Übungsblatt 1

Abgabe: 26.04.2024

Ablauf

Die Übungsblätter sind in Zweiergruppen gemeinsam zu bearbeiten. Eine Person macht dabei die Implementierung, die andere die Tests. Dies wird getrennt bewertet. Beide Teile sind mindestens in JavaDoc und bei Bedarf zusätzlich mit weiteren Kommentaren und in LaTeX zu dokumentieren. 80 % der Punkte gibt es jeweils für die Implementierung bzw. Tests, 20 % für die Dokumentation. Es ist für jede Abgabe ein LaTeX-Dokument zu erstellen, das auf der Vorlage *pi2.cls* basiert, in das alle erstellten Quelltexte geeignet einzubinden sind.[1]

Bei den erstellten Tests wird grundsätzlich erwartet, dass alle positiv durchlaufen, eine Code Coverage von 100 % erreicht wird und das PIT-Plugin alle Mutationen als erkannt attestiert. Dies ist zu dokumentieren. Es kann sein, dass sich dies nicht immer erreichen lässt. In dem Falle wird eine Erklärung erwartet, warum dies nicht möglich ist.

Auf diesem Übungsblatt macht die Person die Implementierung, die in einem Telefonbuch zuerst gelistet würde. Die andere Person macht die Tests. Dies wird sich in weiteren Übungsblättern jeweils umkehren.

Richtet unter *gitlab.informatik.uni-bremen.de* ein Repository *pi2-2024* ein und ladet eure Tutor:in dazu als *Developer* ein. Legt eine geeignete *.gitignore*-Datei im Hauptverzeichnis des Repositories ab.[2] In dem Repository wird jede Abgabe in einem eigenen Unterordner abgelegt (*loesung1*, *loesung2* usw.).

Aufgabe 1 Eine Menge Generisches

Vervollständigt die generische Klasse *Set<E>*, die eine Menge von Objekten repräsentiert. Speichert die Daten intern in einem normalen Java-Array, dessen Elemente ihr jeweils in ein größeres kopiert, wenn ein neues Element hinzukommt. Die folgenden Operationen müssen implementiert und getestet[3] werden:

Aufgabe 1.1 Anzahl der Elemente (10 %)

Die Methode *int size()* liefert die Anzahl der Elemente, die in der Menge enthalten sind.

Aufgabe 1.2 Aufzählen der Elemente (30 %)

Die Menge implementiert das Interface *Iterable*, so dass man alle Elemente der Menge aufzählen kann. Für den *Iterator<E>* müssen nur die Methoden *hasNext* und *next* implementiert werden.

[1] Eure Tutor:in kann wahlweise auch darauf verzichten, wenn ihr die dokumentierten Quelltexte ausreichen. Es muss aber immer klar sein, wer die Bearbeiter:innen einer Abgabe sind.

[2] Z.B. die, die als *gitignore.txt* auf Stud.IP zur Verfügung steht.

[3] Da es keinen Zugriff auf den internen Zustand der Menge gibt, überprüfen Tests immer die Interaktion verschiedener Methoden, z.B. beeinflussen Aufrufe von *add*, was *size* danach zurückgibt.

Die Implementierung des Interfaces erlaubt die Aufzählung der Elemente in einer *for(:)*-Schleife, was auch in weiteren Methoden der Klasse selbst genutzt werden kann (*for(: this)*).

Aufgabe 1.3 Test auf Vorhandensein (10 %)

Die Methode *boolean contains(E)* gibt zurück, ob das übergebene Element in der Menge enthalten ist. Vergleiche werden über *equals* durchgeführt.

Aufgabe 1.4 Einfügen eines Elements (20 %)

Die Methode *void add(E)* fügt ein übergebenes Element in die Menge ein, wenn es nicht bereits vorhanden ist. Ob es bereits vorhanden ist, kann mit *contains* überprüft werden. Das Einfügen von *null* wird abgelehnt.

Aufgabe 1.5 Schnittmenge (10 %)

Die Methode *Set<E> intersect(Set<E>)* liefert die Schnittmenge ($A \cap B$) aus dieser Menge und einer weiteren Menge zurück, also alle Elemente, die in beiden Mengen vorhanden sind. Die ursprünglichen Mengen werden dabei nicht verändert.

Aufgabe 1.6 Vereinigungsmenge (10 %)

Die Methode *Set<E> union(Set<E>)* liefert die Vereinigungsmenge ($A \cup B$) aus dieser Menge und einer weiteren Menge zurück, also alle Elemente, die in mindestens einer der beiden Mengen vorhanden sind. Die ursprünglichen Mengen werden dabei nicht verändert.

Aufgabe 1.7 Differenzmenge (10 %)

Die Methode *Set<E> diff(Set<E>)* liefert die Differenzmenge ($A \backslash B$) aus dieser Menge und einer weiteren Menge zurück, also alle Elemente, die in dieser Menge vorhanden sind, in der anderen aber nicht. Die ursprünglichen Mengen werden dabei nicht verändert.