Konzept zu Aufgabe 09

Aufgabenstellung:

1. Schreiben Sie bitte einen Objektbrowser, der möglichst viel (alle) Eigenschaften eines
   Objekts anzeigt!
   Berücksichtigen Sie dabei bitte folgende Anforderungen:
2. Wie Sie an das zu inspizierende Objekt kommen, ist Ihnen freigestellt: Dies kann über einen Konstruktor oder eine set-Methode geschehen.
3. Die Eigenschaften des Objekts sollen möglichst vollständig angezeigt werden.
4. Es soll in Eigenschaften enthaltener Objekte „hineingezoomt“ werden können (unabhängig
   von der Sichtbarkeit): Hat eine Klasse ein Attribut von nicht-primitiven Typ,
   so sollen die Eigenschaften wie Operationen und ggf. Attributwerte dieses Objekt angezeigt werden können.
5. Reine Konsolausgaben reichen nicht für eine Akzeptanz der Lösung. In den vergangenen
   Semestern habe von HTML-Seiten in einem Frame bis zu Darstellungen mittels Baum viele akzeptable Darstellungen gesehen.
6. „Alle“ ist hier ein hochgestecktes Ziel. Für die Akzeptanz wird es in diesem Aufgabenblatt
   reichen, wenn Sie viele zeigen können.

Überlegung:

Was wird gebraucht?

Model:

Es werden möglichst viele Eigenschaften eines Objekts gebraucht.
Um diese Eigenschaften zu sammeln wird die Vorgehensweise aus der PR2 Vorlesung verwendet, die zuerst alle relevanten Methoden aus der Klasse Class sammelt. Diese Methoden werden in der Methode ObjectBrowser.searchThroughClass zusammengetragen und in der ArrayList ObjectBrowser.decMeth gespeichert.
Eine Methode ObjectBrowser.reflectObject holt nun möglichst viele Daten aus einem Objekt indem es über das Feld ObjectBrowser.decMeth iteriert.
Die so gesammelten Daten in verschiedenen Arrays sind nun für jede View abrufbar.

Um spezifische Daten eines Feldes oder Methode zu bekommen dienen die Methoden ObjectBrowser.zoomInField und ObjectBrowser.zoomInMethod, welche direkt einen String zurückgeben, um ihn in der View Abbilden zu können.

Nachträgliche Bemerkung:

Da ich in den genannten Methoden einen Filter bzw. eine Begrenzung des Datensammelns einbauen musste, aber nur aus diesem Grund, hält das Model ObjectBrowser eine Referenz auf die View ObjectView. Dies wäre nicht nötig, wenn keine Rückgabewerte aus den Methoden und Feldern abgefragt werden würden. An dieser Stelle habe ich keine Lösung finden können, falls sie einen Tipp haben wie es zu lösen ist würde ich mich freuen.

View:

Es wird eine View gebraucht, die die gesammelten Daten abbildet.
Dazu wird ein JTree zur Navigation und eine JTextArea zum Darstellen der Eigenschaften genutzt. Bei dem Versuch null zu inspizieren wird ein MessageDialog als Fehlerausgabe dargestellt.
Die Methode ObjectView.createObjectTree erstellt den Baum mit zugehörigen Listenern – ExpansionListener, SelectionListener, die beim Erweitern bzw. beim Auswählen eines Nodes zum Sammeln der Daten dienen. Es muss für jedes Node, das Element der Listener ist, unterschieden werden welchem Typ es angehört.

Test-Klasse:

Die Klasse Clazz dient ausschließlich zum Testen der Implementierung des Objektbrowsers.
Die Klasse TestAnno wurde ebenfalls nur zum Testen erstellt.

Anwendungseinstig:

Die Klasse MainApp fungiert wie immer mit der Main-Methode als Einstiegspunkt. In der Main wird der Thread für den Objektbrowser erzeugt und versucht ihn zu starten. Hier gilt es nun auch das zu inspizierende Objekt anzugeben.s

Thread:

Die Klasse ObjApp implementiert Runnable und somit die Methode run, damit man sie als eigenständigen Thread starten kann. Nun kann man hier das zu inspizierende Objekt anzugeben.

ThreadController:

Diese Klasse dient nur zum prüfen der laufenden Threads.

UML:

Siehe a09 package.