Fachinformatiker für Anwendungsentwicklung

Dokumentation zu Java-Projekt

Weltraumsimulation

Entwicklung einer Simulation nach Vorgaben mit Eigeninitiative

Abgabedatum: Bremen, den 21.04.2024

Teilnehmer:

Dennis Banasch

fi23a

Ausbildungsbetrieb:

cbm GmbH

Wegesende 3-4

28195 Bremen

Betreuer:

Sven Lilienthal

1.1 Projektvorgaben

Die Projektvorgabe war es eine Weltraumsimulation in Java zu programmieren, um dabei das gesammelte Wissen aus dem Unterricht in der Praxis anwenden zu können. Dabei sollten die Projektvorgaben des Dozenten eingehalten werden. Darüber hinaus hatten die Teilnehmer auch die Möglichkeit Eigeninitiative zu zeigen und mehr zu leisten, als gefordert war. Der zeitliche Projektrahmen umfasste dabei eine Woche inklusive der Wochenendtagen.

1.2 Projektziel

Ziel war es, eine funktionierende Weltraumsimulation in Java zu programmieren. Der Spieler sollte dabei zwischen verschiedenen Planeten und Monden reisen können, sowie Schiffe bauen, Ressourcen sammeln und Gegner bekämpfen können. Weitere besondere Funktionalitäten umfassen unter anderem das Speichern des Spielstandes, sowie einer täglichen Versionierung über git.

1.3 Projektbegründung

Das Projekt erschien für den Teilnehmer am Herausforderndsten, da er auch so sein gelerntes Wissen aus dem Unterricht nicht nur weiter vertiefen konnte, sondern sich auch neuen Herausforderungen stellen konnte, ähnliches aus dem Unterricht nochmal in einem Projekt, aber größer und umfangreicher umsetzen zu können.

2.1 Idee und Planung

Das Projekt wurde zuerst geplant, es sollte eine sinnvolle Klassen- & Vererbungshierarchie geschaffen werden. Dabei stand die Erweiterbarkeit im Vordergrund. Das Projekt sollte auch nach Abschluss des Projekts gut erweiterbar, sowie auch wartbar sein. Java bietet durch seine festgelegten Strukturen der Objektorientierung die Möglichkeit relativ früh und geschickt sein Programm in Klasen einzuteilen, sodass eine Datenkapselung relativ gut möglich ist.

Die Vorgaben des Projekts sollten erfüllt werden. So wurden relativ schnell Klassen zu Objekten erstellt, die als Reisepunkte gelten sollten (Planten & Monde). Auch die geforderten Sonnenklassen inklusive ihrer Unterarten in Form von Subklassen wurden dem Projekt hinzugefügt. Diese sollten in jedem Sonnensystem präsent sein. So wurde auch eine Klasse für ein Sonnensystem erstellt. Viele Objekte besitzen dabei unterschiedliche Methoden und

Eigenschaften, die teilweise schon vorher definiert wurden, teilweise bei Bedarf aber auch erst nach implementiert werden mussten, um die Struktur des Codes auf Recht zu erhalten.

Die grundsätzliche Idee war es erstmal viele Klassen & Subklassen anzulegen, welche in verschiedenen Paketen je nach Zugehörigkeit gruppiert wurden. So wurden Pakete mit Himmelskörpern erstellt, welche alle Elemente beinhalten, die diesbezüglich eine sinnvolle Gruppierung bilden können.

2.2 Abweichung vom Projekt

Die Simulation umfasst zusätzliche Funktionen, wie das Hinzufügen von Waffen zu Raumschiffen oder das Eintreten von bestimmten Aktionen, wenn der Spieler einen Planeten betritt und dieser keinen besonderen Anzug anhat. Da diese Funktionen von Anfang des Projekts mitgeplant wurden, hat dies das Arbeiten mit den Klassen beeinflusst, da immer die zusätzlichen Funktionalitäten mitberücksichtigt werden mussten. Dies hatte zur Folge, dass das Projekt inhaltlich einen sehr großen Rahmen angenommen hatte. Später dazu mehr im Fazit.

3.1 Erstellung von Diagrammen

Die in den Anlagen[1] beigefügten Diagramme selbst wurden relativ spät im Projekt erstellt, da immer wieder neue Klassen und Funktionalitäten hinzukamen, sodass Konzepte auch teilweise wieder verworfen oder nicht vollends durchdacht wurden. Dennoch ist eine besondere Struktur vorhanden. Es gibt Klassen, die eine Object-Endung haben, unter welcher alle Kassen des zugehörigen Pakets oder Unterpakets zugeordnet sind. Gruppierungen sind oft sinngemäß gewählt wurden. Gruppenähnliche Klassen wurden zu Paketen zusammengefasst und untergebracht, sodass eine klare hierarchische Struktur ersichtlich ist.
Das Projekt selbst hat einen großen Rahmen angenommen und viele Klassen besitzen sehr viele Methoden, weswegen sich der Teilnehmer für die Übersichtlichkeit für Paketdiagramme entschieden hat.

4.1 Benutzeroberfläche

Der Spieler wird am Anfang des Programms begrüßt, jedoch ist zu Beginn der Simulation, vorausgesetzt der Spieler hat nicht schon mal bis zu einem gewissen Punkt weitergespielt, keine Welt vorhanden. Der Spieler wird gebeten dem Universum und seiner Galaxie einen Namen zu geben. Dann kann er entscheiden, wie viele Sonnensysteme er erstellen möchte. Dabei müssen minimal eins und maximal drei Sonnensysteme erstellt werden. Danach kann der Spieler den Sonnensystemen einen Namen seiner Wahl geben. Nun werden die Sonnen, Planeten & Monde im Hintergrund zufällig erstellt und den jeweiligen Sonnensystemen zugeordnet. Der Spieler kann einen einmaligen Blick in die Informationen der erstellten Objekte einsehen, wenn er möchte. Diese Funktion, ursprünglich zum Debuggen vorgesehen, wurde als Feauture eingebaut, um Spielern, die nicht erst alles erforschen wollen, die Möglichkeit zu geben, alle interessanten Orte im Vorfeld zu wissen. Danach wird der Spieler selbst in der Welt erstellt. Der Spieler vergibt seinen Namen und wird dann gefragt, in welches der von ihm erstellten Sonnensysteme er reisen möchte. Dies ist eine wichtige Entscheidung, denn man kann ein System danach nicht mehr verlassen, wenn man sich entschieden hat.
Anschließend wird gefragt, ob er seinen Fortschritt speichern möchte. Dies ist eine der drei vorgesehenen Schlüsselpunkte, wo man sein Spiel speichern kann. Der Spieler kann danach jeder Zeit seinen Spielstand laden ohne immer wieder den Erstellungsprozess des Universums durchlaufen zu müssen.

Irgendwann ist der Spieler in der Hauptmenüführung angekommen. Er wurde zufällig einem Planeten mit einer normalen Atmosphäre, welcher um einen Hauptsequenz-stern kreist, zugeordnet. Dabei sind setzt er am Anfang in einem Kampfraumschiff. Ab hier hat der Spieler mehrere Optionen. Er kann sich z.B. dazu entscheiden, die Charakterinformationen abzufragen, wo ihm sein Name, seine Werte, sein Inventar und seine besuchten Orte angezeigt werden. Eine weitere Möglichkeit ist die Abfrage der Ortsinformationen. Dort kann er sich die Informationen über seinen aktuellen Ort ansehen. Diese Option kann aber erst aufgerufen werden, wenn der Spieler den Planeten betreten hat. Eine weitere Option ist das Aufrufen der Eigenschaften des Raumschiffs, im welchem sich der Spieler befindet. Auch dort werden Werte, wie Stärke, Lebenspunkte usw. angezeigt.
Der Spieler kann sich aber auch dazu entscheiden sein Raumschiff zu verlassen und seinen lokalen Ort, kann Planet oder Mond sein, zu betreten. Dort würde er dann zu einer Untermenüführung weitergeleitet werden. Bei dieser Option gibt es jedoch zu bedenken, dass einige Planeten eine verschmutze oder toxische Atmosphäre besitzen und dies Auswirkungen auf den Spieler haben könnte, wenn er den Planeten mit seinem Standard-Anzug betritt.

Wenn dem Spieler sein jetziger Ort nicht gefällt, dann kann er auch zu einem anderen Ort reisen, wobei es nicht garantiert ist, dass er heile ankommt, da potenzielle Gegner ihn überraschen könnten. Auch hier kann der Spieler sein Spiel speichern und das immer, wenn er sich in dieser Menüführung befindet.

Wenn der Spieler sich dazu entscheidet aus dem Planeten auszusteigen, dann wird er in eine Untermenüführung weitergeleitet, wo er sich immer noch die Charakter- & Ortsinformationen anzeigen lassen kann, aber er kann auch Ressourcen abbauen und diese seinem Inventar hinzufügen. Die Ressourcen auf Planeten und Monden sind begrenzt.

Wenn der Spieler sich stattdessen dazu entschieden hat einen anderen lokalen Ort zu besuchen muss er mit seinem Raumschiff dort hinreisen. Wie bereits erwähnt könnten, kann er dabei jeder Zeit von feindlichen Schiffen angegriffen werden. Er kann sich dann dazu entscheiden zu fliehen, wo die Wahrscheinlichkeit aber gering ist, dass es funktioniert, oder er kann versuchen mit den Gegnern zu kämpfen. Dabei wird zufällig entschieden wer anfängt. Die beiden Parteien wechseln sich dann in einem rundenbasierten Kampfsystem abwechselnd ab und können sich gegenseitig Schaden zu fügen. Die Gegnerschiffe sind allerdings nicht immer von der selber Sorte. So sind einige Schiffe mit Waffen ausgerüstet, die ihnen Vorteile im Kampf verschaffen. Außerdem ist es so, dass wenn das Schiff des Spielers zerstört wird, er automatisch tot ist und das Spiel beendet wird. Das liegt daran, dass sich der Spieler innerhalb des Schiffs befand, als er gestorben ist. Der Spieler kann aber auch außerhalb des Schiffs sterben, wenn er z.B. toxische Atmosphäre einatmet.

4.2 Entwicklung

Die Benutzerführung für den Nutzer verbirgt die Hintergrundprozesse, die im Hintergrund laufen. Zum einem werden die zufälligen Planeten, Sonnen und Monde aus Dateien herausgelesen. Das bedeutet, dass dort Daten herausgelesen werden und basierend auf diesen Daten die Objekte erstellt und jeweils zugeordnet werden. Diese Daten basieren aber nicht auf realistischen Daten, sondern wurden vom Open Source AI-Tool ChatGPT[2] erstellt.
Der zweite besondere Hintergrundprozess ist das Speichern des Spielstands. Dieser wird über das Serialize-Interface realisiert, sodass die Daten der Instanzen der Objekte immer gespeichert und geladen werden können.

Das Programm selbst wird in sehr viele Klassen ausgelagert, um gewisse Methoden nicht zu überladen. So wurden sogenannte Orakel-Klassen erstellt, um Methoden auszulagern. Dabei sind diese Orakel von überall zugreifbar, da sie als Konstanten in der game-Methode deklariert sind. Dies hat den Vorteil, dass man nicht alle Methoden innerhalb der Orakel als statisch festlegen muss, sondern man kann sie über die Konstanten aufrufen.

Die Menüführung findet in zwei Haupt-Loops statt. Einmal dass Hauptmenü, welches ausgeführt wird, wenn der Spieler sich in seinem Raumschiff befindet und dann noch das Local-Menü, welches aufgerufen wird, wenn der Spieler ausgestiegen ist. Andere nennenswerte Menüs sind das Reise-Menü oder das Kampfmenü, welche alle eigenen Orakelklassen bekommen haben.

Fazit

5.1 Soll-Ist-Vergleich

Die Anforderungen des Projekt wurden nur teilweise erfüllt, da das Projekt relativ schnell einen großen Umfang angenommen hat, der zeitliche Rahmen von nur einer Woche aber nicht daraus ausgelegt war. Dementsprechend sind einige Funktionalitäten, wie das Crating-System, das Bau-System, das Transportieren von Ressourcen usw. nicht vorzufinden.

5.2 Gewonnene Kenntnisse

Dieses Projekt hat mit sehr gefallen. Auch wenn ich mit dem abgegebenen Ergebnis nicht wirklich zufrieden war, da die zeitlichen Umstände dies nicht ermöglicht haben, habe ich einiges über Programmierung gelernt und dass man seine Fantasie manchmal doch zurückhalten muss, wenn man gewünschte Ergebnisse erreichen möchte. Das Programmieren selbst hat meine Fähigkeiten auf einen neuen Horizont erweitert, da ich mit Java, einer höheren Programmiersprache, eine neue Art zu denken erlernt habe.

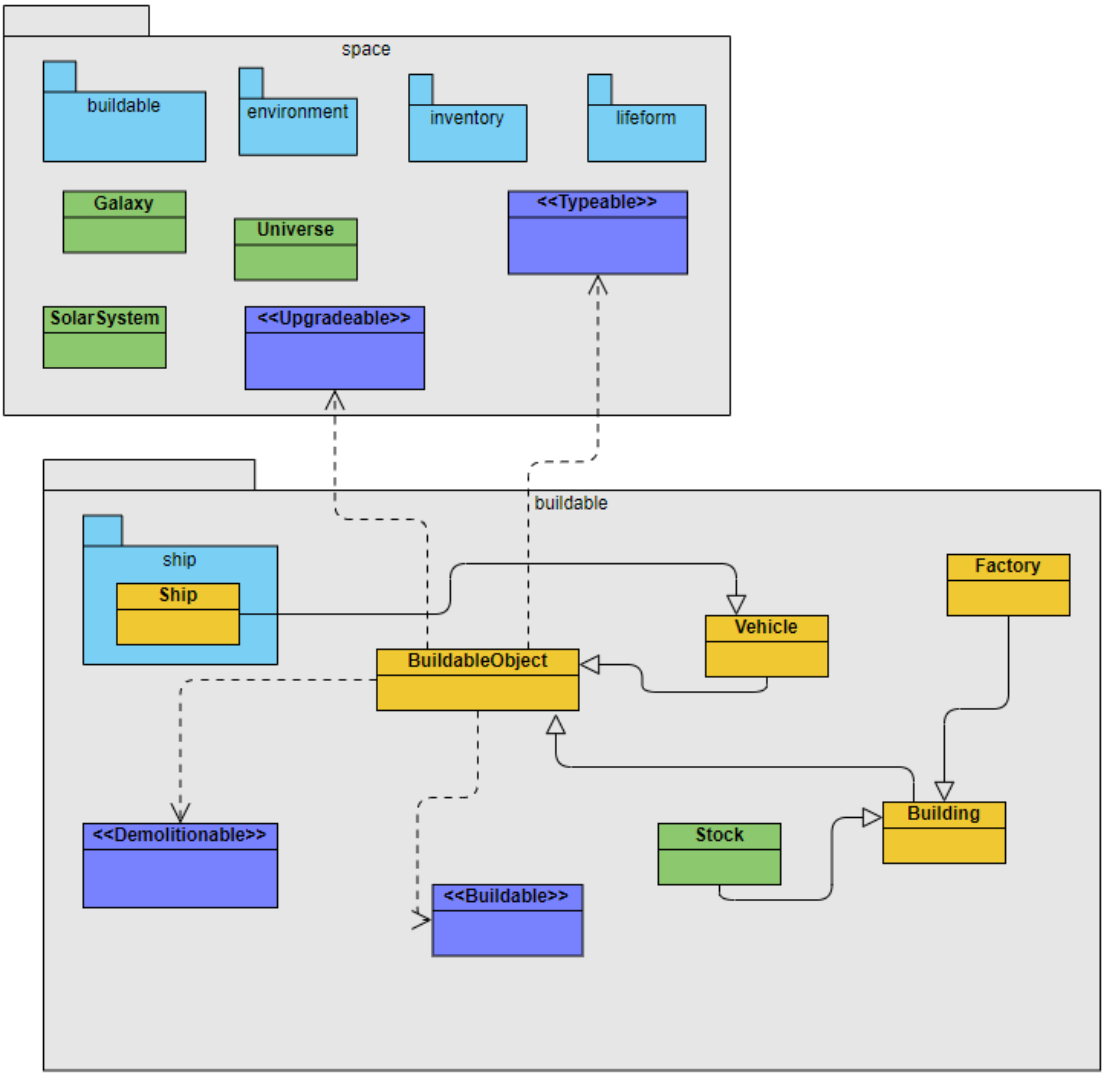
5.1 Soll-Ist-Vergleich

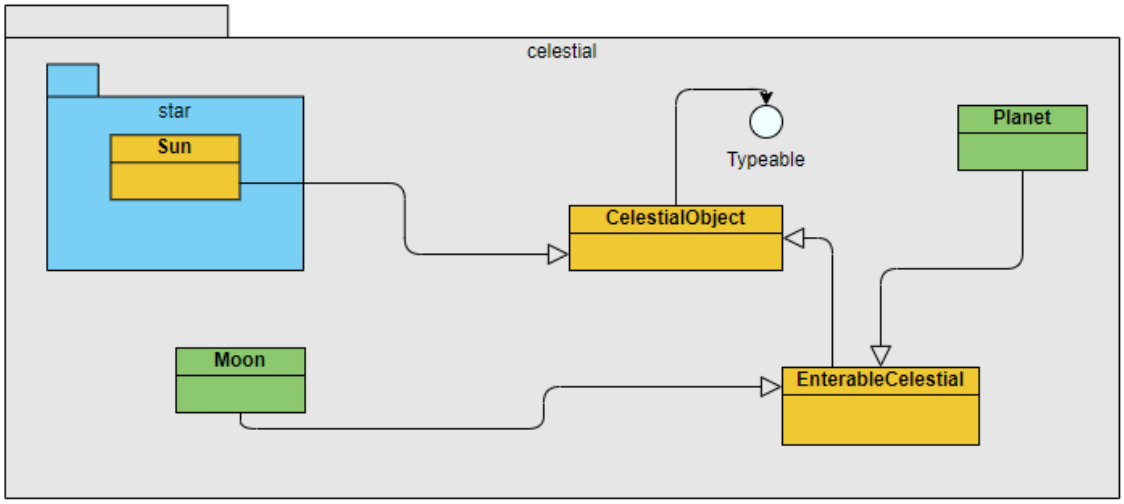
Die Anforderungen des Projekt wurden nur teilweise erfüllt, da das Projekt relativ schnell einen großen Umfang angenommen hat, der zeitliche Rahmen von nur einer Woche aber nicht daraus ausgelegt war. Dementsprechend sind einige Funktionalitäten, wie das Crating-System, das Bau-System, das Transportieren von Ressourcen usw. nicht vorzufinden.

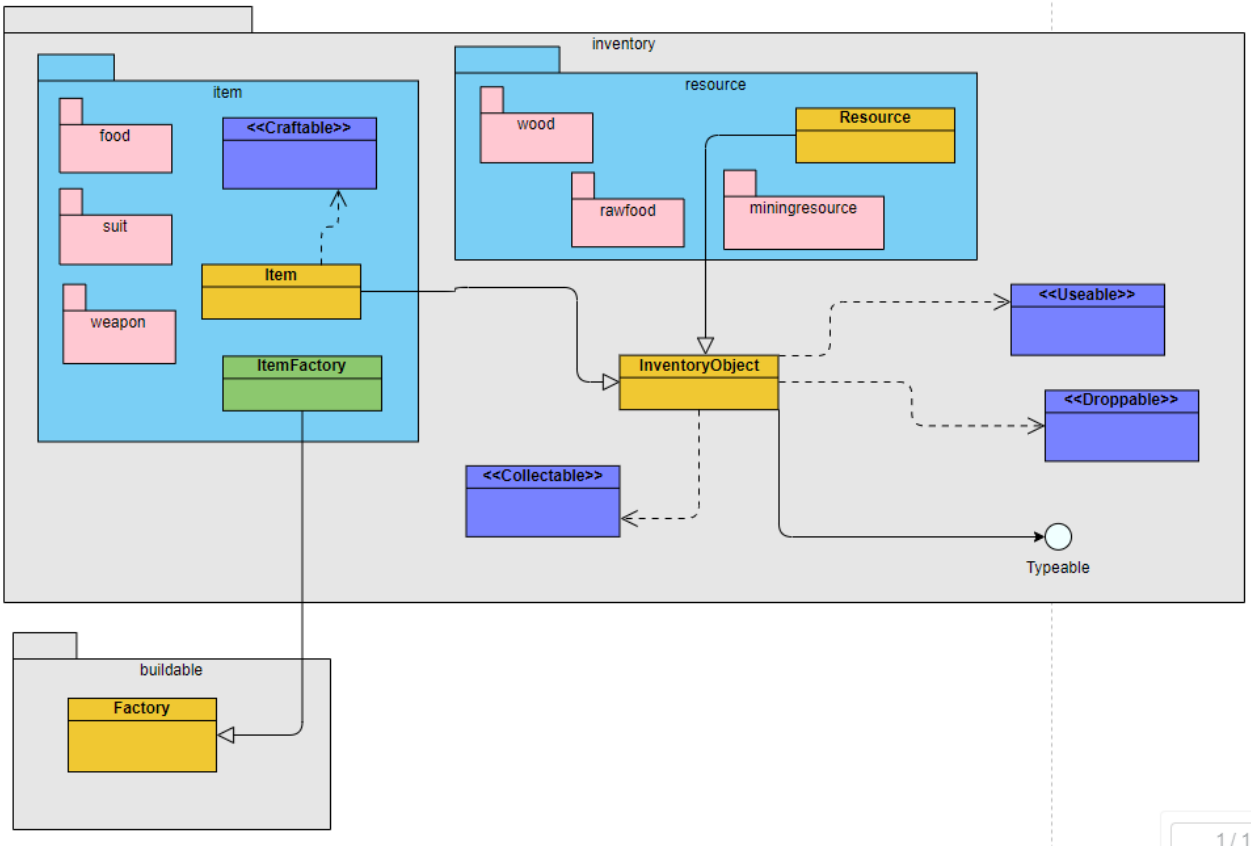
Quellenverzeichnis

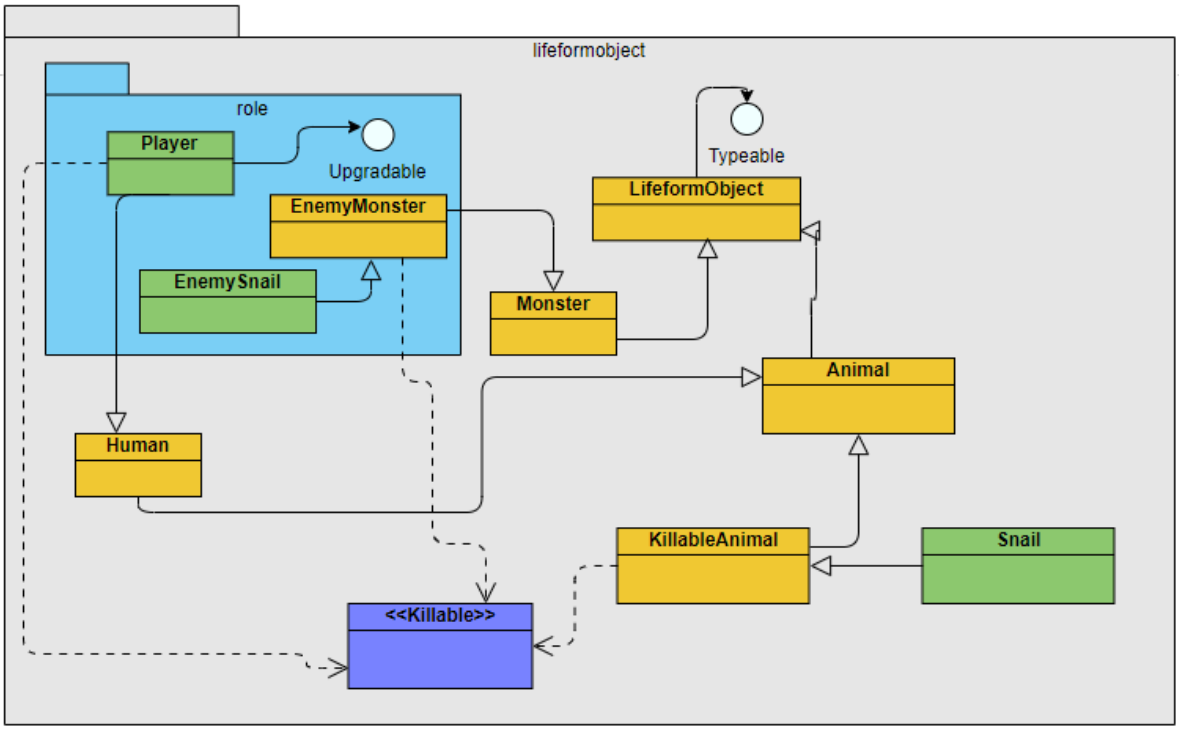
1. https://online.visual-paradigm.com/de/
2. https://chat.openai.com/

Paketdiagramme






inventory
item
food
<<Craftable>>
suit
Item
weapon
ItemFactory
resource
wood
Resource
rawfood
miningresource
InventoryObject
<<Useable>>
<<Droppable>>
<<Collectable>>
Typeable
buildable
Factory


lifeformobject
role
Player
Upgradable
EnemyMonster
EnemySnail
Typeable
LifeformObject
Monster
Animal
Human
KillableAnimal
Snail
<<Killable>>