Praktikumsaufgabe 7

Thema Modellierung Refaktoring Abstrakte Klassen

Lernziele

1. Das Klassenmodell aus Aufgabe 6 nach Anleitung refaktorisieren.
2. Das Wissen über abstrakte Klasse und deren Methodenkategorien anwenden.
3. Ein interaktives Spiel entwerfen und die Welt konfigurieren.
4. Ein bestehendes Modell verwenden.

Hinweis

1. Sie sollen in Aufgabe 7.4 ein Spiel entwerfen. Darin soll von den Gegenständen mindestens die Kombination Waffe, Zaubertrank und Wertgegenstand enthalten sein. Sie dürfen natürlich auch alle Gegenstände verwenden. Wenn Sie sich für eine Teilmenge der Gegenstände entscheiden, müssen Sie nur die für diese Teilmenge erforderlichen Methoden implementieren.
2. In der Klasse Spiel sind bereits alle neuen Befehle implementiert. Dort werden bereits Typkonvertierungen vorgenommen. Ebenfalls enthalten ist eine minimale Behandlung von Eingabefehlern.

7.0 Überblick

Das Spiel aus Aufgabe 6 soll interaktiv werden. Folgende Interaktionen zwischen Spielerin und den Gegenständen bzw. den Bewohnern der Räume sollen möglich sein:

1. Spielerin:
    a. Die Spielerin kann Monster mit einer Waffe attackieren. Dadurch verliert das Monster so viel Lebenspunkte, wie die Waffe Schlagkraft hat. Eine Waffe wird nach einem Einsatz erst wieder nach dem 4'ten Versuch nutzbar. (Befehl „attack")
    b. Die Spielerin kann Monster mit einem Zaubertrank, der negative Wirkung hat, attackieren. Dadurch verliert das Monster Lebenspunkte und der Zaubertrank an Wirkung. (Befehl „attack")
    c. Die Attacken erfolgen aus einem Nachbarraum des Monsters. Dazu kann sich die Spielerin über den Inhalt eines Nachbarraums informieren, um zu prüfen ob und welche Gefahr dort lauert. (Befehl „look_ahead")
    d. Die Spielerin kann durch das Trinken eines Zaubertranks ihre Lebenspunkte erhöhen. Der Zaubertrank verliert danach an Wirkung. (Befehl „drink")
    e. Die Spielerin kann mit einem Teleporter mehrere Räume überwinden (KEINE DIAGONAL BEWEGUNGEN!). Die Grenze für die Strecke ist die

Reichweite des Teleporters. Der Teleporter verliert danach an Reichweite. (Befehl „teleport“)

f. WENN der Zaubertrank oder der Teleporter wirkungslos geworden sind, DANN werden diese Gegenstände nicht weiter im Rucksack mitgeschleppt. (Methode: `pruefe_gebrauchswert` gegeben)
g. Die Spielerin kann einen Teleporter an einer Tankstelle auftanken, wenn Spielerin, Teleporter und Tankstelle im gleichen Quadranten sind. Dadurch sinkt der verfügbare Kraftstoff an der Tankstelle. (Befehl „fuel“)
h. Die Spielerin kann Gegenstände (Wertgegenstände) aufnehmen und an einen Zielort transportieren. (Befehl „take“ bereits implementiert)
i. Die Spielerin kann sich zu jedem Zeitpunkt einen Überblick über die Welt verschaffen. (Befehl „overview“ bereits implementiert)

2. Bewohner:
   a. Ein Monster attackiert die Spielerin, wenn diese den Raum betritt. Die Spielerin verliert so viel Lebenspunkte, wie das Monster Effekt hat. Das Monster verliert ebenfalls Effekt-mal Lebenspunkte.
   b. Ein Freund hilft der Spielerin, wenn diese den Raum betritt. Die Spielerin gewinnt so viel Lebenspunkte, wie der Freund Effekt hat. Der Freund verliert ebenfalls Effekt-mal Lebenspunkte.
3. Spiel und Welt:
   a. Nach jedem Spielzug prüft das Spiel, ob das Spiel beendet ist. Das ist der Fall, wenn die Spielerin das Abenteuer nicht überlebt hat, oder das Spielziel erreicht ist. Die Gewinnsituation für das Spiel zu definieren ist Ihre Aufgabe.
   b. Nach jedem Spielzug wird die Welt aufgeräumt und alle nicht mehr aktiven oder unbrauchbaren Gegenstände werden aus den Räumen entfernt.

7.1 Abstraktionen

Aus dem Text in 7.0 lassen sich mit einiger Überlegung folgende Klassen und Beziehungen ableiten.

1. Monster und Freunde sind Bewohner.
2. Bewohner und Spielerin sind Akteure.
3. Akteure und Gegenstände sind Weltobjekte.
4. Akteure sind Ressourcen-Inhaber.
5. Ressourcen-Inhaber haben eine Ressource (z.B. Lebenspunkte, Liter, Strecke in Quadranten). Die Ressource kann erhöht oder reduziert werden.
6. Waffen, Zaubertränke und Bewohner sind Effektoren.
   a. Wenn ein Effektor zum Einsatz kommt, dann wird der Effekt auf einen Ressourcen-Inhaber angewendet und erhöht bzw. reduziert die Ressource.

   b. Der Effekt sind die Lebenspunkte, die ein Akteur gewinnt / verliert, wenn die Gegenstände zum Einsatz kommen. In unserem Spiel kommen diese z.B. bei einem Angriff der Spielerin zum Einsatz.
   c. Der Effekt der Bewohner sind die Lebenspunkte, die sie der Spielerin beim Betreten des Raumes nehmen oder schenken.
7. Tankstellen, Teleporter und Zaubertränke sind Gegenstände mit Ressourcen.
   a. Zaubertränke sind Ressourcen-Inhaber, da sie einen Teil ihrer Wirkung verlieren, wenn sie zum Einsatz kommen.
   b. Teleporter und Tankstellen sind Ressourcen-Inhaber, da sie Kraftstoff verlieren, wenn sie zum Einsatz kommen.
8. Gegenstände mit Ressourcen sind Ressourceninhaber und Gegenstände

7.1 Aufgabe

Sie sollen bei dieser Aufgabe noch keine zusätzliche Funktionalität implementieren!

1. Entwickeln Sie aus den Abstraktionen zunächst ein Klassenmodell auf dem Papier. Überlegen Sie auch welche der Klassen konkret und welche abstrakt sind.
2. Implementieren Sie das Klassenmodell mit den Vererbungsbeziehungen.
3. Refaktorisieren Sie die Gemeinsamkeiten, indem Sie diese in die passenden Superklassen auslagern.

7.2 Erweiterung des Modells um Funktionen

Jetzt erweitern wir das Modell um Funktionen, so dass Monster und Freunde mit der Spielerin interagieren und die Spielern die Gegenstände im Rucksack anwenden kann.

Dabei lässt sich die Implementierung jeder Form von Interaktion der Akteure und Gegenstände auf die Methoden der Ressourcen-Inhaber und Effektoren zurückführen.

7.2.1 Weltobjekte und Akteure

Nach jedem Spielzug sollen alle verbrauchten Gegenstände und die toten Akteure aus der Welt gelöscht werden. Dazu führen wir in der Klasse Akteur die Methode `lebendig?` und in die Klasse Gegenstand die Methode `verbraucht?` ein. Ein Akteur ist lebendig, wenn seine Ressource nicht verbraucht ist. Ein Gegenstand ist verbraucht, wenn seine Ressource verbraucht ist.
Damit die Welt beim Aufräumen nicht zwischen Gegenständen und Akteuren unterscheiden muss, führen wir für die Weltobjekte eine Methode `aktiv?` ein. Nicht aktive Gegenstände werden aus der Welt gelöscht. Ein Akteur ist aktiv, wenn er lebendig ist. Ein Gegenstand ist aktiv, wenn er nicht verbraucht ist.

7.2.1 Aufgabe

Implementieren Sie die Methoden `lebendig?`, `verbraucht?` und `aktiv?` in den geeigneten Klassen.

7.2.2 RessourceInhaber und Ressource

Ein Ressourcen-Inhaber hat Methoden um die Ressource zu erhöhen und zu reduzieren (*reduziere_ressource, erhoehe_ressource*). Dies erledigt der Ressourcen-Inhaber, indem er *reduziere / erhoehe* auf seiner Ressource aufruft. Da die Art der Ressource vom konkreten Inhaber abhängt, hat der Ressourcen-Inhaber eine **abstrakte** Methode *ressource.* Des Weiteren hat der Ressourcen-Inhaber eine Hook-Methode *ressource_praefix()*, die in der Ausgabe über die Verwendung/Funktion der Ressource Auskunft gibt.

Beispiel: In den ersten beiden Zeichenketten ist der Präfix rot markiert. In der letzen ist der Präfix der Default (die leere Zeichenkette).

„Ich Teleporter flydsk [10 kg] überwindet 20 Quadranten"
„Ich Tankstelle orbitfuel [Infinity kg] Vorrat: 200 liter"
„Ich Zaubertrank elixa [2 kg] 100 lp nimmt Dir 10 lp"

Eine Ressource hat eine Höhe (eine ganze Zahl) und eine Einheit (z.B. lp liter etc.). Die Einheit ist eine Zeichenkette, da wir nicht auch noch ein Modell für das Rechnen mit Einheiten-behafteten Größen bauen wollen. Eine Ressource hat die Methoden *reduziere / erhoehe*, die Methode *verbraucht?* und ein *to_s* für die Darstellung der Ressource.

7.2.2 Aufgabe

1. Implementieren Sie die Methoden für Ressourcen-Inhaber und Ressource.
2. Refaktorisieren Sie die Klassen, die Ressourcen-Inhaber sind, unter Berücksichtigung der Implementierungen für Ressourcen-Inhaber und Ressource.
    a. Refaktorisieren Sie die *initialize* Methoden. Dabei sollen die Ressource-Objekte im *initialize* der Klassen erzeugt werden und müssen z.B. der Spielerin nicht beim Erzeugen übergeben werden.

       Beispiel: Beim Aufruf von *super* wird das Ressource-Objekt mit den Lebenspunkten und der Einheit „lp" erzeugt.

       ```
       def initialize(name, aktueller_raum,
                           maximale_tragkraft, lebenspunkte)
         super(name, Ressource.new(lebenspunkte,"lp"))
         @aktueller_raum = aktueller_raum
         @rucksack = Rucksack.new(maximale_tragkraft)
       end
       ```

    b. Refaktorisieren Sie die *to_s* Methoden.
3. Implementieren Sie insbesondere auch die abstrakten und die Hook-Methoden in den passenden Subklassen.

7.2.3 Effektoren und Effekte

Ein Effektor wendet seinen Effekt auf einen Ressourcen-Inhaber an. Dazu hat er die Methode `effekt_anwenden(ressource_inhaber)`. Die Methode ruft auf dem Effekt die Methode `anwenden(ressource_inhaber)` auf. Da die Art des Effekts vom jeweiligen Effektor abhängt, hat der Effektor eine **abstrakte** Methode `effekt`. Bevor der Effekt angewendet wird, gibt der Effektor noch einen Hinweis über seine Wirkung auf die Ressource des Inhabers aus.

Beispiel: 1'te Zeile negativer Effekt, 2'te Zeile positiver Effekt

„x_troll verliert durch xcalur 30 lp"
„x_troll gewinnt durch elixa 10 lp"

Der Effekt wird mit einer Stärke, einer Einheit (vgl. Ressource) und einem booleschen Marker, der aussagt ob der Effekt negativ oder positiv ist, erzeugt. Der Effekt hat die Methode `anwenden(ressource_inhaber)`, die die Ressource des Inhabers um die Stärke des Effekts reduziert, wenn der Effekt negativ ist, ansonsten erhöht. Des Weiteren hat der Effekt die Methoden `positiv?()` und `positiv=(ja_nein)`, die die positiv/negativ Wirkung des Effekts steuern. Sowie die Methode `to_s`.

7.2.3 Aufgabe

1. Implementieren Sie die Methoden für Effektor und Effekt.
2. Refaktorisieren Sie die Klassen, die Effektoren sind, unter Berücksichtigung der Implementierungen für Effektor und Effekt.
    a. Refaktorisieren Sie die `initialize` Methoden. Dabei sollen die Effekt-Objekte im `initialize` der Klassen erzeugt werden und müssen z.B. der Waffe nicht beim Erzeugen übergeben werden.
    b. Refaktorisieren Sie die `to_s` Methoden.
3. Implementieren Sie insbesondere auch die abstrakten Methoden.

7.2.4 Bewohner

Monster und Freunde sind Bewohner. Bewohner interagieren mit der Spielerin, sobald diese den Raum betritt, in dem die Bewohner leben. Dafür haben Bewohner die Methode `interagiere_mit(spielerin)`. Diese Methode arbeitet mit folgender Logik:

1. Wenn der Bewohner nicht lebendig ist, dann passiert nichts und es wird z.B. ausgegeben:

    „Toter Bewohner x_troll ist aus dem Spiel"

2. Wenn der Bewohner lebendig ist, aber die Höhe seiner Ressource nicht mehr ausreicht, um mit der Spielerin zu interagieren, dann passiert nichts und es wird z.B. ausgegeben:

   „x_troll ist zu schwach attackiert Julia nicht"

3. Sonst wird der Effekt auf die Spielerin angewendet und die eigene Ressource um die Stärke des Effekts verringert. Des Weiteren wird z.B. ausgegeben:

   „elfe_7 hilft Julia mit 20 lp"
   „x_troll attackiert Julia mit 30 lp"

4. Da sich die Ausgaben für Monster und Freund unter 3. nur an einer Stelle unterscheiden (im Text rot markiert), hat Bewohner eine Hook-Methode `interaktions_form`, die in den Subklassen überschrieben werden kann.

7.2.4 Aufgabe

Implementieren Sie die Methode `interagiere_mit(spielerin)` für die Klasse Bewohner und die Methode `interaktions_form` für die Klassen Bewohner, Monster und Freund.

7.2.5 Waffe

Da Waffen nach einmaligem Gebrauch für die folgenden 4 Versuche nicht nutzbar sind, muss sich die Waffe merken, wann sie das letzte Mal eingesetzt wurde. Eine Methode `nutzbar?` prüft dazu einfach, ob der n-te Einsatz ein Vielfaches von 4 ist. Wenn eine Waffe nicht nutzbar ist, darf der Effekt der Waffe nicht angewendet werden. Dazu muss das Standardverhalten des Effektors in der Methode `effekt_anwenden` überschrieben werden:

1. Wenn die Waffe nutzbar ist, muss der Zähler für den Einsatz erhöht werden und dann die Methode `effekt_anwenden` der Superklasse aktiviert werden.
2. Wenn die Waffe nicht nutzbar ist, soll z.B. folgende Ausgabe darauf hinweisen:

   „Ich Waffe xcalur [20 kg] nimmt Dir 30 lp momentan in der Wartung"

7.2.5 Aufgabe

Implementieren Sie die Erweiterungen für die Klasse Waffe und implementieren Sie die Methode `nutzbar?` auch für die Klasse Gegenstand.

7.2.6 Zaubertrank

Wenn der Effekt des Zaubertranks angewendet wird, muss der Zaubertrank danach seine Wirkung reduzieren. Daher muss in der Klasse Zaubertrank das Standardverhalten

der Methode `effekt_anwenden` der Superklasse um den Aufruf von `reduziere_ressource` erweitert werden.

7.2.6 Aufgabe

Überschreiben Sie die Implementierung der Methode `effekt_anwenden` in der Klasse Zaubertrank. Nutzen Sie für das reine Anwenden des Effektes die Methode der Superklasse.

7.2.7 Tankstelle

Für die Tankstelle benötigen wir noch die Methode `betanke(fahrzeug, liter)` um z.B. den Teleporter wieder aufzutanken. Die Methode prüft, ob noch genügend Kraftstoff vorhanden ist, um das Fahrzeug mit der gewünschten Literzahl zu betanken, betankt dann das Fahrzeug, indem es dessen Ressource erhöht, und reduziert anschließend die eigene Ressource. Wenn nicht genügend Kraftstoff vorhanden ist, soll z.B. folgende Ausgabe erfolgen:

„Zapfsäule orbitfuel hat nicht ausreichend Krafstoff:200<300“

7.2.7 Aufgabe

Implementieren Sie die Methode `betanke(fahrzeug, liter)` für die Klasse Tankstelle.

7.2.8 Teleporter

Mit dem Teleporter kann die Spielerin beliebige Räume in Reichweite des Teleporters überspringen. Dazu übergibt sie in der Methode `teleportiere(y, x, y_richtung, x_richtung)` die Koordinaten des aktuellen Raumes und die Anzahl der Quadranten in `y_richtung` und `x_richtung` an. Wenn der Teleporter genügend Reichweite hat und das Ziel nicht außerhalb der Welt liegt, dann ist die Teleportation möglich. Die Methode arbeitet wie folgt:

1. Wenn die Reichweite nicht ausreicht, soll z.B. folgende Ausgabe erfolgen:

   „Nicht ausreichend Treibstoff in flydsk für 7 (> 6) Quadranten“

2. Wenn das Ziel außerhalb der Welt liegt, soll z.B. folgende Ausgabe erfolgen:

   „Quadrant_6_7 Flug geht ins Nirwana: Teleportation nicht möglich“

3. Sonst soll der Zielraum bestimmt werden und die Ressource des Teleportes um die Strecke reduziert werden.
4. Das Ergebnis der Methode ist im positiven Fall der Zielraum, sonst `nil`.

Das Wechseln des Raumes passiert in einer Methode der Spielerin.

7.2.8 Aufgabe

Implementieren Sie die Methode `teleportiere(y, x, y_richtung, x_richtung)` für die Klasse Teleporter.

7.2.9 Spielerin

In der Klasse Spielerin müssen wir insgesamt 4 neue Methoden schreiben. (Eigentlich mehr aber der Rest ist vorgegeben)

1. `im_nachbarraum_umsehen(richtung)`: Die Methode ermittelt den Nachbarraum in der angegebenen Richtung und gibt den Raum aus.
2. `attackiere(richtung, bewohner_name, gegenstand_name)` : Mit dieser Methode attackiert die Spielerin einen Bewohner `bewohner_name` mittels einem Gegenstand `gegenstand_name` im Nachbarraum in Richtung `richtung`. Die Methode arbeitet wie folgt:
    a. Ermittelt den Nachbarraum. Endet wenn der Raum nicht existiert.
    b. Sucht im Nachbarraum nach dem Bewohner.
    c. Nimmt den Gegenstand aus dem Rucksack.
    d. Wenn Bewohner oder Gegenstand `nil` sind, wird die Methode beendet.
    e. Wenn der Gegenstand kein Effektor ist, dann wird z.B. folgendes ausgegeben:

       „Gegenstand flydsk nicht gegen x_troll wirksam“

    f. Sonst wird er Effektor auf den Bewohner angewendet.
    g. Danach wird geprüft, ob der Gegenstand noch brauchbar ist und dieser ggf. in den Rucksack zurückgelegt.
3. `teleportiere(vertikal,horizontal,teleportierer_name)`: Die Methode arbeitet wie folgt:
    a. Nimmt den Gegenstand `teleportierer_name` aus dem Rucksack.
    b. Endet, wenn der Gegenstand nicht im Rucksack ist.
    c. Prüft, ob der Gegenstand vom Typ Teleportierer ist, wenn nicht wird beendet und z.B. folgendes ausgegeben:

       „xcalur kein geeignetes Fahrzeug“

    d. Sonst:
        i. Ermittelt die Koordinaten des aktuellen Raums (→ Klasse Welt)
        ii. Lässt sich vom Teleportierer befördern.
        iii. Betritt den Raum, den der Teleportierer zurückliefert. (Methode `betrete_raum(raum)`)
        iv. Danach wird geprüft, ob der Gegenstand noch brauchbar ist und dieser ggf. in den Rucksack zurückgelegt.
        v. Abschließend wird die Info über die Spielerin ausgegeben.

4. `auftanken(fahrzeug_name, tankstelle_name,liter)`: Die Methode arbeitet wie folgt:
    a. Nimmt das Fahrzeug `fahrzeug_name` aus dem Rucksack.
    b. Sucht die Tankstelle `tankstelle_name` im Raum.
    c. Lässt die Tankstelle das Fahrzeug betanken.
    d. Danach wird geprüft, ob der Gegenstand noch brauchbar ist und dieser ggf. in den Rucksack zurückgelegt.
    e. Abschließend wird die Info über die Spielerin ausgegeben.

7.2.9 Aufgabe

Implementieren Sie die Methoden für die Klasse Spielerin.

7.3 Regeln/Bedingungen für das Spielende

Das Spiel soll beendet werden, wenn

1. Mit einem Befehl das Spiel explizit beendet wird. (wie immer)
2. Die Spielerin das Abenteuer nicht überlebt. (bereits implementiert)
3. Die Spielerin das Ziel des Spiels erreicht hat.

7.3 Aufgabe

Formulieren Sie die Erfolgsbedingungen für das Spiel und implementieren Sie die Methode `ziel_erreicht?` in der Klasse Spiel.

7.4 Entwerfen Sie ein Spiel mit Spielregeln und spielen Sie das Spiel