Praktische Informatik in der Wirtschaft
Westfälische Wilhelms-Universität Münster
Prof. Dr. Herbert Kuchen 14. November 2016

---

5. Übung zu Informatik I im WS 16/17
Abgabe: 21. November 2016, 12 $\frac{00}{}$ Uhr.

Hinweis: Bitte verwenden Sie ab sofort und für diesen und jeden weiteren Übungszettel Kommentare in Form von **JavaDoc** für alle von Ihnen entwickelte Methoden. Eine Erklärung dazu, wie JavaDoc funktioniert, finden Sie unter:
http://openbook.rheinwerk-verlag.de/javainsel/javainsel_19_003.html
Zusätzlich wird JavaDoc diese Woche in den Tutorien kurz vorgestellt. Verwenden Sie bei Ihrer Abgabe mindestens die Annotationen `@param`, `@return` und `@author` sofern angemessen! Verwenden Sie `@author` dabei einmalig oberhalb der Klassensignatur, um dort ihre persönlichen Daten in den Dateien zu hinterlegen. Achten Sie außerdem, falls noch nicht geschehen, immer auf eine einheitliche Einrückung und Formatierung Ihres Codes!

Aufgabe 11 - Gruppenabgabe (4 + 4 =8 Punkte)
Sie haben in der Vorlesung drei verschiedene Schleifentypen kennengelernt: **For**-, **While**- und **Do-While**-Schleifen. Diese haben unterschiedliche Vor- und Nachteile, können aber generell zur Lösung derselben Probleme verwendet werden. Gegeben sind die folgenden Methoden:

a)
```java
public Integer a_doWhile(int x){
  int i = 0;
  Integer result;
  do {
    result = ++i*i;
  } while (result < x);
  return result;
}
```

b)
```java
public String b_while(int x){
  String result ="";
  while (result.length()<x) {
    result += 0.0 / 0.0;
  }
  return result+"Batman";
}
```

Verwenden Sie die java-Datei `SchleifenTypen.java`, welche Ihnen im Learnweb zur Verfügung gestellt ist und implementieren Sie die fehlenden Methoden. Die Methoden mit dem Präfix `a_` gehören dabei zum Aufgabenteil a und die Methoden mit dem Präfix `b_` gehören zum Aufgabenteil b. Die Methoden sollen dabei unter allen möglichen Eingaben von x zum selben Ergebnis führen, wie die beiden bereits implementierten Methoden! Verwenden Sie für die Implementierung aber jeweils den aus der Methodensignatur zu entnehmenden Schleifentypen! Die bereits implementierten Methoden sowie alle Klassen- und Methodensignaturen sollten **nicht** verändert werden!

Aufgabe 12 - Einzelabgabe (12 Punkte)

Gegeben sei folgende mathematische Formel zur Berechnung der Sinusfunktion:

$$\sum_{i=0}^{n} \left((-1)^i * \frac{x^{2i+1}}{(2i+1)!} \right) \text{für } n \to \infty$$

Entwickeln Sie zwei Methoden innerhalb einer Klasse MathFormel, welche beide das Ergebnis dieser Formel berechnen. Die Klasse soll folgende Signatur haben:

```
public class MathFormel {
  public double loeseRekursiv(double x) {
   ...
  }
  public double loeseSchleife(double x) {
   ...
  }
}
```

Da eine endloser Durchlauf in Java zu einem Systemabsturz führt, soll als Abbruchkriterium das n genommen werden, für das der Summand innerhalb der Klammer so klein ist, dass er zwischen $-1 * 10^{-7}$ und $1 * 10^{-7}$ liegt. In der ersten Methode soll das Problem über Rekursion gelöst werden! Es ist dafür ausdrücklich erlaubt zusätzliche Methoden anzulegen und aufzurufen. In der zweiten Methode soll das Problem mit Hilfe von Schleifen gelöst werden! Verwenden Sie in dieser Methode **keinen** Methodenaufruf! Die Math-Klasse von Java darf innerhalb der Methoden **nicht** verwendet werden! Sie können allerdings die Methode `Math.sin(double x)` zur Überprüfung Ihrer Ergebnisse verwenden. Diese berechnet ebenfalls das Ergebnis der oben dargestellten Formel. Das resultierende Ergebnis ihrer Methoden sollte mindestens auf die ersten 5 Nachkommastellen genau sein.

Hinweis: Achten Sie auf ausreichend große Datentypen, um einen Überlauf zu verhindern!

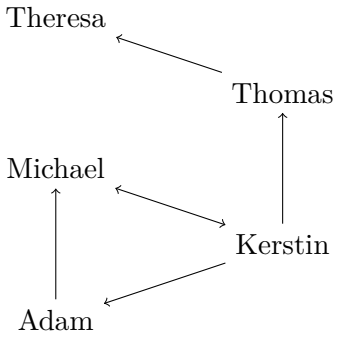
Aufgabe 13 - Einzelabgabe (10 Punkte + **4 Bonuspunkte**)

In einem sozialen Netzwerk dreht sich alles um die Interaktion mit Freunden und den Freunden der Freunde. Diese Bekanntschaften über Eck kann man beliebig weit analysieren. Laut dem Kleine-Welt-Phänomen, kennt ein Mensch jeden anderen Menschen im Durchschnitt über ca. 6 Ecken. Gegeben ist die Klasse `Person`, welche im Learnweb bereitgestellt ist. Entwickeln Sie eine Klasse `SozialesNetzwerk`, welche die Methode

```
public Person[] getFreundschaftskette(Person start, Person ende)
```

enthält. Die Methode analysiert die Bekanntschaften des sozialen Netzwerks hinsichtlich der Bekanntschaften, durch welche Person `start` mit der Person `ende` verbunden ist. Dabei wird davon ausgegangen, dass die Person `ende` nicht weiter als 6 Ecken von der Person `start` entfernt ist. Der Rückgabewert der Methode ist daher ein Personen-Array mit `length = 6`, welches die Personen in der identifizierten Freundschaftskette beinhaltet. Dabei enhält der Index 0 den Ausgangspunkt und damit einen direkten Freund der Person `start`. Die letzte Person im Array ist die Person `ende` selbst. Dies muss aber nicht zwangsläufig der Index 5 sein, sondern ist abhängig von der Länge der Freundschaftskette. Überschüssige Plätze im Array sollen den Wert null enthalten. Die Aufgabe ist von Ihnen durch **Rekursion** zu lösen! Als Beispiel sollen die Personen des folgenden Netzwerks dienen. Pfeile zwischen Personen symbolisieren eine einseitige oder beiderseitige Freundschaft.

Netzwerk



Aufruf und Ergebnis

```
result = getFreundschaftskette(adam, theresa);

result[0].getName(); // "Michael"
result[1].getName(); // "Kerstin"
result[2].getName(); // "Thomas"
result[3].getName(); // "Theresa"
result[4];           // null
result[5];           // null
```

Auf der rechten Seite ist das Ergebnis eines Aufrufs der Methode symbolisiert.

Hinweis: Es ist ausreichend, wenn Ihre Methode eine beliebige existierende Freundschaftskette findet. Falls keine Kette der Länge 6 existiert, sind alle Einträge des Arrays null.

Bonus: Versuchen Sie Ihre Lösung so zu modifizieren, dass die Methode immer die kürzeste mögliche Freundschaftskette findet.

Anmerkung: *Versehen Sie ihre Abgaben bitte mit* ***Namen****,* ***Matrikelnummern****,* ***E-Mail-Adressen*** *und* ***Studiengängen*** *der beteiligten Bearbeiter/innen und laden Sie diese in der entsprechenden Aktivität im Learnweb hoch. Aufgaben, die mit dem Hinweis* ***Gruppenabgabe*** *versehen sind, dürfen mit maximal 3 Bearbeiter/innen gelöst werden. Organisieren Sie sich dafür bitte zu Beginn der Veranstaltung in* ***Dreiergruppen****. Aufgaben mit dem Hinweis* ***Einzelabgabe*** *müssen von jedem Studenten und jeder Studentin eigenständig gelöst und abgegeben werden. Viele der Aufgaben werden über das* ***EASy*** *(E-Assessment System) System im Learnweb eingereicht und automatisch vorausgewertet. Bei ihrem ersten Übungstermin werden Sie eine kurze Einführung dazu bekommen. Um Probleme bei der Abgabe wegen Überlastung des EASy-Servers zu vermeiden, würde ich Sie bitten, eine Abgabe auf den letzten Drücker nach Möglichkeit zu vermeiden. Bei Fragen zum Übungsbetrieb wenden Sie sich bitte an ihren/ihre Tutor/in oder an Tobias Reischmann.*