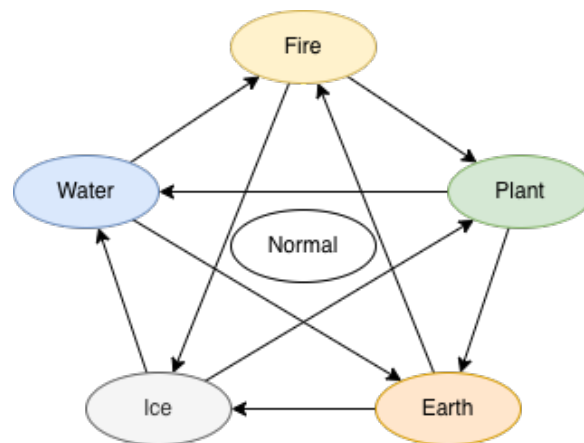


Abbildung 2: Ability and Monster Types, Strengths and Weaknesses (example)

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/TestFX/TestFX>