University of Zurich
Department of Informatics

s.e.a.l. software evolution & architecture lab

RERG

Software Engineering Übung 4

Architektur, Entwurf

1 Informationen

1.1 Daten

- Ausgabe: Di. 30.10.2007
- Abgabe: So. 11.11.2007
- Besprechung: Fr. 16.11.2007

1.2 Formales

Die Dateien, welche zu Ihrer Abgabe gehören, müssen in eine .zip-Datei gepackt werden (Diagramme und andere Dokumente als PDF, Quellcode als Textdateien in einem separaten Unterverzeichnis). Die Abgabe erfolgt per Email an stoiber@ifi.uzh.ch.

1.3 Gruppen

Die Übung ist in 3er Gruppen zu lösen. Falls die Aufgaben aufgeteilt wurden, muss klar ersichtlich sein, wer welchen Teil bearbeitet hat. Alle Gruppenmitglieder müssen über alle Teile Auskunft geben können.

2 Ausgangslage und Aufgabenstellung

Anmerkung: Um die folgenden Aufgaben zu lösen, laden Sie bitte von der Übungswebseite die Datei *SourceUe4.zip* herunter. Diese Datei enthält den benötigten Quellcode zum Lösen der Übung.

Die Firma Simulation Systems Inc. entwickelte eine prototypische Software in Java in der sie das Wildleben in einer Savanne per Computer vereinfacht simulieren kann. Diese Realisierung ist noch relativ einfach. Die gegebene prototypische Implementierung bildet den Grundstein für einen erweiterten Wegwerfprototypen.

Die Software funktioniert derzeit so, dass es eine Landkarte mit spezifizierter Grösse gibt und darauf eine definierte Anzahl von Tieren zufällig verteilt sind. Die Tierarten die bisher realisiert wurden sind Löwen und Antilopen. Die Routinen für die Bewegungen der Tiere in jeder Runde

sind bereits realisiert. Ebenso ist bereits das Verhalten der Tiere beim Zusammentreffen mit anderen Tieren implementiert. Bei jeder Bewegung eines Tieres gibt es eine bestimmte Wahrscheinlichkeit, dass dieses hungert und dabei Lebenskraft verliert.

Um ein ökologisches Gleichgewicht zu simulieren, muss die Anwendung noch weiter an die Realität angepasst werden. Bisher noch nicht realisiert wurde ein Geschlecht von Tieren und auch die Möglichkeit Nachwuchs zu zeugen. Die Simulation läuft derzeit so ab, dass die Tiere solange in der Savanne herum reisen und aufeinander treffen, bis nur noch ein einzelnes Tier am Leben ist, welches die Simulation quasi gewinnt.

Um die Software auszuführen, öffnen Sie den Ordner mit dem Quellcode (*SourceUe4*), kompilieren alle Java-Klassen und führen die Klasse SavannahSim im Paket simulation aus. Sie können auch eine Entwicklungsumgebung wie z.B. Eclipse verwenden, laden dort den Quellcode als Projekt und führen die Datei SavannahSim.java als Anwendung aus. Wichtig ist in beiden Fällen dass Sie beim Starten der Anwendung zwei Parameter zur Initialisierung der Anzahl Tiere mit angeben. Zum Beispiel in Eclipse im Run-Menü unter 'Arguments' / 'program arguments' bspw. "2 4" oder beim Ausführen in der Kommandozeile bspw. "java simulation/SavannahSim 2 4". Diese Parameter bestimmten die Anzahl der Löwen und Antilopen mit denen die Simulation gestartet wird.

2.1 Architekturüberblick (4 Punkt)

Verschaffen Sie sich zunächst einen Überblick über die Software, wie sie Ihnen vorliegt. Versuchen Sie die Architektur der Software zu visualisieren, indem Sie ein geeignetes Modell erstellen, welches einen Überblick über die Software gibt. Dies können Sie machen indem Sie z. B. ein UML-Klassendiagramm oder ein UML-Composite-Structure-Diagram (http://en.wikipedia.org/wiki/Unified_Modeling_Language) verwenden. Versuchen Sie die Software in Komponenten einzuteilen und beschreiben Sie diese.

Hinweis: Sie können für diese Aufgabe zum Beispiel das Werkzeug Borland Together verwenden, welches auf den Geräten in den Übungsräumen installiert ist.

2.2 Analyse der verwendeten Architekturstile (3 Punkte)

Analysieren Sie die Implementierung und überlegen Sie wie sie zu dieser eine graphische Benutzerschnittstelle (kurz GUI) realisieren könnten. Überlegen Sie welches Architekturmuster dazu ideal geeignet ist. Entscheiden Sie sich für ein Architekturmuster und begründen Sie Ihre Entscheidung, warum Sie genau dieses Muster wählen und wo die Vorteile gegenüber Anderen liegen. Skizzieren Sie mit wenigen Sätzen wie Sie das Architekturmuster umsetzen können und wie Sie damit eine graphische Benutzerschnittstelle für die gegebene Implementierung realisieren können.

2.3 Entwurfsmuster in der Software (2 Punkte)

Überlegen Sie welches Entwurfsmuster gut passen könnte um das in Aufgabe B gewählte Architekturmuster umzusetzen. Beschreiben und begründen Sie Ihre Überlegung kurz.

2.4 Dokumentation der Architektur (3 Punkte)

Dokumentieren Sie die um das GUI erweiterte Architektur. Verwenden Sie eine dafür geeignete Modellierungstechnik.

2.5 Architekturanpassungen (8 Punkte)

Realisieren Sie die von Ihnen vorgeschlagene Architektur und erweitern sie die Implementierung um eine graphische Benutzeroberfläche. Ihre erweiterte Implementierung soll mindestens folgende Features haben:

- Die gesamte Landkarte (bzw. ein eigens definierter Ausschnitt) soll graphisch angezeigt werden und es soll darauf ersichtlich sein wo sich zum aktuellen Zeitpunkt welche Tiere befinden.
- Eine interaktive Steuerung der Simulationsgeschwindigkeit soll möglich sein, zum Beispiel 1er, 10er und 100er Schritte. Ausserdem soll ein konstanter Ablauf mit zum Beispiel 50 Schritten pro Sekunde möglich sein.
- Eine Realisierung der Monte-Carlo-Simulation (Siehe Übung 2) ist in dieser Übung nicht gefordert.

Für diese Aufgabe ist es empfehlenswert sich auf den GUI-Prototypen den Sie bereits in Übung 2 entwickelt haben zu stützen. Integrieren bzw. realisieren Sie diesen Prototypen mit der nötigen Funktionalität.

Bei der Implementierung eines GUIs helfen Ihnen folgende Seiten:

- Ein Tutorial zu Java Swing:
  http://java.sun.com/docs/books/tutorial/uiswing/
- Zur Realisierung der Benutzeroberfläche:
  http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JComponent.html
  http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JPanel.html
- Zur Realisierung der Interaktionen:
  http://java.sun.com/j2se/1.4.2/docs/api/java/awt/event/MouseListener.html

2.6 Implementierung - freiwillige Zusatzaufgabe

Um nun die Anwendung selbst auch noch etwas realistischer zu gestalten, erweitern Sie die Simulation um das Attribut Geschlecht bei jedem Tier und ermöglichen Sie eine Fortpflanzung von zwei Tieren gleicher Tierart und unterschiedlichen Geschlechts. Falls so ein Zusammentreffen stattfindet, soll es eine Chance von 30 Prozent geben, dass ein neues Tier dieser Tierart auf ein zufälliges Nachbarfeld hinzukommt. Wenn eine Fortpflanzung stattgefunden hat darf es in den nächsten 5 Runden nicht möglich sein dass sich ein beteiligtes Tier erneut fortpflanzt. Die Attributwerte Geschlecht, Kraft, etc. der neuen Tiere werden zufällig gewählt.

Implementieren Sie ausserdem noch einen Mechanismus zum manuellen Hinzufügen weiterer Tiere und ebenso zum manuellen Entfernen existierender Tiere. Hier sollen alle Parameter (Grösse, Position, etc.) des neuen Tieres durch den Benutzer wählbar sein, also keine Zufallswerte.

Passen Sie die Implementierung und die Benutzerschnittstelle entsprechend an.