technische universität
dortmund

Bearbeitet von:
Sebastian Hauer
Sergej Karsten

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm,
Alexander Lochmann, Bartho Rudak, Thomas Schmitz

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2014/15

Übungsblatt 8

Bearbeitungszeit:
15.12.14 -
19.12.14

Aufgabe 8.1 – Organisation

Bearbeitungszeit: 20 Minuten

Vervollständigen Sie die Klasse `Calculator` aus dem Übungsblatt 7. Wenn Sie dies getan haben, erstellen Sie ein neues Paket namens `util` und verschieben Sie in Eclipse die Klasse in dieses Paket. Bestätigen Sie den Restrukturierungsdialog, damit Eclipse automatisch Anpassungen an den Paketinformationen in der verschobenen Java-Datei vornimmt.

Aufgabe 8.2 – Arrays und Methoden: Call by Value vs. Call by Reference

Bearbeitungszeit: 30 Minuten

Erstellen Sie eine neue Klasse mit dem Namen `ArrayTest` im Paket blatt08. Erstellen Sie in dieser Klasse zuerst die Skelette für folgende Methoden:

- Die **main**-Methode.
- Eine Methode namens **integerIncrement**, welche einen `int` als Parameter besitzt und nichts zurück gibt.
- Eine Methode namens **arrayIncrement**, welche ein `int`-Feld als Parameter besitzt und nichts zurück gibt.

Erstellen Sie im Paket `util` eine neue Klasse mit dem Namen `ArrayUtil` und erstellen Sie in dieser Klasse ein Skelett einer Methode mit dem Namen **printIntegerArray**, welche nichts zurück gibt und ein Integer-Feld als Parameter entgegen nimmt.

Implementieren Sie zuerst die Methode **printIntegerArray**, in welcher Sie über das übergebene Feld mit einer **for**-Schleife iterieren und den Inhalt des Feldes für einen Menschen lesbar ausgeben.

Bevor Sie das Programm testen, notieren Sie sich die erwartete Ausgabe ihres Programmes sowohl vor als auch nach den Aufrufen ihrer Unterprogramme:

Implementieren Sie anschließend die Methoden **integerIncrement** und **arrayIncrement** in der Klasse `ArrayTest`. Die Methoden sollen den übergebenen `int`, bzw. jeden `int` des übergebenen Feldes, um 1 inkrementieren.

Schreiben Sie anschließend in der **main**-Methode ein Programm, welches einen `int`, sowie ein `int`-Feld mit beliebigen Werten initialisiert und ausgibt. Nutzen Sie zum Ausgeben des Feldes das soeben von Ihnen geschriebene Unterprogramm der Klasse `ArrayUtil`.

Erweitern Sie das Programm anschließend um die Aufrufe der Methoden **integerIncrement** und **arrayIncrement** mit den im Hauptprogramm deklarierten Variablen als Argumenten. Geben Sie nach den Aufrufen der Unterprogramme den `int` und das Feld erneut aus.

Debuggen Sie das Programm zusätzlich wie auf Blatt 3 geübt. Verwenden Sie, wenn Ihr Programm vor Ausführung einer von Ihnen geschriebenen Methode hält, den Button „Step Into(F5)“, um in die Ausführung ihrer Methode hinein zu springen und das Unterprogramm zu debuggen. Alternativ können Sie auch einen Breakpoint in Ihrem Unterprogramm setzen und den Programmfluss bis zu dieser Stelle im Unterprogramm fortsetzen lassen (durch Betätigung des grünen „Abspielen“-Buttons in der Debug-Perspektive).

Aufgabe 8.3 – Quersumme

Bearbeitungszeit: 20 Minuten

Fügen Sie ihrer Klasse `Calculator` eine neue Methode hinzu, welcher die letzte Ziffer (im Dezimalsystem) eines übergebenen `int` zurück gibt.

Schreiben Sie anschließend eine neue Klasse mit dem Namen `DigitSum`. Implementieren Sie eine Methode **digitSum** in dieser Klasse, welche die Quersumme einer Zahl mithilfe der soeben von Ihnen implementierten Methode Ihrer Klasse `Calculator` berechnet und zurück gibt. Das Hauptprogramm dieser Klasse soll vom Benutzer eine Zahl anfordern und deren Quersumme ausgeben.

Aufgabe 8.4 – Fibonacci Laufzeit

Bearbeitungszeit: 50 Minuten

Erstellen Sie eine neue Klasse mit dem Namen `Fibonacci`. Schreiben Sie folgende Methoden:

- Eine Methode, welche die *n*-te Fibonacci-Zahl *iterativ* wie aus dem Übungsblatt 5 berechnet und als `long` zurückgibt.
- Eine Methode, welche die *n*-te Fibonacci-Zahl *rekursiv* wie aus dem Übungsblatt 7 berechnet und als `long` zurückgibt.
- Eine Methode, welche die *n*-te Fibonacci-Zahl *dynamisch* berechnet, indem Sie ein `long`-Feld der Größe *n* initialisieren und die Werte an den ersten beiden Stellen mit den ersten beiden Fibonacci-Zahlen belegen. Berechnen Sie anschließend die weiteren Fibonacci-Zahlen, indem Sie auf die bereits berechneten Werte im Feld zugreifen und im entsprechenden Index speichern. Geben Sie anschließend das gesamte Feld zurück.

Schreiben Sie anschließend ein Hauptprogramm, in dem die Laufzeit der einzelnen Methoden gemessen wird. Speichern Sie in einem `long` einen Zeitstempel mithilfe der Methode **System.nanoTime()** ab. Nach dem Aufruf einer Fibonacci-Methode mit der Zahl *n* soll die Differenz zwischen aktuellem und dem gespeicherten Zeitstempel von **System.nanoTime()** berechnet und ausgegeben werden. Die Differenz entspricht der vergangenen Zeit in Nanosekunden. Wiederholen Sie diesen Vorgang für jede Version der Fibonacci-Zahlen-Berechnung und notieren Sie die Laufzeit in folgender Tabelle:

n	**Iterativ**	**Rekursiv**	**Dynamisch**
2			
4			
8			
16			
32			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
64			
128			