Übungen zur Vorlesung
„Praktische Informatik 2“

AG Betriebssysteme, FB 3
Jan Bredereke

SoSe 2002

ausgegeben: 27.5.2002

Blatt 3

Reflektion in Java

Aufgabe 1: Erforschung des Marsianers (100%)

Auf dem Dach des MZH ist gestern eine fliegende Kaffeetasse gelandet. Studentischen Hilfskräften des Studiengangs Biologie gelang es, den aussteigenden Marsianer in einem großen Eimer einzufangen. Den Professoren des Fachbereichs 2 glückte es allerdings nicht, irgend etwas über seine Eigenschaften in Erfahrung zu bringen. Eine der studentischen Hilfskräfte fand lediglich heraus, daß der Marsianer aus 100% reinem Java besteht.

Eure Aufgabe ist es, ein Forscher-Programm zu schreiben, das alles über den Marsianer herausfindet. Der Forscherroboter wird anschließend von uns aus Sicherheitsgründen mit dem Marsianer zusammen in eine Java-Virtual-Machine gesperrt und soll von dort berichten, welche Eigenschaften der Marsianer hat.

Schreibt eine Klasse `Forscher`, die zum `package blatt3` gehört. Sie soll den Marsianer, der aus einer einzigen Java-Klasse besteht, untersuchen. Dafür soll die Funktionalität aus dem Paket `java.lang.reflect` verwendet werden. Die Klasse `Forscher` soll durch den Aufruf ihrer Methode `static void main(String[] args)` aktiviert werden, wobei der erste und einzige Parameter den vollen Namen der Klasse des Marsianers einschließlich des Paketnamens enthält, also z.B. `"ujo.Marsianer"` („ujo“ steht für „unbekanntes Java-Objekt“).

a) Wie sieht der Marsianer aus?

Der Forscher soll als erstes ausgeben, von welcher *Superklasse* die Marsianer-Klasse ggf. abgeleitet ist und welche *eigenen Feldvariablen, Konstruktoren* und *eigenen Methoden* die Marsianer-Klasse hat.

Dabei soll er zusätzlich für jede dieser Feldvariablen, Konstruktoren und Methoden die *Modifikatoren* (`public`, `protected`, `private`, `static`, `final` ...) ausgeben. Für jede dieser Methoden und für jeden Konstruktor soll er die Namen der Typen der *Parameter* ausgeben. Auch die Namen aller *Exceptions*, die eine Methode oder ein Konstruktor werfen kann, sollen aufgelistet werden. Für die Methoden soll natürlich auch der Name des Typs des *Rückgabewertes* ausgegeben werden. Die Ausgabe soll übersichtlich formatiert mit `System.out.println()` geschehen.

Es sollen nur die eigenen Methoden und die eigenen Feldvariablen ausgegeben werden, nicht auch die geerbten. Verwendet also nicht `getMethods()` und `getFields()` der Klasse `Class`, sondern `getDeclaredMethods()` und `getDeclaredFields()`. Dagegen sollen ggf. auch die geerbten Konstruktoren ausgegeben werden. (Damit ist sichergestellt, daß auch dann ein Konstruktor in der Liste ist, wenn von der Klasse kein eigener definiert wird.)

Die Modifikatoren könnt ihr jeweils in einfacher Weise für die Ausgabe in einen String verwandeln, indem ihr die Methode `toString(int mod)` der Klasse `Modifier` verwendet.

b) Wie verhält der Marsianer sich?

Weiterhin soll jeder *Konstruktor* und jede eigene, öffentliche (d.h. `public`) *Methode aufgerufen* werden, und zwar in allen möglichen *Paarungen* aus je einem Konstruktor und je einer Methode.

Sofern die jeweilige Methode einen *Rückgabewert* hat, soll der von der Methode am Ende des Aufrufs zurückgegebene Wert mithilfe der Methode `toString()` ausgegeben werden. Falls der aufgerufene Konstruktor oder die aufgerufene Methode eine *Exception* wirft, sollt Ihr sie fangen und diese Tatsache sowie den Namen der Exception und den mit `getMessage()` eventuell verfügbaren Text ausgeben.

Falls für den Aufruf einer Methode oder eines Konstruktors Parameter notwendig sind, müßt Ihr geeignete beliebige Werte bereitstellen. Dafür könnt Ihr den Konstruktor des jeweiligen Parameter-Objekts aufrufen.

Ihr dürft zur Vereinfachung annehmen, daß stets ein parameterloser Konstruktor für die Objekttypen verfügbar ist. Weiterhin dürft Ihr annehmen, daß nirgendwo Arrays von Objekten vorkommen. Schließlich dürft Ihr annehmen, daß alle Konstruktoren des Marsianers öffentlich sind.

Anmerkung: Als Parameter kann auch einer der acht primitiven Typen (`int`, `boolean`, `char`, ...) verlangt sein. In diesem Falle bekommt ihr in der Liste der Parametertypen die Werte `Integer.TYPE`, `Boolean.TYPE`, `Character.TYPE`, ... Erzeugt man Instanzen dafür, bekommt man Objekte der Klassen `Integer`, `Boolean`, `Character`, ... Diese Objekte werden bei Verwendung als Parameter ggf. automatisch zurück in die primitiven Typen umgewandelt. Ihr müßt den Sonderfall der primitiven Typen also nicht anders behandeln als den allgemeinen Fall der Objekte und habt keine Arbeit damit.

Aus dem Paket `java.lang.reflect` benötigt Ihr insbesondere die Klassen `Class`, `Field`, `Constructor`, `Method` und `Modifier`. Diese sind wie üblich in der Java-API-Dokumentation beschrieben, die über unsere Web-Seiten zu finden ist. Außerdem findet Ihr dort das Java-Tutorial, dessen „Specialized Trail“ über „Reflection“ ausführliche Beispiele bringt.

Es wird vermutlich nicht möglich sein, Euren Forscher auf sich selbst anzuwenden, da einige Parameter einiger Methoden Arrays sein werden, mit denen der Forscher nicht umgehen können muß. Schreibt also besser eigene einfache Test-Marsianer. Dies kann auch parallel zum Schreiben des Forschers geschehen. Schließlich ist der Marsianer vom Dach des MZH auch bereits vor 517 Jahren von einem nordmarsianischen Binärbaum gepflückt worden, lange bevor er nach Bremen kam.

Bei dem Entwurf von Test-Marsianern ist es übrigens nicht verboten, besonders phantasievolle Methodennamen usw. zu wählen! `;-)`

Abgabe: 3.–6. Juni 2002 in den Übungen bzw. beim Tutor

Die Abgabe soll sowohl elektronisch (Programm-Quellcode) als auch in gedruckter Form (mit LaTeX gesetzter kommentierter und erläuterter Quellcode) erfolgen. Dabei ist auch auf geeignete Testfälle und deren Dokumentation zu achten!