1. Programmieraufgabe

Objektorientierte Programmiertechniken

LVA-Nr. 185.A01
2021/2022 W
TU Wien

Themen:

Aufbau der Zusammenarbeit in der Gruppe, Einrichten einer Arbeitsumgebung, Datenabstraktion, Klassenzusammenhalt, Objektkopplung

Ausgabe:

12. 10. 2021

Abgabe (Deadline):

19. 10. 2021, 10:00 Uhr

Abgabeverzeichnis:

`Aufgabe1-3`

Hochladen ins Git-Repository mittels `push`

Programmaufruf:

`java Test`

Grundlage:

Kapitel 1 und Anhang A des Skriptums

Kontext

Heimische Wälder spielen im Kampf gegen den Klimawandel eine große Rolle[1]: Sie dienen als Kohlenstoffsenken und mildern Auswirkungen wie Hitzestaus, Überschwemmungen und Dürren. Die EU erarbeitet gerade eine Strategie zur Steigerung der Quantität und Qualität heimischer Wälder. Unter anderem wird diskutiert, welcher Mindestanteil der Wälder naturbelassen sein soll. Bewirtschaftete Wälder versorgen uns mit Rohstoffen. Naturbelassene Wälder fördern die biologische Vielfalt und dienen zur Katastrophenvorsorge. Alle Wälder binden CO_2, stabilisieren den Boden und reinigen Wasser und Luft, aber unterschiedlich effektiv:

- Es dauert lange, bis sich in einem sich selbst überlassenen Wald die gewünschte biologische Vielfalt im Gleichgewicht einstellt. Erst dann erzielt der Wald in allen genannten Kriterien sehr gute Leistungen und bindet zuverlässig viel CO_2 für lange Zeit, das nur bei seltenen großen Katastrophen wie Waldbränden freigesetzt wird.
- Bewirtschaftete Wälder sind auf raschen Holzzuwachs ausgelegt und binden viel CO_2, mehr als naturbelassene. Unter optimalen Bedingungen bleibt viel davon lange in Holzprodukten (wenn nicht thermisch verwertet) und im Waldboden (verrottete Holzreste) gebunden. Allerdings werden bewirtschaftete Wälder mangels biologischer Vielfalt häufiger von Katastrophen heimgesucht (Schädlinge, Krankheiten, Sturm, Hangrutschungen), wobei große Mengen CO_2 freigesetzt werden können. So verhält es sich auch mit der Bodenstabilisierung und Reinigungskraft: Bewirtschafteter Wald ist ähnlich leistungsfähig wie naturbelassener (nach der Holzernte eingeschränkt), verliert bei Katastrophen aber stark an Wirksamkeit.

Welche Aufgabe zu lösen ist

Entwickeln Sie ein Java-Programm, das mögliche Entwicklungen eines fiktiven Walds von 1 ha Größe über tausend Jahre simuliert und nach jeweils hundert Jahren die Zustandsparameter ausgibt.

Waldzustand: Entwickeln Sie zunächst ein Modell für den Zustand des Walds, das zumindest folgende Parameter beinhaltet:

Baumbestand: wachsende Bäume gemessen in Festmetern (fm) an Holz

Altersstruktur: Anteil der Bäume (gemessen in fm an Holz) jeden Baumalters (in ganzen Jahren) am gesamten Baumbestand

Gesundheit: Kennzahl für Diversität und Widerstand gegen Krankheiten, Schädlinge, Umwelteinflüsse (≥ 0.25, ≤ 1, kleiner ist besser)

[1] Regenwälder und nordische Wälder sind wichtig, aber außerhalb unserer Kontrolle.

Zielbestand: erwarteter idealer Baumbestand in fm an Holz, abhängig von Entwicklungsstadium, Baumarten, Standort und Pflege

Ernte: Gesamtsumme der dem Wald entnommenen Holzmenge in fm

CO_2-Vorrat: Summe des Vorrats im Wald einschließlich Waldboden (in Tonnen CO_2, wobei eine Tonne CO_2 einem fm Holz entspricht)

Zuwachs- und Ausfallsfaktor: Diese zwei Werte fassen alle Arten von Umwelteinflüssen in einem Jahr zusammen. Sie drücken (vereinfacht) aus, wie stark sich der Baumbestand durch das Baumwachstum erhöht (Zuwachsfaktor zwischen 0 und 0.08, Durchschnitt 0.04) und wie er sich durch absterbendes Holz verringert (Ausfallsfaktor meist zwischen 0 und 0.08 mit Durchschnitt 0.04, kann in Ausnahmefällen (Katastrophenjahren) bis zu 1 betragen). Zwei Faktoren pro Jahr sind es deswegen, weil ein starker Zuwachs und ein starker Ausfall auch im gleichen Jahr auftreten kann, was zu einer anderen Entwicklung führt als schwaches Wachstum bei geringem Ausfall. Bestimmen Sie zufallsgesteuert für tausend Jahre die jährlichen Zuwachs- und Ausfallsfaktoren.

Bewirtschaftungsmodelle legen fest, wie aus einem Waldzustand und dem Zuwachs- und Ausfallsfaktor des entsprechenden Jahres der Waldzustand für das nächste Jahr ermittelt wird. Entwickeln Sie zwei Bewirtschaftungsmodelle, je eines für bewirtschaftete und naturbelassene Wälder. Folgende Werte werden in allen Modellen benötigt:

Ausfall (als Anteil am Baumbestand) = Ausfallsfaktor mal Gesundheit.

Zuwachs (Absolutwert, gemessen in fm, kann auch negativ werden) = Zielbestand mal Zuwachsfaktor minus Baumbestand mal Ausfall.

Naturbelassenes Modell: Es gibt keine Holzernte. Die anderen Parameter ändern sich jährlich wie folgt:

Baumbestand: Der Zuwachs wird zum Baumbestand addiert.

Altersstruktur: Bäume werden älter, das heißt, der Anteil für Alter n wird zum Anteil für Alter $n + 1$. Zusätzlich macht abgestorbenes Holz gleichmäßig verteilt auf jedes Alter Platz für neue Bäume, das heißt, der Anteil für jedes Alter $n > 0$ wird mit $(1 - \text{Ausfall})$ multipliziert und der Anteil für Alter 0 wird gleich dem Ausfall.

Gesundheit: Sie hängt von der Altersstruktur ab. Sie ist gut (niedriger Wert), wenn jede Altersklasse zwischen 1 und 250 Jahren einen ausreichenden Anteil enthält und umso schlechter, je mehr und größere Lücken es gibt. Legen Sie Berechnungsdetails selbst fest.

Zielbestand: Der Höchstwert beträgt 250 fm. Bei einem Ausfall ≥ 0.3 wird der aktuelle Zielbestand mit $(1 - \text{Ausfall})$ multipliziert und erhöht sich danach bis zum Höchstwert jährlich um 5 fm.

CO_2-Vorrat: Der Zuwachs wird zum CO_2-Vorrat addiert. Bei Ausfall < 0.3 wird zusätzlich Baumbestand $\cdot$ Ausfall/3 addiert (1/3 des CO_2 im abgestorbenen Holz bleibt im Boden), bei Ausfall ≥ 0.3 wird der gesamte CO_2-Vorrat mit $(1 - \text{Ausfall}/2)$ multipliziert (Waldbrand).

Bewirtschaftetes Modell: Ausgangspunkt ist das naturbelassene Modell, das abgewandelt wird. Der Höchstwert für den Zielbestand beträgt wegen künstlich optimierter Bedingungen 350 fm (statt 250 fm). Bei Ausfall ≥ 0.3 gibt es sonst keinen Unterschied zum naturbelassenen Modell. Bei Ausfall ≥ 0.1 (aber < 0.3) liegt eine kleine Katastrophe vor, der durch eine ungeplante Holzernte zu begegnen ist. Dabei werden alle verbliebenen Bäume ab 45 Jahren gefällt, und die Hälfte des gefällten sowie auf natürliche Weise ausgefallenen Holzes wird dem Wald entnommen. Im jeweils elften aufeinander folgenden Jahr mit einem Ausfall < 0.1 und ohne Holzernte erfolgt eine geplante Holzernte, wobei alle Bäume im Alter von 75 Jahren und mehr gefällt werden; zwei Drittel des gefällten (aber nicht des natürlich ausgefallenen) Holzes werden dem Wald entnommen. In die Berechnung des Baumbestands und der Altersstruktur wird das gesamte gefällte Holz einbezogen. Das Fällen von Holz kann als künstliche Erhöhung des Ausfalls betrachtet werden (aber nicht auf jedes Baumalter gleich verteilt). Ist der durch Fällen erhöhte Ausfall ≥ 0.3, wird eine entsprechende Reduktion des Zielbestands und CO_2-Vorrats nötig, wie im naturbelassenen Modell beschrieben. Andernfalls ist in der Berechnung des CO_2-Vorrats zu berücksichtigen, dass für entnommenes Holz kein CO_2 in den Boden eingetragen wird.

Simulation: Machen Sie je einen Simulationslauf mit den beiden Bewirtschaftungsmodellen basierend auf den gleichen Anfangszuständen sowie Zuwachs- und Ausfallsfaktoren und stellen Sie die Ergebnisse nach jeweils 100 Jahren vergleichend gegenüber.

Test, Aufgabenaufteilung: Testen Sie Ihr Programm sorgfältig. Nach dem Programmstart sollen ohne Benutzerinteraktion (das heißt, ohne Eingabe über die Tastatur oder Maus) die Simulationsergebnisse möglichst übersichtlich angezeigt werden. Das Programm soll (nach neuerlicher Übersetzung aller `.java`-Dateien) mittels `java Test` vom Verzeichnis `Aufgabe1-3` aus aufrufbar sein.

Programmname „`Test`“ aus gutem Grund gewählt

im richtigen Verzeichnis ausführbar?

Neben Programmtext soll die Datei `Test.java` als Kommentar eine kurze, aber verständliche Beschreibung der Aufteilung der Arbeiten auf die einzelnen Gruppenmitglieder enthalten – wer hat was gemacht. Beschreibungen wie die folgende reichen nicht: „Alle haben mitgearbeitet.“

Arbeitsaufteilung kurz, verständlich beschreiben

Wie die Aufgabe zu lösen ist

einfach halten

Der Programmtext Ihrer Lösung soll möglichst einfach sein und keine unnötige Funktionalität haben. Er soll wiederverwendbar sein, da die nächste Aufgabe auf Teilen davon aufbaut. Vermeiden Sie jedoch Vorgriffe, das heißt, schreiben Sie keine Programmteile aufgrund der Vermutung, dass diese Teile in der nächsten Aufgabe verlangt sein könnten.

Datenabstraktion

Achten Sie auf Datenabstraktion: Alles, was kaum trennbar miteinander verbunden ist, soll in einem Objekt gekapselt sein, leicht voneinander trennbare Einheiten sollen zu verschiedenen Objekten gehören. Es soll in Ihrem Programm mehrere, voneinander möglichst unabhängige Objekte geben. Auf Daten soll nur über dafür vorgesehene Methoden zugegriffen werden. Unnötige Zugriffe und unnötige Zugreifbarkeit von Daten und

Methoden sind zu vermeiden. Achten Sie auf hohen Klassenzusammenhalt und schwache Objektkopplung.

Klassenzusammenhalt, Objektkopplung

Diese Aufgabe hilft Tutoren, Ihre Kenntnisse sowie die Zusammenarbeit in der Gruppe einzuschätzen. Bitte sorgen Sie in Ihrem eigenen Interesse dafür, dass jedes Gruppenmitglied etwa in gleichem Maße mitarbeitet. Sonst könnten Sie bei einer Fehleinschätzung wertvolle Zeit verlieren. Scheuen Sie sich bitte nicht, Ihren Tutor um Hilfe zu bitten, falls Sie bei der Lösung der Aufgabe Probleme haben oder keine brauchbare Zusammenarbeit in der Gruppe zustande kommt.

Warum die Aufgabe diese Form hat

Umfang und Schwierigkeitsgrad der Aufgabe wurden so gewählt, dass die eigentliche Programmierung bei guter Organisation und entsprechendem Vorwissen nicht zu viel Zeit in Anspruch nimmt, aber eine Einarbeitung in neue Themen nötig ist und Diskussionsbedarf entsteht. Nutzen Sie die Gelegenheit um die Aufgabenverteilung und interne Abläufe innerhalb der Gruppe zu organisieren. Auf eine ganz genaue Spezifikation der Aufgabe wird bewusst verzichtet:

- Sie sollen in der Gruppe diskutieren, wie Sie die Aufgabe verstehen und welche Lösungswege geeignet erscheinen.
- Sie sollen sich daran gewöhnen, dass Aufgaben nicht vollständig spezifiziert sind, aber trotzdem Vorgaben eingehalten werden müssen.
- Sie sollen sich eine eigene brauchbare Faktorisierung überlegen und dabei von Merkmalen wie Datenkapselung, Klassenzusammenhalt und Objektkopplung (statt starrer Vorgaben) leiten lassen.
- Sie sollen auch die Verantwortung für die Korrektheit Ihrer Lösung (so wie Sie sie selbst verstehen) übernehmen, indem Sie entsprechende Tests durchführen.

Allgemeine Informationen zur Übung

Folgende Informationen betreffen diese und auch alle weiteren Aufgaben.

Was bei der Lösung der Aufgabe zu beachten ist

Schwerpunkte beachten

Unter der Überschrift „Wie die Aufgabe zu lösen ist“ finden Sie Hinweise darauf, wie Sie die Lösung der Aufgabe vereinfachen können und welche Fallen Sie umgehen sollen, erfahren aber auch, welche Aspekte bei der Beurteilung als wichtig betrachtet werden. In der ersten Aufgabe kommt es beispielsweise auf Datenabstraktion, Klassenzusammenhalt, Objektkopplung und die Einfachheit der Lösung an. Das heißt, in späteren Aufgaben können Ihnen bei solchen Hinweisen auch für unnötig komplizierte oder umfangreiche Lösungen Punkte abgezogen werden, weil Sie sich nicht an die Vorgaben gehalten haben. Unterschiedliche Aufgaben haben unterschiedliche Schwerpunkte. Die nächste Aufgabe wird nicht nach dem gleichen Schema beurteilt wie die vorige. Richten Sie sich daher nach der jeweiligen Aufgabenstellung.

Ein häufiger Fehler besteht darin, eine Aufgabe nach Gefühl zu lösen ohne zu verstehen, worauf es ankommt. Meist bezieht sich die Aufgabe auf ein Thema, das zuvor theoretisch behandelt wurde. Versuchen Sie, eine Beziehung zwischen der Aufgabenstellung und dem OOP-Stoff herzustellen. Achten Sie darauf, Fachbegriffe (wie Datenabstraktion, Klassenzusammenhalt und Objektkopplung) nicht nur umgangssprachlich zu interpretieren, sondern verwenden Sie diese Begriffe so wie im Skriptum beschrieben. Die ersten Aufgaben sind vermutlich auch ohne Skriptum lösbar, spätere aber kaum. Als Hilfestellung sind in jeder Aufgabenstellung Teile des Skriptums genannt, in denen die relevantesten Themen behandelt werden – bei komplizierten Themen oft nur wenige Seiten.

strukturiert vorgehen

Versuchen Sie nicht, Teile der Aufgabenstellung durch Tricks oder Spitzfindigkeiten zu umgehen. Beispielsweise gibt es immer wieder Lösungsversuche, in denen die Test-Klasse nur den String „Tests erfolgreich“ ausgibt, statt tatsächlich Tests durchzuführen. Solche Versuche werden durch händische Beurteilungen mit hoher Wahrscheinlichkeit erkannt. Spätere Aufgaben enthalten oft Schwierigkeiten, die mit Allgemeinwissen alleine oder über aufgabenbezogene Web-Recherchen kaum zu lösen sind. Gerade in solchen Fällen ist davon abzuraten, die Schwierigkeiten durch Tricks zu umgehen. Hinweise zur richtigen Lösung lassen sich im Skriptum und auf den Vorlesungsfolien finden.

keine Spitzfindigkeiten

Ihre Lösung bestehend aus `.java`-Dateien muss am Tag der Abgabe um 10:00 Uhr pünktlich im Git-Repository Ihrer Gruppe stehen. Übersetzte `.class`-Dateien sollen nicht ins Repository gestellt werden, da sie die Zusammenarbeit in der Gruppe erschweren und vor der Beurteilung ohnehin neu generiert werden. Zugangsinformationen zum Repository erhalten Sie in Kürze. Informationen zum Umgang mit dem Repository finden Sie im Anhang des Skriptums. Es wird empfohlen, rechtzeitig vor der Deadline die Lösung auf dem Übungsrechner auszuprobieren und die Daten dabei durch `pull` aus dem Repository zu laden. So können Sie typische Fehler bei der Abgabe (z.B. auf `push` vergessen, Lösung im falschen Verzeichnis, falsche `package`- und `include`-Anweisungen, Klassen aus Nicht-Standard-Paketen verwendet und nicht mit abgegeben) sowie Inkompatibilitäten aufgrund unterschiedlicher Versionen und Betriebssystemeinstellungen (z. B. Dateinamen mit Umlauten sowie neueste Sprach-Features nicht unterstützt) erkennen und beseitigen.

ausprobieren

Was Ihr Tutor von Ihnen wissen möchte

Ihr Tutor wird Ihnen in Kürze eine Mail schreiben um sich vorzustellen und um Informationen über Sie zu bitten. Geben Sie diese Informationen möglichst bald, damit die für Sie am besten geeignete Form der Betreuung gewählt werden kann. Unabhängig von der Form der Betreuung kann natürlich jedes Gruppenmitglied jederzeit konkrete Fragen an den Tutor richten. Scheuen Sie sich bitte nicht, sich auch mit organisatorischen oder gruppeninternen Problemen, die Sie möglicherweise nicht selbst lösen können, an den Tutor zu wenden. Früh im Semester sind Probleme meist einfacher zu lösen als im bereits weit fortgeschrittenen Semester.