PR2-Aufgabe Roboterfabrik

(ursprüngliche Autoren: Sebastian Tschirpke und Steffen Hennhöfer; mit Erweiterungen von Oliver Hummel)

Nach Abschluss Ihres Bachelorstudiums sind Sie auf Jobsuche. In einem Zeitungsartikel lesen Sie, dass die Firma ASDF123-Roboter AG, junge und kreative Mitarbeiter für den Aufbau einer Roboterfabrik sucht. Von der Stellenausschreibung ganz hingerissen, machen Sie sich sofort auf den Weg und reichen Ihre Bewerbungsunterlagen ein. Tatsächlich meldet sich die Firma kurz darauf bei Ihnen und lädt Sie zu einem Bewerbungsgespräch ein. Für Sie läuft scheinbar alles perfekt und Sie können es kaum erwarten Ihren zukünftigen Arbeitgeber von Ihren Qualitäten zu überzeugen. Jedoch zeigt dieser keinerlei Interesse an Ihren Zeugnissen, sondern knüpft vielmehr Ihre Einstellung an die Lösung folgender Aufgabe für die Sie Klassen auf der Domänenebene implementieren und testen sollen:

Die Firma möchte eine neue Roboterfabrik aufbauen und benötigt dazu ein Softwaremodell, welches den Produktionsprozess und die Roboter simuliert. Es handelt sich nur um eine einzige Fabrik mit einer Fabrikmethode, die flexibel auf Bestellungen der Kunden reagieren kann. Die Roboterfabrik bietet ihren Kunden zunächst zwei verschiedene Modelle an, beide aus der Star-Wars-Produktreihe: die Modelle heißen C3PO und R2D2 und sollen gemeinsam genutzte Funktionalität von einer Superklasse erben.

Alle Roboter haben eine einfache Schnittstelle Robot, mit der sie gesteuert werden können. Diese Schnittstelle besteht insgesamt aus den folgenden Dateien:

- Robot.java
- RobotControl.java
- RobotInstructions.java

Die Fabrik ist von ihren Fertigungsanlagen her so konzipiert, dass jeder Roboter einzeln gefertigt wird. Übergeben wird lediglich das gewünschte Roboter-Modell und der Name und schon erhält der Kunde den von ihm gewünschten Roboter (also in der Simulation eine Instanz).

Der Unterschied zwischen den angebotenen Modellen liegt in der Ausführung der Befehle aus dem Interface RobotInstructions. Während R2D2 Arrays immer aufsteigend sortiert und bei der Umwandlung in ein Array den Inhalt durch Kommas trennt (z.B. 1, 2, 3, 4, 5, 6), sortiert C3PO Arrays stets absteigend und wandelt deren Inhalt in einen String mit Semikolon als Trennzeichen um (z.B. 6; 5; 4; 3; 2; 1). R2D2 soll für die Sortierung der Daten einen selbst implementierten SelectionSort-Algorithmus verwenden, C3PO InsertionSort. Die Zusammenstellung des Strings für die Rückgabe soll in einer privaten Methode der Roboter-Basisklasse mit Hilfe eines Streams und Lambda-Ausdrücken erfolgen.

Programmieren Sie zwei Klassen R2D2 und C3PO, die beide das Interface Robot implementieren. Lesen Sie bitte sorgfältig die Dokumentation der Interfaces, damit Sie die beschriebene Semantik in Ihrer Implementierung einhalten. Die Seriennummern der R2D2-Modelle bewegen sich im Bereich von 0–9999, die der C3PO-Modelle von 10000–19999. Schreiben Sie ferner die in den Interfaces referenzierten Klassen für die Ausnahmen. Sorgen Sie dafür, dass die Ausnahmen den Namen des Roboters tragen, in dem sie entstanden sind und dass man diesen Namen über die Methode getRobotName() wieder auslesen kann.

Auch für die Implementierung der Exceptions kann daher eine gemeinsame Superklasse hilfreich sein.

Entwickeln Sie eine Klasse RobotFactory mit deren Hilfe man Instanzen der beiden Roboter-Typen erzeugen kann. Eine Enumeration mit dem Namen RobotType dient dazu, bei der Erzeugung anzugeben, ob man einen R2D2- oder einen C3PO-Roboter haben möchte.

Vergessen Sie nicht, dass jeder Roboter eine Seriennummer aus dem jeweiligen Bereich benötigt und dass er bei der Erzeugung seinen unveränderlichen Namen bekommt. Verbergen Sie die Implementierung der Roboter vor dem Verwender und erlauben Sie nur Zugriff auf die Factory, die Interfaces und die Ausnahmen. Wählen Sie entsprechende Pakete gemäß ihrem Schema, um dies zu realisieren.

Zusätzlich zu den beiden produzierten Roboter-Modellen R2D2 und C3PO steht in der Firma noch ein alter Nexus-6-Roboter mit dem Namen “Pris“ (Seriennummer 19281982) herum, der leider seit einer Begegnung mit einem Blade-Runner irreparabel defekt ist und nicht eingeschaltet werden kann (man kann den Schalter zwar drücken, dies hat aber keine Wirkung). Da es nur dieses eine Exemplar gibt, können auch keine weiteren Nexus-6-Modelle hergestellt werden. Implementieren Sie daher den Nexus-6-Roboter (Klassenname Nexus6) als Singleton. Beachten Sie, dass die speak- und think-Methoden nicht funktionieren sollen, sondern grundsätzlich eine Ausnahme Robot"-Illegal"-State"-Exception werfen.

Schreiben Sie automatisierte JUnit-Tests mit denen Sie die korrekte Funktionsweise der Implementierungen und der Factory-Klasse überprüfen. Denken Sie daran, auch die Ausnahmen zu testen.

Angenommen, Ihre Firma möchte eine weitere Produktlinie eröffnen und in Zukunft auch noch T1000-Roboter herstellen, die natürlich auch das Robot-Interface implementieren: Welche Änderungen müssten Sie an Ihrem Projekt durchführen? Sie brauchen keine Implementierung dafür zu erstellen, legen Sie dafür bitte eine Textdatei in Ihr Projekt, in der Sie die notwendigen Erweiterungen und evtl. Änderungen kurz erläutern.

- Mit Ihrer Implementierung sollten folgende Begrifflichkeiten abgedeckt sein: Singleton-Pattern, Factory-Pattern, Exceptions, Interfaces, abstrakte Klassen, JUnit-Tests, Enumerationen, Sortieralgorithmen, Streams & Lambdas.
- Achten Sie bei Ihrer Implementierung auf die Konsistenz der Sprache (Englisch/Deutsch).
- Kommentieren Sie Ihre Methoden ausführlich mit Javadoc-Kommentaren, wie in den Interfaces gezeigt (googlen Sie ggf. nach weiteren Details dafür). Bei der Implementierung von Methoden aus den Interfaces, dürfen Sie mit @see... auf deren Javadocs verweisen und müssen die Dokumentation nicht duplizieren.
- Sie müssen kein UML-Diagramm anfertigen, können dies aber gerne tun.
- Testen Sie Ihre Implementierung möglichst umfangreich. Wir werden es auf jeden Fall tun. 😉
- Nutzen Sie für die gemeinsame Arbeit an der Implementierung ein Git-Repository und nutzen Sie regelmäßig. **Es müssen von allen Team-Mitgliedern jeweils mindestens fünf Commits mit substanziellem Inhalt auf den main-Branch gepusht werden.**