Übungen – Pakete, Testen, Auslieferung

Aufgabe 1 – Pakete

Überlegen Sie sich eine sinnvolle Paketstruktur für die bisher geschriebene Software, v.a. für die Warenbestellung und die Fahrzeug-/Halter-Hierarchie.
Wählen Sie einigermaßen korrekte Paketnamen und verschieben Sie die Klassen.
Versuchen Sie, zunächst dies auf Dateisystem-/File-Basis zu machen. Wenn Sie sich sicher im Umgang mit import und Deklarationen/Lokationen fühlen, können Sie die Möglichkeiten Ihrer IDE ausloten, bei der so etwas komfortabler erreicht werden kann.

Aufgabe 2 – Tests

Überlegen Sie sich sinnvolle Tests für die Warenbestellung und die Fahrzeug-Hierarchie, ggf. auch für die Bruchzahl. Schreiben Sie diese Tests als JUnit-Tests.

Aufgabe 3 – Teamarbeit

Jetzt wird's mal interaktiv… Suchen Sie sich Ihre Nebensitzerin/Ihren Nebensitzer oder eine kleine Gruppe. Jeder macht nun folgendes:

1. Schreiben Sie eine Hilfsklasse zur Berechnung der Fakultät n! und zur Berechnung der statistisch so beliebten „über"-Funktion $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
   Sie brauchen den Code nicht übermäßig zu testen, das wird gleich von Ihren Kollegen erledigt.
2. Dokumentieren Sie diese Funktionen.
3. Geben Sie Ihren Code in der Gruppe weiter.
4. Arbeiten Sie zunächst Blackbox-Tests für den erhaltenen Code aus und schreiben Sie nur Tests für das, was Sie in der Dokumentation lesen bzw. abfragen können.
5. Machen Sie nun einen Glassbox-Ansatz. Schauen Sie sich den Code an. Hätten Sie andere Tests bei Kenntnis des Codes entwickelt? Wenn ja, ziehen Sie die Tests nach.
6. Stellen Sie nun sicher, dass Ihre Tests zumindest eine Zweigüberdeckung realisieren.

Aufgabe 4 – Auslieferung

1. Erstellen Sie eine JAR-Datei für die Warenbestellung, die Sie per Kommandozeile ausführen.
2. Erstellen Sie eine JAR-Datei, die direkt gestartet werden kann (`java -jar`). Erstellen Sie diese JAR einmal mit Ihrer IDE und einmal mit dem Kommandozeilentool `jar` selbst.
3. Verteilen Sie Ihre Pakete des Warenbestellsystems in verschiedene JAR-Dateien. Wie bekommen Sie Ihren Code jetzt zum Laufen?
4. Für Detektive: versuchen Sie, einen `NoSuchMethodError` zu provozieren.

Aufgabe 5 – Annotations

Führen Sie in Ihre bisherige Software noch die Annotations `@Override` und `@SuppressWarnings` ein, wo sie sinnvoll sind.