# Beleg Slither-Buddy - Bug Fixing and Finishing

## Software Engineering II

Gruppe A
Bauhaus-Universität Weimar

21. January 2024

# Hilfreiche Themen

## Kommentare

Lecture 6: Slides 42 - 45.
Durch die Zusammenarbeit mehrerer Teams bei den Projekten und den regelmäßigen Austausch des Codes untereinander, war eine ausreichende Dokumentation durch Kommentare welche den Code verständlicher machen sehr von Vorteil. An dieser Stelle haben wir teilweise durchaus zu wenig kommentiert und haben entsprechende Beschwerden von den Teams, welche unseren Code verstehen mussten, bekommen. Ein besonderer Aspekt den wir aus den Folien mitnehmen werden ist, dass man die Kommentare wirklich parallel zum Code schreiben sollte und nicht erst im Nachhinein, da dies meist nicht wirklich funktioniert. Ein gut kommentierter Code ermöglicht es Entwicklern, welche Code eines anderen Teams verstehen wollen, nachzuvollziehen warum Features so implementiert wurden wie es der Fall war und welche Funktonalitäten Methoden aufweisen ohne das der Körper dieser komplett durchgegangen werden muss.

## Requirements representation

Lecture 4: Slides 29 - 47 Use Cases
Im Nachhinein betrachtet, hätten wir für die Formulierung unserer Requirements noch eine andere Form der Darstellung hinzu nehmen sollen als nur Natural Language, da wir von der Nachfolgergruppe ein paar Beschwerden zur Eindeutigkeit bekommen haben. Für diese zweite Form hätten sich wohl ein Use Case Diagramm angeboten, um besser zu zeigen wie die Interaktion zwischen User und Software verlaufen soll oder auch um die formulierten Requirements übersichtlicher darzustellen.

# Wünschenswerte Themen

## Aufsetzten von Projekten

Am meisten Probleme hat uns bei der Arbeit an den Projekten das nötige Setup gemacht. Bei fast allen Belegen bei denen man Code schreiben musste, hatten wir am Anfang Schwierigkeiten das Programm überhaupt erst mal zum Laufen zu bekommen. Oft gab es Fehler durch nicht vorhandene Libraries oder ungesetzte Umgebungsvariablen, welche wir nur mit großem Zeitaufwand beheben konnten. Im Fall von JUnit kamen wir sogar zu gar keiner Lösung wodurch wir unsere eigenen Unit Tests implementieren mussten. Obwohl uns zu den verschiedenen genutzten Tools Tutorialvideos bereitgestellt wurden, traten beim bearbeiten der Belege fast nur Fehler auf die in den Videos nicht adressiert wurden bzw. nicht vorkamen. Wegen dieser Schwierigkeiten wäre es schön gewesen, wenn man sich mehr mit den Hintergründen der genutzen Tools auseinander gesetzt hätte, damit man auftretende Probleme ohne stundenlange Nachforschungen selber hätte lösen können.

## Eleganten Code schreiben

Mehr konkrete Beispiele, wie bei der Implementierung eleganter Code geschrieben werden kann. Zum Beispiel unter der Einschränkungen keine zu hohe Verschachtelung unter Schleifen und Gabelungen zu haben aber trotzdem keine break oder continue Anweisungen zu nutzen und den Code dabei lesbar und übersichtlich zu behalten.