1. Programmieraufgabe

Objektorientierte Programmiertechniken

LVA-Nr. 185.A01
2020/2021 W
TU Wien

Themen:

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

Ausgabe:

14. 10. 2020

Abgabe (Deadline):

21. 10. 2020, 12:00 Uhr

Abgabeverzeichnis:

`Aufgabe1-3`

Hochladen ins Git-Repository mittels `push`

Programmaufruf:

`java Test`

Grundlage:

Kapitel 1 und Anhang A des Skriptums

Kontext

Grillen zirpen, Bienen summen, Schmetterlinge flattern über die blühende Wiese. Davon träumen wir. Noch ist das Grundstück kahles, ungenutztes Ackerland. Dank einer Ökologisierungsinitiative soll unser Traum Wirklichkeit werden. Es liegt an uns, was wir daraus machen.

Wir müssen dafür sorgen, dass der Boden sich für für eine blühende Wiese eignet. Humus aus zersetzten Planzenresten spielt dabei eine große Rolle: In gut gereiftem, fruchtbarem Humus leben unzählige Mikroorganismen, aber frischer Humus stört die Bodenfruchtbarkeit (wobei der Reifungsprozess kontinuierlich voranschreitet, aber mehrere Jahre dauern kann). Humus bindet große Mengen CO_2 für lange Zeit im Boden und kann viel Wasser speichern. Regelmäßiges Mähen einer Wiese ist notwendig, damit Wiesenpflanzen nicht durch Gestrüpp verdrängt werden. Wir gehen davon aus, dass die Wiese gemulcht wird, also die Pflanzenreste als frischer Humus auf der Wiese bleiben. Das Ziel ist eine über viele Jahre möglichst artenreiche Wiese. Mulchen beeinflusst die Vegetation:

- Zu häufiges Mulchen begünstigt eine kleine Zahl an Pflanzenarten und vernichtet andere, darunter viele Wiesenblumen, die keine Samen bilden können. Zu seltenes Mulchen begünstigt unerwünschte Stauden, die Wiesenblumen verdrängen.
- Sehr fruchtbare Böden fördern wenige angepasste Pflanzenarten, die insbesondere Wiesenblumen verdrängen. Unfruchtbare Böden führen natürlich auch zu keiner schönen Wiese.
- Mulchen steigert die Fruchtbarkeit langfristig, verringert die Fruchtbarkeit aber kurzfristig.

Welche Aufgabe zu lösen ist

Entwickeln Sie ein Java-Programm, das verschiedene Mulch-Konzepte über hundert Jahre simuliert. Konkret:

- Entwickeln Sie ein Modell für Wiesenpflanzen, das für verschiedene Pflanzenarten abhängig von Einflussfaktoren den Zuwachs (Änderung der Masse einer Pflanze) und die Vermehrung (Änderung der Pflanzenzahl dieser Art) in folgenden Situationen berechnet:
  - Zuwachs und Vermehrung an einem normalen Tag innerhalb einer angenommenen Wachstumsperiode,
  - negativer Zuwachs und Vermehrung an einem Tag, an dem gemulcht wird,
  - negativer Zuwachs und Vermehrung über den gesamten Winter (außerhalb der Wachstumsperiode).

  Einflussfaktoren sind beispielsweise:

  - Stärken der Abhängigkeiten von gereiftem Humus (positiv) und frischem Humus (negativ),
  - Stärke des Durchsetzungsvermögens gegenüber anderen Arten,
  - Neigung zur Bildung von Ausläufern (vegetative Vermehrung),
  - Mindestdauer für die Produktion von Samen (in Tagen),
  - Resistenz gegenüber Mulchen und Kälte (Winterfestigkeit).

- Legen Sie mindestens zehn (besser mehr) verschiedene Pflanzenarten an, die Sie durch Festlegen der Einflussfaktoren in obigem Modell beschreiben. Legen Sie fest, wie groß Masse und Anzahl der Pflanzen der jeweiligen Art zu Beginn der Simulation ist.

- Entwickeln Sie ein Modell des Bodens, das die täglich verfügbare Menge an gereiftem sowie frischem Humus darstellt. Täglich wird ein kleiner Anteil des frischen Humus zu gereiftem Humus.

- Entwickeln Sie ein Modell für Mulch-Konzepte, das es erlaubt, eine beliebige Anzahl beliebiger Tage in einer Wachstumsperiode zu wählen, an denen gemulcht wird (jedes Jahr an den gleichen Tagen).

- Legen Sie mindestens fünf verschiedene Mulch-Konzepte an.

- Führen Sie für jedes der fünf Konzepte je einen Simulationslauf über jeweils hundert Jahre aus, in denen Sie ausgehend von den gleichen Anfangsbedingungen jede tagesaktuelle Anzahl und Masse für jede Wiesenpflanzenart sowie den Zustand des Bodens berechnen, wobei Sie an im Konzept beschriebenen Tagen das Mulchen simulieren. Geben Sie am Ende jeden Simulationslaufs die Werte des letzten Tages im hundertsten Jahr aus. Versuchen Sie, ein Mulch-Konzept zu finden, bei dem möglichst viele Pflanzenarten überleben.

Testen Sie Ihr Programm sorgfältig. Nach dem Programmstart sollen ohne Benutzerinteraktion (das heißt, ohne Eingabe über die Tastatur oder Maus) die Simulationsergebnisse 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

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.

einfach halten

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

Datenabstraktion

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

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

Schwerpunkte beachten

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 12:00 Uhr pünktlich im Git-Repository Ihrer Gruppe stehen. Übersetzte `.class`-Daten 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. 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.