Vorgehensweise und Entscheidungsfindung

Um bei dieser Aufgabe ein wenig Arbeit zu sparen und mal Code wiederzuverwenden, haben wir uns dazu entschieden, a03 zu refactoren.

An das zu inspizierende Objekt kommen wir über einen Konstruktor. Hier haben wir derzeit drei Objekte, mit denen wir unter Anderem getestet haben. Die Klasse DummyClass haben wir zum systematischen Testen unserer Implementierung gebaut.

Die buildFrame() Methode haben wir direkt übernommen. Die Methode buildExplorerTree() bekommt unser Objekt übergeben. Hier wird dann die rootNode vom Objektbaum angelegt mit dem Namen unseres Objekts. Anschließend holen wir uns die Methoden und Felder der Klasse. Nun erstellen wir den Baum, die Methode convertValueToText von JTree mussten wir überschreiben, damit wir selber bestimmen können, wie die Nodes benannt werden, ob entweder das standardmäßige toString erfolgen soll oder eben eine Sonderbehandlung.

Zum treeSelectionListener ist noch der treeExpansionListener gekommen gekommen. Wenn wir rekursiv in die Methoden und Felder absteigen, haben wir sonst das mögliche Problem einer Endlosschleife, dass beim Start des Programms versucht wird, durch alle Felder immer weiter hinabzusteigen. So geschieht das nur dann, wenn ein Objekt/Feld selektiert und der Baum ausgeklappt wird.

Beim „hineinzoomen“ in die Felder des Objekts wird dann auf primitive Typen geprüft und anschließend werden Methoden und Felder geholt und dem Tree hinzugefügt.

Bei den Methoden haben wir überlegt, wie wir dort weiter absteigen sollen. Der Rückgabewert hängt ja von den Übergabeparametern und dem Inhalt der Methode selber ab, wir müssten also entsprechende Werte in die Übergabeparameter einsetzen, um den Rückgabetyp/das Rückgabeobjekt zu untersuchen. Das war uns allerdings nicht einfach möglich. Wir könnten natürlich wieder genau wie bei den Feldern nur die Rückgabetypen inspizieren, ob sie primitiv sind oder eben nicht und dann selbiges wie für Felder ausgeben.

In der Methode addMethodsAndFields holen wir uns zunächst alle Felder, ist ein Feld nicht greifbar, verschaffen wir uns Zugang mittels setAccessible(true) (ohne böswillige Absichten ☺).

Die Methoden zum Holen der Objektinformationen haben wir in die Helferklasse ObjectInspectHelper ausgelagert.

Die innere Klasse FieldAndValue nutzen wir, um den Wert des Feldes und dazu noch seinen "Field"-Datentyp hineinzuwerfen. Wir wollten ja dann auch die Felder und Methoden dieses Wertes haben. Damit wir uns dann nicht "hochhangeln" müssen, haben wir den Wert einfach gleich rausgeholt und zusammen mit Field gemerkt.

Martin Slowikowski Jan-Tristan Rudat

Matrikelnummer: 1999 166 Matrikelnummer: 2007 852