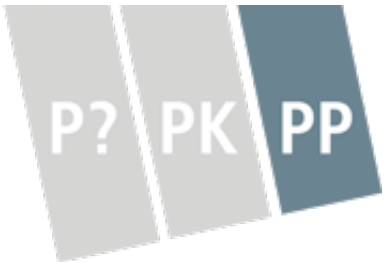

- *angabe-russiandoll*

Aufgaben

- 1. Aufgabe
  Rekursion
- 2. Aufgabe
  Rekursion

Dokumentation

- Java API
- Javainsel (8. Auflage)
- javabuch.de (Download)

Status

Noch 0 Minuten

Weg damit, das will ich gar nicht wissen!

Alles Speichern

Rekursion und Iteration

Sie finden zu jeder Aufgabe einen entsprechenden **Unterordner** im Test-Verzeichnis. Speichern Sie dort Ihre Lösungen, wenn gefordert. Alle anderen Fragen beantworten Sie **direkt in der Angabe** durch Auswählen der richtigen Antworten bei **Single- oder Multiple-Choice-Fragen** bzw. in den dafür vorgesehenen **Textfeldern** bei offenen Fragen.

Beachten Sie weiters in Ihrem eigenen Interesse die folgenden Hinweise:

- Bitte lesen Sie die Angaben der Teilaufgaben **komplett** durch, bevor Sie mit der Umsetzung beginnen. Halten Sie sich dabei genau an die Angabe. Haben Sie Verständnisprobleme, wenden Sie sich bitte an die Tutoren!
- Haben Sie **Schwierigkeiten** bei einer der Teilaufgaben, **versuchen Sie andere Aufgaben zuerst zu lösen**. Wenden Sie sich zuletzt nochmal der ungelösten Teilaufgabe zu. Sie erhalten auch Punkte für nicht vollständig gelöste Teilaufgaben

- Speichern Sie regelmäßig (Code als auch Antworten zu den Theoriefragen)

Laden Sie die Dateien für diesen Test herunter und entpacken Sie die Dateien mit "jar -xvf russiandoll.jar".

1. Aufgabe: Rekursion

(12 Punkte)

Sie haben auf dem Flohmarkt russische Puppen erstanden. Jede Puppe hat ein Gewicht, das als double-Wert gespeichert wird. Mit der Methode `void insert(RussianDoll russianDoll)` kann eine Puppe in eine andere Puppe hineingelegt werden. In jeder Puppe kann direkt maximal eine andere Puppe enthalten sein, jedoch können Puppen ineinander verschachtelt werden (z.B.: in Puppe1 ist Puppe2, in Puppe2 ist Puppe3, etc.). Eine russische Puppe kann mit der Methode `boolean isWellNested()` gefragt werden, ob die in ihr enthaltenen Puppen richtig verschachtelt sind, d.h. dass in der Puppe nur leichtere Puppen enthalten sind, wobei das rekursiv für jeder der enthaltenen Puppen gelten muss. Weiters ist es für eine russische Puppe mit der Methode `int numberOfDollsBetween(double lower, double upper)` möglich festzustellen, wieviele Puppen ein Gewicht haben, das innerhalb des mit Unter- und Obergrenze (lower und upper) angegebenen Intervalls liegt (auch hierbei wird die russische Puppe selbst und alle in ihr enthaltenen russische Puppeen betrachtet).

- Die Methode `isWellNested` kann iterativ oder rekursiv implementiert werden. Die Methode `numberOfDollsBetween` soll rekursiv implementiert werden. Vervollständigen Sie die vorgegebene Klasse im Verzeichnis "aufgabe1/impl". Nutzen Sie die mitgelieferte ausführbare Klasse um Ihre Implementierung zu testen.
  (12 Teilpunkte)

2. Aufgabe: Rekursion

(10.5 Punkte)

Wandeln Sie die Methode `PiTest.pi` in eine rekursive Methode um, ohne dabei den Methodenkopf zu ändern. Die Methode soll nach der Änderung das unverändertes Verhalten haben jedoch keine Schleife mehr enthalten. Die Verwendung von zusätzlichen Methoden (auch von Bibliotheksmethoden) ist nicht erlaubt.

`PiTest`

`public static void main (String[] args)`

Diese Methode können Sie zum Testen Ihrer Lösung verwenden. Das Programm soll vor und nach der gefragten Änderung folgende Ausgabe erzeugen:

```
Math.PI == 3.141592653589793.


WurzelTest.pi(0) == 2.0.


WurzelTest.pi(1) == 2.6666666666666665.
```

```
WurzelTest.pi(5) == 3.0021759545569062.

WurzelTest.pi(100) == 3.1337874906281584.

WurzelTest.pi(5000) == 3.141435593589907.
```

`public static double pi(int grad)`

Berechnet eine Näherung von PI. Je größer `grad` ist, desto genauer ist die Näherung. Wandeln Sie die Methode in eine rekursive Methode um, sodass für die Implementierung keine Schleife benutzt wird.

- Modifizieren Sie die im Verzeichnis "aufgabe2/impl" bereitgestellte(n) Datei(en) entsprechend der Aufgabenstellung. Sollte es im Rahmen der Aufgabenstellung erforderlich sein, können Sie auch weitere Klassen erstellen. Stellen Sie sicher, dass Ihr Programm nach den Änderungen wieder kompiliert.
  (10.5 Teilpunkte)

Modul Programmkonstruktion | Fakultät für Informatik | TU Wien