Informatik I
Prof. Dr. Carsten Damm
Dr. Henrik Brosenne

Wintersemester 2018/19
Georg-August-Universität Göttingen
Institut für Informatik

Übungsblatt 10

E-Learning

Absolvieren Sie die Tests bis Di., 15.01.2019, 23:55 Uhr.

Die Tests sind in der Stud.IP-Veranstaltung *Informatik I* unter *Lernmodule* hinterlegt.

Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.

Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

ILIAS 4–10-Minuten-Aufgaben – 12 Punkte

Absolvieren Sie die Tests *Informatik I - ILIAS 10 Teil 1, Teil 2 und Teil 3*.
(12 Punkte)

LON-CAPA – 14 Punkte

Java-Speicherbedarf, Begriffe der Objektorientierung

Absolvieren Sie im Test *Informatik I - LON-CAPA* die *Übung 10*.
(14 Punkte)

Übung

Abgabe bis Di., 15.01.2019, 18 Uhr.

Werfen Sie Ihre Lösung in den Zettelkästen Ihrer Gruppenübung. Für die Übungen im Nordbereich stehen die Zettelkästen im Sockelgeschoß (Ebene -1) **oder** auf dem Flur vor dem Seminarraum auf Ebene 0 des Instituts für Informatik.

Wenn Ihre Übung im Südbereich stattfindet, klären Sie mit Ihrem Tutor wo die Lösungen abzugeben sind.

Achten Sie darauf, dass Ihr **Name**, Ihre **Gruppe** und Ihre **Matrikelnummer** auf **jedem** Blatt stehen!

Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen.

Aufgabe 1 – 8 Punkte

Java

Betrachten Sie folgende Java-Klassendefinitionen.

```
class A {
  public void foo (int x, int y) {
    System.out.println(x+y);
  }
  public void foo (String str) {
    System.out.println(str);
  }
}

class B extends A {
  private String x = "foo";
  public void foo (int x, int y) {
    foo(this.x);
  }
  public static void main (String[] args) {
    B objB = new B();
    objB.foo(1,1);
  }
}
```

1. Geben Sie im obigen Beispiel alle Situationen an, wo Methoden überladen und Methoden oder Attribute verdeckt (verschattet) werden.
   (4 Punkte)

2. Sind die Klassen im obigen Beispiel übersetzbar? Geben Sie das Ergebnis, das die Ausführung von `B` liefert oder die Fehlerbeschreibung des Compilers an und erläutern Sie dieses.
   (4 Punkte)

Aufgabe 2 – 15 Punkte

Algorithmen

Betrachten Sie folgende Methode.

```
public static double calc(int n) {
    if (n < 0)
        throw new RuntimeException();

    double summe = 0.0;
    int produkt = 0;
    double quotient = 0.0;
    int i = 0;
    // I
    while(i < n) {
        i++;
        produkt = i * (i+1);
        quotient = 1.0/produkt;
        summe += quotient;
        // II
    }
    return summe;
}
```

Die Methode `calc(n)` berechnet `n/(n+1)` für nicht negative Eingaben `n`.

Welcher der folgenden Aussagen gilt für jede Eingabe $\texttt{n} \geq 0$,

- wenn die im Quellcode mit `I` bezeichnete Stelle
- jedes mal wenn die im Quellcode mit `II` bezeichnete Stelle

durchlaufen wird? Begründen Sie Ihre Antwort oder geben Sie ein Gegenbeispiel an.

a) `summe == calc(i)`

b) `i <= produkt`

c) `produkt < i`^{2}`+10`

d) `i > quotient`

e) `quotient <` $1/2^{\texttt{i}}$

(15 Punkte)

Aufgabe 3 – 16 Punkte

Korrektheit rekursiver Algorithmen

Betrachten Sie folgende Methode.

```
public int f(int n) {
  if(n > 100)
    return n-10;
  else
    return f(f(n+11));
}
```

Beweisen Sie mit vollständiger Induktion (siehe Skript Kapitel 7.8), dass für ganzen Zahlen $n \leq 100$ gilt $f(n) = 91$.
(16 Punkte)

Hinweis
Unterscheiden Sie im Induktionsschritt die Fälle $n > 90$ und $n \leq 90$.

Praktische Übung

Abgabe der Prüfsumme bis Di., 15.01.2019, 23:55 Uhr.
Testat von Di., 22.01.2019., 8-10 Uhr bis Fr., 25.01.2019, 18-20 Uhr.

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen, insbesondere in den Rechnerübungen am **Montag**, in denen **keine Testate** stattfinden.

Hinweise zu den praktischen Übungen, insbesondere zur **Abgabe der Prüfsumme** und zur **Durchführung der Testate**, sind in der Stud.IP-Veranstaltung *Informatik I* unter *Dateien→Übungsblätter* hinterlegt.

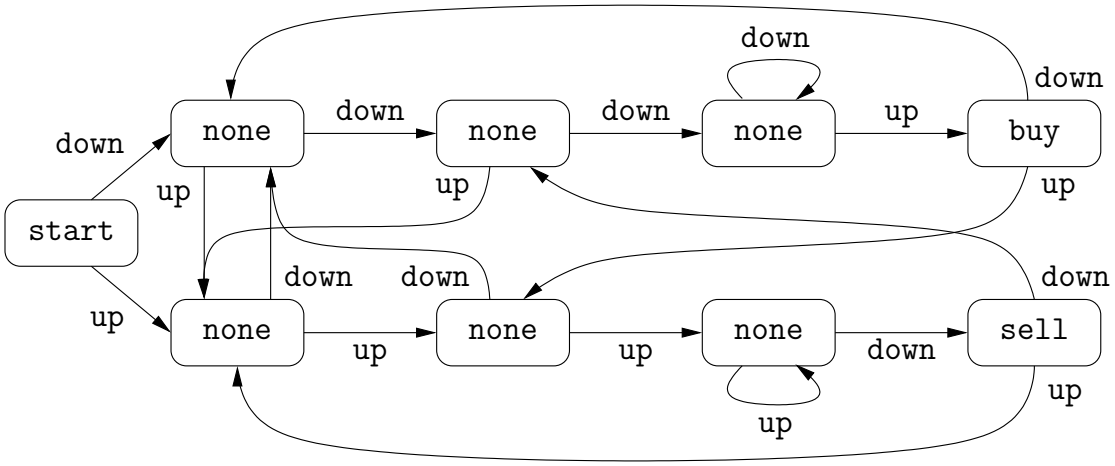
Aufgabe 1 – 35 Punkte

Automatisches Day-Trading

Für automatisches Day-Trading werden Muster in den Kursbewegungen einer Aktie gesucht. Zwei einfache Muster sind `buy` und `sell`.

- `buy`. Der Kurs ist mindestens dreimal hintereinander nicht gestiegen und anschließend einmal gestiegen.
- `sell`. Der Kurs ist mindestens dreimal hintereinander gestiegen und anschließend einmal nicht gestiegen.

Zum Erkennen dieser Muster kann man folgenden *endlichen Automaten* (engl. *finite state machine*) verwenden.



Der Automat besteht aus Knoten, die mit `start` (für den Knoten bei dem die Mustererkennung beginnt), mit dem aktuell erkannten Muster (`buy`, `sell`) oder der Information, dass noch kein Muster erkannt wurde (`none`), markiert sind. Jeder Knoten hat genau zwei ausgehende Kanten, die auf andere Knoten oder den Knoten selbst verweisen. Eine Kante für steigenden Kurs (`up`) und eine Kante für nicht steigenden Kurs (`down`).

Die Mustererkennung in einer Liste von Aktienkursen läuft wie folgt ab.

1. Setze als aktuellen Knoten den mit `start` markierten Knoten. Setze als aktuellen Kurs den ersten in der Liste aufgeführten Aktienkurs.

2. Lese den nächsten Aktienkurs aus der Liste und vergleiche diesen mit dem aktuellen Aktienkurs. Ist der Aktienkurs gestiegen wird der Knoten, auf den die mit `up` markierte Kante des aktuellen Knoten verweist zum aktuellen Knoten, sonst der Knoten, auf den die mit `down` markierte Kante verweist.

   Setze als aktuellen Kurs den aus der Liste gelesen Aktienkurs.

3. Ist der aktuelle Knoten mit `buy` oder `sell` markiert wurde das jeweilige Muster erkannt, sonst wurde (noch) kein Muster erkannt.

4. Sind noch Aktienkurs in der Liste vorhanden fahre fort mit Schritt 2., sonst ist die Mustererkennung beendet.

1. Die Applikation `Pattern` soll die Muster `buy` und `sell`, mit Hilfe des oben angegebenen endlichen Automaten, in einer Liste von Aktienkursen erkennen.

   Beachten Sie folgende Anforderungen.

   a) Verwenden Sie als Grundlagen die Klassen `Pattern` und `DayTrader`, sowie für Tests die Listen `example15.csv`, `xing20160106.csv` und `bmw20160106.csv`. Die zugehörigen Dateien finden Sie in der Stud.IP-Veranstaltung *Informatik I* unter *Dateien→Übungsblätter→Daten*.

   Die Applikation `Pattern` ist lauffähig und kann, z.B. mit `example15.csv`, getestet werden.

   ```
   > java Pattern < example15.csv
   ```

   Bemerkung. Um `Pattern` zu übersetzen wird die Klasse `StdIn` benötigt.

   b) Die Klasse `Pattern` darf, bis auf Kommentare, nicht verändert werden.

   c) Erweitern Sie `DayTrader` um symbolische Konstanten (`public static final`) für die möglichen Muster (`BUY`, `SELL`), sowie die Information, dass noch kein Muster (`NONE`) erkannt wurde.

   d) Erstellen Sie eine Klasse `Node` für die Knoten des endlichen Automaten.

   - Es gibt eine `private` Instanzvariable für den aktuellen Status (z.B. `state`), die zu den in `DayTrader` definierten symbolischen Konstanten passt.
   - Es gibt `private` Instanzvariablen für die Referenzen auf die bei steigenden und nicht steigenden Aktienkurs nachfolgenden Knoten (z.B. `up` und `down`).
   - Sehen Sie die nötigen `get`- und `set`-Methoden für die Instanzvariablen vor.

     Bemerkung. Ein kombinierte `set`-Methoden (z.B. `setUpDown`) könnte sinnvoll sein.
   - Implementieren Sie einen Konstruktor, der alle Instanzvariablen mit passenden default-Werten belegt.

   e) Erweitern Sie `DayTrader`.

- Sehen Sie einen Konstruktor vor, der aus `Node`-Objekten den oben angegeben Automaten erzeugt.
- Es gibt `private` Instanzvariable für den aktuellen Knoten (z.B. `current`) und den aktuellen Aktienkurs (z.B. `price`).
- Implementieren Sie die Methode `setPrice`, die den aktuellen Knoten und den Aktienkurses aktualisiert.
- Implementieren Sie die Methode `toString`, die `buy`/`sell` zurückliefert, wenn das entsprechende Muster gefunden wurde (siehe Status des aktuellen Knotens) und sonst eine leer Zeichenkette.

f) Versehen Sie alle Klassen mit ausführlichen Kommentaren.

g) Ergänzung Sie, wenn nötig, `DayTrader` und `Node`, sodass der Test von `Pattern` mit `example15.csv` folgende Ausgabe liefert.

```
> java Pattern < example15.csv
    1    59.75
    2    58.00
    3    57.25
    4    57.00
    5    57.50   buy
    6    61.25
    7    63.50
    8    59.00   sell
    9    58.00
   10    57.00
   11    57.25   buy
   12    57.50
   13    59.75
   14    58.00   sell
   15    58.00
```

Weiterhin soll `Pattern` auch in `xing20160106.csv` und `bmw20160106.csv` die Muster `buy` und `sell` erkennen.

(25 Punkte)

2. Schreiben Sie eine Applikation `Invest`, die, basierend auf den Mustern `buy` und `sell` in einer Liste von Aktienkursen, den An- und Verkauf von Aktien simuliert.

   Beachten Sie folgende Anforderungen.

   a) `Invest` startet mit 10000 Euro Kapital und keinen Aktien.

   b) Nutzen Sie `Pattern` als Vorlage zum Einlesen der Aktienkurse und `DayTrader` um die Muster `buy` und `sell` zu erkennen.

      Beim Muster `buy` wird das gesamte Kapital in Aktien investiert, beim Muster `sell` werden alle Aktien verkauft.

      Bemerkung. Bei den Berechnungen können die von Java bereitgestellten Gleitkommaoperationen benutzt werden, extra Überlegungen zu Rundung etc. sind nicht nötig.

   c) Geben Sie für jeden Aktienkurs den Preis der Aktie, das vorhandene Kapital, die Anzahl der gekauften Aktien, den Gesamtwert des Depots (Kapital und Aktien) und das bei diesem Kurs gefundene Muster aus.

d) Der Test von `Invest` mit `example15.csv` sollte (abgesehen vom Format) folgende Ausgabe liefern.

```
> java Invest < example15.csv
period   price        cash   shares       value trade
-----------------------------------------------------------
     1   59.75    10000.00     0.00    10000.00
     2   58.00    10000.00     0.00    10000.00
     3   57.25    10000.00     0.00    10000.00
     4   57.00    10000.00     0.00    10000.00
     5   57.50        0.00   173.91    10000.00   buy
     6   61.25        0.00   173.91    10652.17
     7   63.50        0.00   173.91    11043.48
     8   59.00    10260.87     0.00    10260.87  sell
     9   58.00    10260.87     0.00    10260.87
    10   57.00    10260.87     0.00    10260.87
    11   57.25        0.00   179.23    10260.87   buy
    12   57.50        0.00   179.23    10305.68
    13   59.75        0.00   179.23    10708.94
    14   58.00    10395.29     0.00    10395.29  sell
    15   58.00    10395.29     0.00    10395.29
```

e) Der Test von `Invest` mit den tatsächlichen Kursen der XING- (`xing20160106.csv`) und der BMW-Aktie (`bmw20160106.csv`) vom 06.01.2016 sollte für XING einen Gewinn von 359,00 Euro und für BMW einen Gewinn von 134,84 Euro ergeben (inklusive Rundungsfehler).

(10 Punkte)