EPROG 1.Test

Dauer 70min. Keine Unterlagen erlaubt. Loginname ist q<matrikelnummer> (***also z.B.*** q0797801). ***Passwort ist Ihre Matrikelnummer.***

Speichern Sie Ihre Lösungen in den dafür vorgesehenen Unterverzeichnissen `~/test/Beispiel1/` ***und*** `~/test/Beispiel2/`.
Die Java-Dokumentation finden Sie im Verzeichnis `~/test/doc/`

Aufgabe 1:

Vervollständigen Sie die Datei
`~/test/Beispiel1/Box.java`

Die Klasse `Box` besitzt eine Methode *printBox(int s, int left, int upper, int right, int lower).* *s* ist die Seitenlänge `s` eines quadratischen Rasters in dem ein Rechteck (Box) gezeichnet werden soll. Die Kanten der Box werden mit dem Zeichen '#' gesetzt, alle übrigen Stellen des Rasters werden mit '.' gesetzt. Die restlichen Argumente der Methode bestimmen die Koordinaten der linken oberen Ecke und der rechten unteren Ecke des Rechtecks (Vorbedingungen müssen nicht überprüft werden). Der erste Punkt links oben im Raster hat die Koordinaten (0,0). Hinweis: Die einzelnen Zeichen des Rasters können Zeichen für Zeichen direkt ausgegeben werden. Ein Zwischenspeichern (z.B. in einem Array) ist nicht notwendig.

Beispiele:

Aufruf:

```
printBox (8, 1, 2, 5, 6);
```

Ausgabe:

```
........
........
.#####..
.#...#..
.#...#..
.#...#..
.#####..
........
```

Aufruf:

```
printBox (10, 4, 2, 6, 9);
```

Ausgabe:

```
..........
..........
..####....
...#..#...
...#..#...
...#..#...
...#..#...
...#..#...
...#..#...
...####...
```

Hinweis:

Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

Aufgabe 2:

Vervollständigen Sie die Datei
`~/test/Beispiel2/Polynomial.java`

Die Klasse Polynomial repräsentiert Polynome mit ganzzahligen Koeffizienten der Form

$$P(x) = \sum_{i=0}^{d} c_i x^i = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + ... + c_d x^d \qquad (1)$$

Ein Polynom soll mit einem String, der die Koeffizienten c_i enthält erzeugt werden. Schreiben Sie den entsprechenden Konstruktor. Dabei ist auf folgendes zu achten:

- Die Koeffizienten stehen nach Ihrem Index sortiert von links nach rechts durch Leerzeichen getrennt in dem String. Beispiel: Das Polynom $2+1x+3x^2$ wird durch den String "2 1 3" erzeugt.
- Falls eine Folge von Koeffizienten (oder ein einzelner Koeffizient) mit Wert 0 am Ende des Strings auftritt, werden diese Koeffizienten weggelassen, da beispielsweise der String "1 2 7 0 0 0" dem Polynom $1 + 2x + 7x^2$ entsprechen soll.
- Der Grad des Polynoms ist der Wert des höchsten Exponenten der im Polynom auftritt. Beispielsweise ist der Grad des Polynoms $1 + 2x + 7x^2$ also 2. Der Grad ist also auch gleich dem größten Index i mit $c_i \neq 0$.

Die Koeffizienten des Polynoms können wieder in einem Datenelement des Typs String gespeichert werden, wobei das Format wie oben beschrieben sein kann. Dabei sollten Koeffizienten mit Wert 0 am Ende des String weggelassen werden. Also aus "2 0 3 0 0 0" wird "2 0 3". Alternativ kann ein Array benutzt werden.

Die zu vervollständigende Klasse Polynomial soll folgende Methoden besitzen:

- *coefficient(int i)* liefert den Koeffizienten c_i mit dem Index i zurück.
- *degree()* liefert den Grad des Polynoms.
- *eval(double x)* evaluiert das Polynom mit dem Wert x, liefert also für ein bestimmtes x den double-Wert $P(x)$.
- *toString()* liefert eine Textrepräsentation des Polynoms in der Form "c_0`x^0 +` c_1`x^1 +` c_2`x^2`" zurück. Dabei sollen Terme mit Koeffizienten 0 weggelassen werden. Beispielsweise soll `"2x^0 + 3x^2"` statt `"2x^0 + 0x^1 + 3x^2"` zurückgegeben werden.

Hinweis:

Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

gegeben:

```
konstruktor (string coefficients)
scanner sc = new scanner(coefficients);

ncoeff = 0;
while(sc.hasNextInt())
        int in = nextInt();
        zeros = 0;
if (in == 0)
zeros++;
else ncoeff += zeros + 1;

sc = new scanner (coefficients)
```

und methoden... (siehe Angabe)

EPROG 1.Test

Dauer 70min. Keine Unterlagen erlaubt. Loginname ist q<matrikelnummer> (***also z.B.*** q0797801). ***Passwort ist Ihre Matrikelnummer.***

Speichern Sie Ihre Lösungen in den dafür vorgesehenen Unterverzeichnissen `~/test/Beispiel1/` ***und*** `~/test/Beispiel2/`.
Die Java-Dokumentation finden Sie im Verzeichnis `~/test/doc/`

Aufgabe 1:

Vervollständigen Sie die Datei
`~/test/Beispiel1/Prime.java`

Schreiben Sie eine Methode, die alle Primzahlen in einem bestimmten Intervall ausgibt. Eine Primzahl ist eine natürliche Zahl mit genau zwei natürlichen Zahlen als Teiler, nämlich 1 und sich selbst.

Bemerkung zu Teiler: Ein Teiler einer ganzen Zahl a ist jede ganze Zahl b, für die die Division a/b 0 Rest ergibt.
Schreiben Sie eine Methode

```
public static void printPrime(int lower, int upper),
```

die alle Primzahlen im Wertebereich `lower` bis `upper` auf dem Bildschirm ausgibt.
In der Datei `~/test/Beispiel1/Prime.java` ist der Methodenkopf bereits vorgegeben. Ergänzen Sie den Methodenrumpf.

Hinweis:

Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

1 Methode 1 % Zahl
2 Methode Zahl % Zahl

EPROG 1.Test

Dauer 70min. Keine Unterlagen erlaubt. Loginname ist q<matrikelnummer> (***also z.B.*** q0797801). ***Passwort ist Ihre Matrikelnummer.***

Speichern Sie Ihre Lösungen in den dafür vorgesehenen Unterverzeichnissen `~/test/Beispiel1/` ***und*** `~/test/Beispiel2/`.
Die Java-Dokumentation finden Sie im Verzeichnis `~/test/doc/`

Aufgabe 1:

Vervollständigen Sie die Datei
`~/test/Beispiel1/Prime.java`

Schreiben Sie eine Methode, die alle Primzahlen in einem bestimmten Intervall ausgibt. Eine Primzahl ist eine natürliche Zahl mit genau zwei natürlichen Zahlen als Teiler, nämlich 1 und sich selbst.

Bemerkung zu Teiler: Ein Teiler einer ganzen Zahl a ist jede ganze Zahl b, für die die Division a/b 0 Rest ergibt.
Schreiben Sie eine Methode

```
public static void printPrime(int lower, int upper),
```

die alle Primzahlen im Wertebereich `lower` bis `upper` auf dem Bildschirm ausgibt.
In der Datei `~/test/Beispiel1/Prime.java` ist der Methodenkopf bereits vorgegeben. Ergänzen Sie den Methodenrumpf.

Hinweis:

Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

1 Methode 1 % Zahl
1 Methode Zahl % Zahl

Aufgabe 2:

Vervollständigen Sie die Dateien

`~/test/Beispiel2/Bulb.java`

`~/test/Beispiel2/Chandelier.java`

Die Klasse `Bulb` repräsentiert eine Glühbirne einer bestimmten Glühbirnengröße (Typ der Fassung) in Form eines Strings (fitting) und einer bestimmten Leistung (power) als Integer. Definieren Sie die entsprechenden Datenelemente. Die Klasse besitzt einen Konstruktor, dem diese beiden Werte übergeben werden können. Weiters soll die Klasse die entsprechenden Auskunftsmethoden (Getter) für die 2 Datenelemente besitzen (siehe java-Datei).

Die Klasse `Chandelier` repräsentiert einen Kronleuchter mit 3 Fassungen ("Glühbirnenplätzen" → 3 Datenelemente vom Typ `Bulb`) gleicher Größe (1 weiteres Datenelement vom Typ `String` zum Speichern der Bezeichnung (fitting) des Gühbirnentyps (=Typ der Fassung).

Die Methode **`boolean mount(Bulb p, int pos)`** übernimmt ein Glühbirnen-Objekt und fügt es dem Kronleuchter hinzu (schaubt es in die Fassung mit der der Position pos). Es gibt nur 3 Positionen. (Sie benötigen hier also **keine** Arrays). Wenn eine Glühbirne eingeschraubt werden kann, liefert die Methode **`true`** zurück, sonst **`false`**. Eine Glühbirne kann nur eingeschraubt werden wenn die Glühbirnengröße (=Typ der Fassung) zur Fassung passt. Eine Glühbirne wird von Position `pos` entfernt, indem **`null`** übergeben wird.

Die Klasse `Chandelier` soll auch eine Methode `getPower()` besitzen, die die Gesamtleistung des Kronleuchters, also die Summe der Leistung aller eingeschraubten Glühbirnen, liefert.

Definieren Sie für die Klasse `Chandelier` die notwendigen Datenelemente und Konstruktoren.

Hinweis:

Die vorgefertigte Datei `Main.java` enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Ihr Inhalt wird nicht beurteilt.

Bulb p = boolean mount (Bulb p, 1)

Bulb 1 = Bulb x

return true;

"Gizem" "Gizem: 3"
3

JJJJuu555566 JTOP

kaç tane 3
kaç tane 4
kaç tane 5

Aufgabe 2:

Vervollständigen Sie die Dateien
~/test/Beispiel2/Counter.java
~/test/Beispiel2/Timer.java

Die Klasse `Counter` repräsentiert einen Zähler, der eine Bezeichnung (String) besitzt, die angibt welche Einheiten gezählt werden (z.B. "Stunden", "Minuten"). Der Zählerstand ist eine Ganzzahl. Definieren Sie die zwei entsprechenden Datenelemente (Bezeichnung der Einheiten und Zählerstand), die `private` sein sollen. Ein Zähler beginnt immer bei 0 zu zählen. Mit der Methode `boolean increment()` soll der Zählerstand um 1 erhöht werden können. Im Normalfall liefert die Methode "false" zurück. Jeder Zähler soll ein Limit besitzen, bis zu dem er Zählen kann. Sobald diese Obergrenze erreicht wird, wird beim nächsten Aufruf von `increment()` der Zählerstand auf 0 zurückgesetzt und nur in diesem Fall der Wert "true" zurückgeliefert.
Die Klasse `Counter` soll über einen Konstruktor verfügen, der den Zählerstand mit 0 initialisiert und Bezeichnung (String) und Limit als Argumente übernimmt und speichert.
Die Klasse soll *keine* weiteren Methoden zum Setzen des Zählerstands zur Verfügung stellen. Ein Zählerstand kann also von einer anderen Klasse nur durch Aufruf von `increment()` gesetzt werden. Durch wiederholte Aufrufe kann aber jeder Zählerstand erreicht werden.
Weiters soll es eine Methode `int getValue()` zum Lesen des Zählerstands geben, die den `int`-Wert liefert. Zusätzlich soll eine Methode `public String toString()` zur Verfügung stehen, welche eine lesbare Repräsentation des Zählers in der Form "Bezeichnung: Zählerstand" zurückliefert.
Die Klasse `Timer` repräsentiert einen Timer ähnlich einer Stoppuhr, die Zeitspannen messen kann. Eine Zeitspanne wird in der Form D:H:M:S repräsentiert, wobei D für die Anzahl der Tage steht, H für die Anzahl der Stunden, M für die Anzahl der Minuten und S für die Anzahl der Sekunden. Zulässige Wertebereiche sind:
für Tage 0,1,...,99, für Stunden 0,1..,23, für Minuten und Sekunden 0,1,...59. Diese vier Einheiten sollen durch 4 Zähler (=Datenelemente vom Typ Counter) repräsentiert werden, wobei die Zählerstandobergrenze durch die Obergrenze der zulässigen Wertebereiche bestimmt werden (also z.B. 23 für Stunden).
Die Klasse soll eine Methode `setTime(int day, int hour, int minute, int second)` besitzen. Die Methode setzt unter Benutzung der Methode `increment()` der Klasse `Counter` die entsprechenden Werte für die Datenelemente. Sie müssen hier den Zähler bis zum erwünschten Wert inkrementieren. Die Zusicherungen für übergebene Werte müssen nicht überprüft werden.
Weiters soll diese Klasse auch einen eigenen Konstruktor besitzen, der die vier `Counter`-Objekte mit entsprechender Obergrenze erzeugt.
Die Methode `tick()` entspricht einem Takt des Timers, wodurch dieser um eine Sekunde hinaufzählt. Sobald ein Sekundenüberlauf passiert (59→0), wird der Minutenzähler gleichzeitig um eins erhöht. Wenn dieser überläuft (59→0), wird der Stundenzähler erhöht, usw. `tick()` liefert nur "true" zurück wenn ein Gesamtüberlauf (Überlauf der Tage) aufgetreten ist.
Eine Methode `public String toString()` soll zur Verfügung stehen, welche eine lesbare Repräsentation der Timers in der Form "Day: D, Hour: H, Minute: M, Second: S" zurückliefert. Führende 0 sollen nicht ausgegeben werden.

3, 22, 53, 43

"Day: 3"
3 "Hour: 22"
"Minute: 53"
"Second: 43"

20 23 59 58
20 23 59 59
21 0 0 0

day 3
value = this.day.getValue()
this.day.increment()
value = this.day.getValue

value = 0
value = 1

RÜCKSEITE!

zurückliefer! Führende 0

sollen nicht ausgegeben werden

Aufgabe 2: Klasse definieren (`Circle` und `Point`)

Vervollständigen Sie die Dateien `~/test/Beispiel2/Circle.java`
`~/test/Beispiel2/Point.java`

Die Klasse `Point` repräsentiert einen 2D-Punkt im karthesischen Koordinatensystem und soll eine Methode

```
public double distanceTo (Point p)
```

zur Verfügung stellen mit der die Distanz zu einem dem Parameter `p` übergebenen Punkt berechnet wird. Hinweis: Die Distanz d zwischen zwei Punkten $p_1=(x_1,y_1)$ und $p_2=(x_2,y_2)$ ist gegeben durch

$$d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

Die Klasse `Circle` repräsentiert einen Kreis, der durch Mittelpunkt (Typ `Point`) und Radius (Typ `double`) definiert wird. Zu implementieren sind zwei Methoden der Klasse `Circle`. Die Methode

```
public boolean onCircle(Point p),
```

die true liefert wenn der angegebene Punkt mit einer Genauigkeit von +/- EPSILON (vordefiniertes Datenelement) auf dem Kreis liegt, sonst false und die Methode

```
public double area(),
```

die den Flächeninhalt des Kreises liefert.

Definieren Sie auch alle notwendigen Datenelemente und Konstruktoren.

Hinweis:
Die Datei `Main.java` können Sie zum Testen Ihrer Klassen benutzen

```
public static double fakultaet (int i)
int sonc = 1;
if (i == 0)
sonc = 1;
else
for (int temp=1; temp
```