Tagesprogramm

Software-Dokumentation und Zusicherungen

Seiteneffektfreie Programme

Wo Information zu finden ist

Kopf von Klasse, Interface:

Zweck, Grobstruktur, Besonderheiten,
wer ist verantwortlich, wer sind die Benutzer

Kopf von Methode, Konstruktor:

Zweck, Vor- und Nachbedingungen (Parameter, Verhalten, Ergebnis)

Variablendeklaration:

Zweck, Invarianten (Konsistenz-Bedingungen)

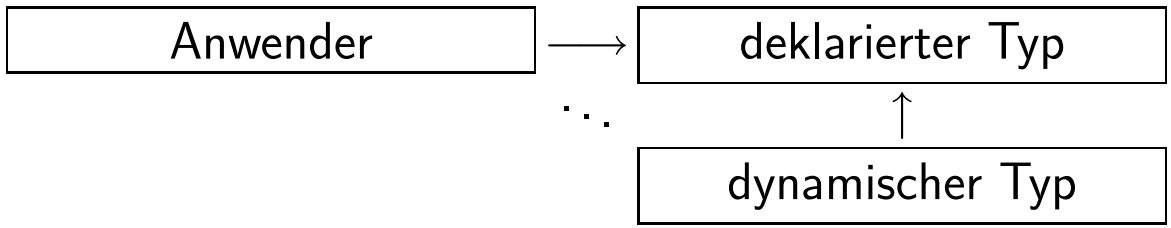
in Implementierungen:

komplexe Schleifeninvarianten,
Hinweise auf Programmänderungen

Ersetzbarkeit und Verhalten

Methoden in Untertypen müssen sich so verhalten,
wie von Methoden in Obertypen erwartet

Zusicherungen in Untertypen spezifizieren selbes Verhalten wie in Obertypen,
können das Verhalten in Untertypen aber genauer spezifizieren



Zusicherungen und Untertypen

Vorbedingung einer Methode:

im Untertyp gleich oder schwächer als in Obertyp

Nachbedingung einer Methode:

im Untertyp gleich oder stärker als in Obertyp

Invariante einer Klasse / eines Objekts:

im Untertyp gleich oder stärker als in Obertyp

Aufgabe: Ersetzbarkeit

Such Sie in Gruppen zu zwei bis drei Personen Antworten auf folgende Frage:

In welchen Fällen und wozu benötigt man Ersetzbarkeit?

Zeit: 2 Minuten

Funktionaler Programmierstil

Verzicht auf Seiteneffekte $\longrightarrow$ Programm(teil)e einfacher verständlich

Programm(teil)e ohne Seiteneffekte $\longrightarrow$ funktionaler Programmierstil

funktionale Faktorisierung $\longrightarrow$ einander aufrufende reine Funktionen

Aufgabe: Zusicherungsart

Welche der folgenden Aussagen treffen für `FuncList.valueAt` zu?

A „element at index 'index' if the list has ...“ ist eine Nachbedingung

B „index >= 0 required“ ist eine Invariante

C „index >= 0 required“ ist eine Nachbedingung

D „IllegalArgumentException is thrown if ...“ ist eine Vorbedingung

Aufgabe: Objekterzeugung

Warum ist der Konstruktor in `EmptyFuncList` als `private` definiert?

A um sicherzustellen, dass nur das Interface ein Objekt der Klasse erzeugt

B um sicherzustellen, dass es keinen Default-Konstruktor gibt

C um sicherzustellen, dass keine zweites Objekt der Klasse erzeugt wird

Aufgabe: equals und hashCode

Warum sind `equals` und `hashCode` in `EmptyFuncList` nicht implementiert?

A Die Default-Implementierungen passen da es nur ein Objekt gibt.

B Diese Methoden werden ohnehin nicht aufgerufen.

C Die Implementierungen werden von `NonEmptyFuncList` geerbt.

Aufgabe: Funktionale Objekte

Suchen Sie eine Standard-Java-Klasse, die wie `FuncList` ganz ohne Änderungen von Objektzuständen auskommt.

Warum wurde für diese Klasse eine solche Implementierung gewählt?