Einleitungsstory:

„So vermehrt sich das gemeine Minion: Sie werden als beschworene Wesen immer dann geschaffen wenn sich ein Minion über eine faulige Banane ärgert und sich jemanden wünscht, an dem es das auslassen kann. Dann teilt sich das wütende Original und erschafft so eine Kopie von sich selbst. Das erklärt auch, warum sie kein Problem damit haben, sich gegenseitig zu hauen – sie tun ja niemand anderem weh.“ (*Prof. J.R.R. Pfitzelhuber, 1985 vor Kirk, „Der Boogeyman und andere Ungeheuer“)*

Als Experte für unkonventionelle Lösungen haben Sie sich bereits einen soliden Ruf erarbeitet, als eines Tages Edith, Adoptivtochter des Superschurken Gru an Ihre Tür klingelt. Da Gru und sein Wissenschaftler Dr. Nefario es nicht so mit Ordnung haben, quillt die Villa von Gru inzwischen über vor kleinen gelben Minions. Edith möchte Sie engagieren, um die Minions zu zählen und zu ermitteln, wie viele frische Bananen pro Tag gebraucht werden, damit sie sich nicht weiter vermehren. Da die Minions sich bei spektakulären Industrieunfällen immer mal selbst reduzieren, sollte sich so das Platzproblem auf Dauer lösen lassen.

Aufgabe1 – Von Daten zur Liste

Dr. Nefario hat die Minions schon gezählt. Da die aber nicht stillhalten wollen, sind ihm Duplikate untergekommen und ungeordnet sind sie auch. Aber er hat dazu wenigstens seine eigene Klasse „Minion“ verwendet.

Übertragen Sie Dr. Nefarios Datensätze in eine Liste und nutzen Sie dabei das Comparable-Interface, um sie nach ihrer evilNumber zu sortieren. Was müsste man tun, um die Duplikate aus der Liste zu entfernen?
Da Sie gute Programmierung gelernt haben, überarbeiten Sie natürlich die Klasse, überschreiben dabei equals() und hashCode() und machen sie auch immutable.

Aufgabe2 – Was sind die Unterschiede?

Irgendwie ist Ihnen das Entfernen der Duplikate mit der Liste zu aufwendig. Sie wissen, dass das einfacher geht: Erklären Sie den Unterschied zwischen TreeSet und HashSet. Warum kann man nicht einfach *Set<Minion>= new Set<>();* verwenden?

Aufgabe 3 – Von Liste zu Ordnung

Sie entscheiden sich für ein TreeSet. Edith erklärt Ihnen, dass die violetten Minions keine Bananen brauchen, da sie sich von Möbelstücken ernähren. Übertragen Sie nur die Daten der gelben Minions von der Liste in ein TreeSet und erklären Sie, was mit den Duplikaten passiert. Schreiben Sie die endgültige Liste in eine .txt-Datei, die Edith als Checkliste für die Bananen-Ausgabe verwenden kann, nachdem Sie toString() angepasst haben. Geben Sie die Anzahl an benötigten Bananen pro Tag an.

Unit-Tests

Überprüfen Sie die Funktionalität Ihrer Implementierung mit entsprechenden JUnit-Tests und weisen Sie mit diesen Tests nach, dass die implementierten Operationen richtig funktionieren.

Achtung

Bitte beachten Sie folgendes, damit es nicht zu unnötigen Punktabzügen in der Bewertung kommt:

- Benennen Sie Klassen und Methoden konsistent und verständlich. Klassen sollten Nomen als Namen,
- Methoden Verben haben.
- Dokumentieren Sie alle Klassen, Interfaces, Konstruktoren und Methoden mit JavaDoc. Dokumentieren Sie auch alle Parameter, Rückgabewerte und Ausnahmen.
- Formatieren Sie Ihren Code konsistent. Ein guter Standard sind 4 Spaces Einrückung pro Ebene ohne Verwendung von Tabulatoren.
- Halten Sie sich an die Sun Java-Code-Convention (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)
- Programmieren Sie defensiv und testen Sie Eingabewerte auf deren Gültigkeit. Ihr Programm darf
- auch mit Daten nicht abstürzen, die außerhalb der Aufgabenstellung liegen.
- Machen Sie keine Konsoleneingaben oder -Ausgaben, es sei denn die Aufgabe fordert dies explizit.
- Halten Sie Daten und Methoden zusammen. Trennen Sie diese nicht unnötig auf.
- Kopieren Sie keinen Code sondern versuchen Sie mit den bekannten Mitteln der Objektorientierung Code-Duplikate zu vermeiden.