Complang
Puntigam
- Über mich
- Research
- Lehre
  - LVAs 2015 W
    - PK
    - OOP
      - 1. Aufgabe
      - **2. Aufgabe**
    - Typsysteme
  - LVAs 2015 S
  - frühere Lehre
- Links

2. Übungsaufgabe

Thema:
Aufwandsabschätzung, Programmiereffizienz, Untertypbeziehungen und dynamisches Binden

Termine:

Ausgabe: 21.10.2015
Abgabe: 28.10.2015, 12:00 Uhr

Abgabeverzeichnis:
Aufgabe1-3

Programmaufruf:
java Test

Grundlage:
Skriptum, Schwerpunkt auf Abschnitt 2.1

Aufgabe

Welche Aufgabe zu lösen ist:
Die Gemeinde Hinterwaldgstetten hat mit Hilfe des Programms aus Aufgabe 1 bereits mehrere Varianten zur Bepflanzung des Gemeindegrundstücks simuliert, aber noch keine Entscheidung getroffen. Immerhin hat man schon mit einer Nachbargemeinde und zwei privaten Waldbesitzern Kontakt aufgenommen, wo ähnliche Entscheidungen anstehen und Interesse an einem Simulationsprogramm besteht. Dabei hat sich gezeigt, dass das Programm aus Aufgabe 1 die Erwartungen bei Weitem noch nicht erfüllt. Folgende als Wesentlich erachtete Punkte wurden angesprochen:

- Neben der Bindung von Kohlendioxid darf der wirtschaftliche Faktor nicht vernachlässigt werden. Die Simulation sollte in der Lage sein, die jährlichen Kosten der Bewirtschaftung sowie erwartete Erlöse aus dem Holzverkauf darzustellen. Natürlich wären auch die erwarteten Gewinne bzw. Verluste sowie sich daraus ergebende Verzinsungen des eingesetzten Kapitals über längere Zeiträume zu betrachten.
- Für die Gemeinden ist es wichtig, dass ein Wald einen gewissen Erholungswert bietet. Je nach Bewirtschaftungsform ist dieser nicht unbedingt gegeben. Beispielsweise ist ein dicht bewachsener Energiewald ebenso wie ein zum Schutz der Bäume umzäunter Jungwald von Menschen kaum zu betreten. Ein Laub- oder Mischwald mit hochgewachsenen Bämen und ausladenden Kronen bietet dagegen einen großen Erholungswert. Auch dieser Aspekt ist in der Simulation zu berücksichtigen und nach Möglichkeit als tourismusbezogener

wirtschaftlicher Faktor in die Gewinn- und Verlustrechnung einzubeziehen.

- Tatsächlich ist der jährliche Holzzuwachs, die Erntemenge, der Totholzanfall etc. nicht so linear wie in der ursprünglichen Simulation angenommen, sondern hängt stark von Art und Alter der Bäume sowie der Bewirtschaftungsform ab. Beispielsweise wachsen einige Baumarten in der Jugend sehr rasch und später viel langsamer. Andere Baumarten wachsen in der Jugend zögerlich und steigern den Holzzuwachs später unter günstigen Bedingungen (die durch die Bewirtschaftung – etwa dem Entfernen konkurrierender Bäme – hergestellt werden) ganz gewaltig. Totholz fällt beispielsweise an, wenn Jungbäume oder tief stehende Äste wegen Verschattung absterben, später bei Fällung oder Erreichen des natürlichen Lebensalters, dazwischen aber höchstens durch Krankheiten oder außergewöhnliche Umwelteinflüsse. Für bestimmte Mischungen von Baumarten und Bewirtschaftungsformen werden eigene Modelle benötigt, die die Gegebenheiten genauer widerspiegeln und für jedes Jahr in einem längeren Zeitraum (Anzahl der Jahre vom Modell abhängig) eigene Werte für Zuwachs, Erntemenge und Totholzanfall beinhalten. In der Simulation wiederholen sich diese Zeiträme zyklisch, sodass unabhängig vom Zeitraum eines Modells beliebig lange Simulationen möglich sind. Natürlich ist es nötig, die in einem Modell verwendeten Holzmengen mit der entsprechenden Waldfläche zu multiplizieren. Es soll auch möglich sein, mehrere Modelle gleichzeitig für unterschiedliche Waldbereiche einzusetzen.
- Zyklisch wiederholte Modelle sind für Simulationen über lange Zeiträume sinnvoll. Aber kurze Phasen wie die Neuanlage oder Rodung eines Waldes sind damit nicht simulierbar. Dafür benötigt man eigene Modelle, die zu Beginn oder am Ende einer Simulation eingesetzt werden (ergänzt durch ein zyklisch wiederholtes Modell).
- Obige Modelle können nur den normalen Ablauf simulieren. Sonderereignisse wie seltene Wetterereignisse oder Waldbrände lassen sich damit kaum darstellen. Dafür benötigt man eigene Modelle, die entweder zu vorgegebenen Zeitpunkten oder zufällig in Simulationen eingestreut werden.
- Alle realitätsnahen Modelle müssen bestimmte Annahmen über klimatische Bedingungen (Temperatur, Niederschlagsmengen, etc.) treffen. Im Idealfall sind die Modelle so parametrisiert, dass Abweichungen von den üblichen Bedingungen berücksichtigt werden können. Zur sinnvollen Nutzung dieser Parameter ist ein weiteres Modell zur Simulation des Klimas nötig, das eine bestimmte vorgegebene oder zufällige Abfolge von feuchten, trockenen, heißen, kalten, ... Jahren festlegt.
- Nicht nur klimatische sondern auch wirtschaftliche Bedingungen ändern sich mit der Zeit. Die Kosten der Bewirtschaftung unterliegen genauso Schwankungen wie Holzpreise. Auch diese Schwankungen sollen in eigenen Modellen berücksichtigt werden.
- Es ist zwar gut, wenn eine Simulationsrechnung viele Daten liefert, aber eine zu große Datenmenge ist nicht mehr überschaubar. Neben oder statt der Darstellung in Form von Tabellen sind daher grafische Darstellungen nötig, die generelle Tendenzen rasch erkennen lassen.
- Mit einem Simulationslauf ist es nicht getan. Es kommt vor allem darauf an, die Ergebnisse unterschiedlicher Simulationsläufe anschaulich miteinander zu vergleichen. Dabei sollen jedoch nicht nur die Unterschiede in den Ergebnissen, sondern auch die Unterschiede in den

verwendeten Modellen und Parametern klar hervorgestrichen werden.
- Die Verwendung von Modellen erlaubt zwar realitätsnahe Berechnungen, es ist aber sehr viel Fachwissen nötig um solche Modelle zu erstellen und richtig einzusetzen. Daher sollte jedes dieser Modelle einerseits mit Hinweisen auf die zugrundeliegende Fachliteratur und die verwendeten Berechnungen verknüpft sein, andererseits aber auch mit allgemein verständlicher Information über die Annahmen und Verwendungsmöglichkeiten des Modells.

Erweitern Sie Ihre Lösung von Aufgabe 1 entsprechend. Konzentrieren Sie sich wieder nur auf den Kern des Programms (ohne Benutzerschnittstelle) sowie das Testprogramm. Der Kern soll einen möglichst großen Teil der oben beschriebene Funktionalität abdecken, aber keinerlei Eingabe von der Tastatur oder Ausgabe auf den Bildschirm machen.

Das Testprogramm soll mittels *java Test* von *Aufgabe1-3* aus aufrufbar sein und die selbst gewählte Funktionalität überprüfen. Tests sollen ohne Benutzerinteraktion ablaufen, sodass Aufrufer keine Testfälle auswählen oder Testdaten eintippen müssen. Ergebnisse sollen (in nachvollziehbarer und verständlicher Form) am Bildschirm ausgegeben werden.

Neben dem Testprogramm soll die Klasse *Test.java* als Kommentar eine kurze, aber verständliche Beschreibung der Aufteilung der Arbeiten auf die einzelnen Gruppenmitglieder enthalten – wer hat was gemacht.

Wie die Aufgabe zu lösen ist:
Die oben aufgezählten gewünschten Eigenschaften sind nur als Anhaltspunkte gedacht. Sie können auch andere sinnvoll erscheinende Eigenschaften einbauen und einzelne Punkte weglassen bzw. unvollständig oder anders als angedeutet lösen.

Eine der größten Schwierigkeiten dieser Aufgabe besteht in der richtigen Abschätzung des Umfangs der Arbeiten, die Sie bis zum Abgabetermin fertigstellen können. Planen Sie entsprechend Ihren Vorkenntnissen und Fähigkeiten möglichst viel ein, das Sie in der vorgesehenen Zeit auch zum Abschluss bringen können – das heißt, so viel Sie können, aber auf keinen Fall mehr als Sie können. Als groben Anhaltspunkt sollten Sie (jedes Gruppenmitglied) mindestens fünf bis sechs Stunden in die Lösung dieser Aufgabe fließen lassen. Versuchen Sie so effizient wie möglich zu arbeiten und sehr rasch zu einer brauchbaren Lösung zu kommen. Ignorieren Sie Details, die Ihnen als unwichtig erscheinen. Bedenken Sie, dass Sie alle Teile Ihrer Lösung durch Testfälle überprüfen sollen und planen Sie die Zeit für die Entwicklung der Testfälle und für die Fehlerbeseitigung ein.

Senden Sie möglichst bald ein grobes Konzept Ihrer geplanten Arbeiten – am besten mit einer Einteilung, welches Gruppenmitglied was machen soll – an Ihre(n) Tutor(in). Die Tutor(inn)en werden sich bemühen, rasch hilfreiche Rückmeldungen zu den Konzepten zu geben. Überschätzen Sie sich nicht. Erstellen Sie eher ein Konzept, von dem Sie sicher annehmen, dass Sie die Arbeiten zeitlich schaffen. Wenn das Konzept Ihrem Tutor oder Ihrer Tutorin nicht reicht, werden Sie um eine Abänderung gebeten.

Einer der Schwerpunkte dieser Aufgabe ist der Umgang mit Untertypbeziehungen. Planen Sie die Verwendung von Untertypbeziehungen

zusammen mit dynamischem Binden ein.

Warum die Aufgabe diese Form hat:
Sie sollen möglichst große Freiheit bei der Lösung der Aufgabe haben und selbst die Verantwortung für alles übernehmen. Es gibt niemanden, der Ihnen vorschreibt, wie die Aufgabenstellung genau zu verstehen ist.

Diese Aufgabe stellt hohe Anforderungen an jedes einzelne Gruppenmitglied sowie die Zusammenarbeit innerhalb der Gruppe – eine Nagelprobe für das Funktionieren der Gruppe und zum Aufdecken möglicher Schwachstellen.

Untertypbeziehungen sind ein schwieriges, aber für die objektorientierte Programmierung sehr wichtiges Thema. Nutzen Sie die Gelegenheit, bei der Lösung der Aufgabe Erfahrungen damit zu gewinnen. Fehler, die Sie dabei noch machen, wirken sich nicht auf Ihre Beurteilung aus. Sie bekommen von den Tutor(inn)en Rückmeldungen und bei Bedarf maßgeschneiderte Hilfe.