J. Dietel. J. Fuchs FH Aachen, Campus Jülich; IT Center, RWTH Aachen

Hausaufgaben 11 18./19.12.2024

Abgabe der Lösung am 07.01.2025

StringBuilder vs. String (Revisited)

Mit dem Wissen aus der Vorlesung möchten wir die Aufgabe aus dem Hausaufgabenblatt 3 besser lösen. Dafür gibt es zwei Möglichkeiten, die beide implementiert werden sollen:

a) mit einer abstrakten Klasse, die aus folgenden Methoden besteht:

```
public abstract void doSomething();
public double measureTime(){
	double start = System.nanoTime();
	doSomething();
	double end = System.nanoTime();
	return end-start;
}
```

Schreiben Sie dazu eine Klasse `Tester`, die neben `main` auch noch die Funktion `testWithAbstractClass` enthält. In dieser Funktion soll mit anonymen inneren Klassen die Laufzeit der einzelnen `StringBuilder` Operationen gemessen werden und mit äquivalenten `String`-Operationen verglichen werden. Es werden also zwei anonyme Klassen benötigt, um eine Operation zu vergleichen. In den Implementierungen der abstrakten Klasse als anonyme innere Klasse soll nur die `doSomething`-Methode implementiert werden. Geben Sie die gemessenen Zeiten auf der Konsole aus. Nutzen Sie hardcodierte Pseudoattribute (siehe Vorlesung 16) in der Funktion `testWithAbstractClass` oder echte statische Attribute in der Klasse `Tester`, um notwendige Parameter wie einen `StringBuilder`, `index`, `String` oder `char` zu setzen.

Ein Aufruf der Funktion soll eine Ausgabe der folgenden Art erzeugen:

```
Test der abstrakten Variante:
Konstruktor StringBuilder 10900.0
Konstruktor String 12400.0
setCharAt(int, char) StringBuilder 11000.0
setCharAt(int, char) String 1.10792E7
delete(int, int) StringBuilder 23600.0
delete(int, int) String 253100.0
deleteCharAt(int) StringBuilder 8400.0
deleteCharAt(int) String 107900.0
insert(int, char) StringBuilder 18000.0
insert(int, char) String 1.25144E7
insert(int, char[]) StringBuilder 18400.0
insert(int, char[]) String 5488600.0
append(char[]) StringBuilder 17800.0
append(char[]) String 154200.0
```

b) mit einem funktionalen Interface names `Strategy` und einer Klasse, die folgende Funktion hat:

```
public static double measureTime(Strategy myInterface) {
      double start = System.nanoTime();
      myInterface.doSomething();
      double end = System.nanoTime();
      return end-start;
}
```

Erweitern Sie die Klasse `Tester` um die Funktion `testWithInterface`. In dieser Funktion soll mehrfach `measureTime` aufgerufen werden, jedes Mal mit einem Lambda, das die `StringBuilder`- oder `String`-Operation enthält. Geben Sie die gemessene Zeit auf der Konsole aus. Verwenden Sie auch hier, wie oben schon erwähnt, Pseudoattribute oder statische Attribute.

Ein Aufruf der Funktion soll eine Ausgabe der folgenden Art erzeugen:

```
Test der Interface-Variante:
Konstruktor StringBuilder 13200.0
Konstruktor String 4100.0
setCharAt(int, char) StringBuilder 14300.0
setCharAt(int, char) String 164900.0
delete(int, int) StringBuilder 13200.0
delete(int, int) String 31500.0
deleteCharAt(int) StringBuilder 7800.0
deleteCharAt(int) String 27300.0
insert(int, char) StringBuilder 8700.0
insert(int, char) String 309100.0
insert(int, char[]) StringBuilder 10700.0
insert(int, char[]) String 171100.0
append(char[]) StringBuilder 7600.0
append(char[]) String 31900.0
```