Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Dipl.-Inform. Steffen Schiele
Dipl.-Inform. Andreas Bienert

Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik

Wintersemester 2014/15

Objektorientierte Programmierung: Hausaufgabenblatt 1

Abgabe: 30.10.2014, 12:00 — Übung: 03./04.11.2014

Hausaufgabe 1: (4 Theoriepunkte)

Schreiben Sie ein Programm, das eine gegebene natürliche Zahl in Primfaktoren zerlegt. Verwenden Sie das folgende Rahmenprogramm:

```
// Hausaufgabe 1: Primfaktorzerlegung
// Name: (bitte hier Ihren Namen eintragen)

import java.util.Scanner;

class PrimFak {
        public static void main(String[] args) {
                Scanner scan = new Scanner(System.in);
                System.out.print("Bitte n eingeben: ");
                int n = scan.nextInt();
                if(n <= 0) {
                        System.out.println("n muss positiv sein");
                        System.exit(1); // Programm-Abbruch
                }

                // Ab hier Ihr Programmcode:
        }
}
```

Verwenden Sie den folgenden Algorithmus zur Primfaktorzerlegung:

- In einer ganzzahligen Variable `p` speichern Sie den aktuellen Wert für den zu testenden Primfaktor. Starten Sie mit dem Wert 2.
- Wir werden `n` durch alle möglichen Primfaktoren dividieren. Dies soll so lange geschehen, bis `p` größer als `n` ist.
- Testen Sie jeweils, ob der aktuelle Wert von `n` durch `p` teilbar ist. Das können Sie machen, indem Sie mit `n % p` den Divisionsrest bestimmen. Ist dieser 0, so ist die Zahl teilbar.
- Falls `n` durch `p` teilbar ist, drucken Sie den Primfaktor `p` aus, und setzen `n` auf das Ergebnis der Division `n / p`. Den Wert von `p` ändern Sie nicht, da der gleiche Primfaktor mehrfach in `n` enthalten sein kann.

- Falls `n` nicht durch `p` teilbar ist, erhöhen Sie `p` um `1`. Es ist natürlich möglich, dass `p` dann gar keine Primzahl ist. Das ist aber kein Problem, da die in `p` enthaltenen Primfaktoren dann schon vorher getestet wurden. Es ist also garantiert, dass `n` dann nicht mehr durch `p` teilbar ist. Dies wird im nächsten Schritt festgestellt, und `p` dann weiter erhöht.

Beispiel:

- Angenommen, der Benutzer gibt 63 ein.
- Zuerst wird geprüft, ob 63 durch `2` teilbar ist. Das ist es nicht, deswegen wird `p` auf 3 erhöht.
- Nun wird geprüft, ob 63 durch 3 teilbar ist. Das ist der Fall, deswegen wird der Primfaktor 3 ausgegeben und `n` auf `63/3=21` gesetzt. Die Variable `p` bleibt auf 3.
- Nun wird getestet, ob 21 durch 3 teilbar ist. Das gilt wieder, also wird 3 nochmals ausgegeben und `n` auf $21/3 = 7$ gesetzt.
- Im nächsten Schritt wird geprüft, ob 7 durch 3 teilbar ist. Das ist nicht der Fall, also wird `p` auf 4 erhöht.
- 7 ist auch nicht durch 4 teilbar, deswegen wird `p` auf 5 erhöht.
- Ebenso ist 7 nicht durch 5 teilbar, also wird `p` auf 6 erhöht.
- Natürlich ist 7 auch nicht durch 6 teilbar (die Primfaktoren 2 und 3 wurden ja vorher schon getestet und abgespalten). Also wird `p` auf 7 erhöht.
- 7 ist durch 7 teilbar, deswegen wird 7 ausgegeben und `n` auf $7/7 = 1$ gesetzt.
- Nun ist `p` größer als `n`, also bricht die Schleife ab.

Schreiben Sie das Programm in die Datei "`PrimFak.java`". Geben Sie diese Datei über die Übungsplattform bis zum Donnerstag, 30.10.2014, 12:00 ab. Beachten Sie, dass es `0` Punkte gibt, falls das Programm nicht compilierbar ist (auf `anubis` mit `javac`). Verwenden Sie das gegebene Rahmenprogramm (ohne `package`-Deklaration). Geben Sie einen Primfaktor pro Zeile aus (nur die Zahl und kein Text). Wir verwenden eine automatische Vorkorrektur, deswegen halten Sie sich bitte genau an die Vorgaben. Bitte achten Sie auch auf sinnvolle Einrückungen und allgemein ein verständliches Programm.

Für schlechten Programmierstil können auch Punkte abgezogen werden. Insbesondere müssen Sie damit rechnen, dass ein Punkt abgezogen wird, wenn abhängige Anweisungen nicht tiefer eingerückt sind als das zugehörige `if` oder `while` oder der Methodenkopf. Anweisungen auf gleicher Schachtelungstiefe sollen gleich weit eingerückt sein.

Übungsaufgabe 1A: (ohne Abgabe)

Bitte bearbeiten Sie die Übungsaufgaben auf dem Hausaufgabenblatt, aber geben Sie diese Aufgaben nicht ab. Diese Aufgaben werden in der Übung besprochen. Sie müssen Ihre Lösung eventuell in der Übung vorführen.

In der Vorlesung und Übung wurde Ihnen bereits gezeigt, wie Sie mit Fehlermeldungen des Compilers umgehen. Schauen Sie sich die folgenden beiden Beispiele an. Versuchen Sie, den Fehler möglichst schon ohne Compiler zu finden. Geben Sie dann den Programmcode ein und rufen Sie den Compiler dafür auf. Erklären Sie, was die jeweilige Fehlermeldung besagt und korrigieren Sie den Fehler.

Listing 1: Fehlerhafter Code (Beispiel 1)

```java
class Fehler1
{
    public static void main(String[] arguments)
    {
        int a=42;
        System.out.println(b);
    }
}
```

Listing 2: Fehlerhafter Code (Beispiel 2)

```java
class Fehler2
{
    public static main(String[] args)
    {
        int i=42;
        System.out. println(i);
    }
}
```


Übungsaufgabe 1B: (ohne Abgabe)

Schauen Sie sich den folgenden Programm-Ausschnitt an. Für welche Belegungen von der Variablen `wert` wird `Ja` ausgegeben und für welche `Nein`?

```java
int wert;
// die Variablen wert wird mit einer Zahl belegt
// ...
if(wert >= 1)
{
    if(wert <= 7)
    {
        if(wert == 3)
        {
            System.out.println("Ja");
        }
        else
        {
            if(wert < 3)
            {
                System.out.println("Nein");
            }
            else
            {
                if(wert != 4)
                {
                    System.out.println("Ja");
                }
                else
                {
                    System.out.println("Nein");
                }
            }
        }
    }
}
else
{
    System.out.println("Nein");
}
```

Auch diese Aufgabe bitte nicht einsenden, aber bis zur Übung bearbeiten.