Klausur SS 2020

Bevor Sie mit der Bearbeitung der Klausur beginnen, lesen Sie bitte folgende Hinweise:

- Prüfen Sie die **Vollständigkeit** Ihres Exemplars. Jede Klausur umfasst
  - das Deckblatt
  - diese Hinweisseite
  - 5 Aufgaben mit Lösungsblättern

  Bei Unvollständigkeit Ihres Exemplars wenden Sie sich bitte an die Aufsicht.
- Tragen Sie auf **jedem Lösungsblatt** Ihre **Matrikelnummer** ein und **unterschreiben** Sie auf dem Deckblatt! Blätter ohne diese Angabe werden nicht bewertet.
- Hinter den Aufgaben ist jeweils hinreichend Platz für Ihre Lösungen.
  Verwenden Sie kein eigenes Papier!
  Sollten Sie weiteres Papier benötigen, so wenden Sie sich bitte an die Aufsicht. Die Zuordnung von Lösung zu Aufgabe muss auf jeden Fall deutlich erkennbar sein.
- Sie haben die Klausur bestanden, wenn Sie mindestens **30 Punkte** erreicht haben.

Ergebnis (bitte nichts eintragen):

Aufgabe	1	2	3	4	5	$\sum$
Punkte	15	15	15	15	10	70
erreicht						

Aufgabe 1 (15 P)
Implementieren Sie eine Klasse `Artikel` mit folgenden Eigenschaften: Artikel haben

- eine Bezeichung `bez` vom Typ `String`,
- eine Artikelnummer `nr` vom Typ `int` und
- einen (Netto-)Preis `nettoPreis` vom Typ `double`.
- Die Bezeichung und der Netto-Preis werden bei Erzeugung eines Objektes als Parameter übergeben.
- Die Artikelnummer wird bei Erzeugung eines Objektes automatisch fortlaufend vergeben.
- Außerdem gibt es einen Mehrwertsteuersatz (`mwst` vom Typ `double`), der zu Beginn (für alle Artikel) bei 19 Prozent, also 0.19 liegt.
- Der Mehrwertsteuersatz kann durch eine Methode `setMwst()` geändert werden. Er ändert sich dann für alle Artikel auf den übergebenen neuen Wert.
- Für Artikel gibt es eine Methode `bruttoPreis()`, die aus dem Netto-Preis und dem aktuell gültigen Mehrwertsteuersatz den Brutto-Preis berechnet.
  (Die Berechnungsformel dafür lautet $\text{bruttoPreis} = \text{nettoPreis} \cdot (1 + \text{mwst})$)
- Die `toString`-Methode soll Artikelnummer und Bezeichnung liefern.
- Nur der Konstruktor und die Methoden sollen von außerhalb der Klasse zugreifbar sein.

Definieren Sie eine Klasse mit geeigneten Klassen- und Instanz-Attributen, Konstruktor und den geforderten Methoden. Außer den ausdrücklich genannten Methoden brauchen Sie keine weiteren (zB get- oder set-) Methoden zu definieren.

Aufgabe 2 (5+10 P)
Betrachten Sie die Klasse `A` und die main-Methode wie unten angeben.

a) Zeichnen Sie die Speicherbild-Situation nach der zweiten der vier Anweisungen der main-Methode, also an der mit „(*)“ gekennzeichneten Stelle.

```
public class A {

        private static int a;
        private double x;

        public A(int n, double y) {
                a = n;
                x = y;
        }

        public void incr() {
                a = a+1;
                x = x+1.0;
        }

        public static void main(String[] args) {
                A p = new A(4, 2.0);
                A q = new A(1, 5.5);
                // (*)
                p.incr();
                q.incr();
        }
}
```

b) Vervollständigen Sie die unten angedeutete Trace-Tabelle um entsprechende Spaltenüberschriften. Tragen Sie die Werte aller vorkommenden Variablen und die Änderung dieser Werte nach den jeweiligen Anweisungen ein.

Anweisung	*(hier bitte Spalten für die vorkommenden Variablen einfügen)*
`A p = new A(4, 2.0);`	
`A q = new A(1, 5.5);`	
`p.incr();`	
`q-incr();`	

Aufgabe 3 (5+2+4+4 P)

a) Schreiben Sie eine (statische) Methode `teilSumme()`, die ein `double`-Array und zwei Positionsnummern i und j (vom Typ `int`) als Eingabeparameter erhält.

   Die Methode soll die Teilsumme der Array-Einträge von Position i (einschließlich) bis Position j (ausschließlich) berechnen und zurückgeben.

b) Definieren Sie eine Fehlerklasse `FalscheParameterException`, die von `Exception` abgeleitet wird und Fehlermeldungen mit dem Fehlertext „unzulaessige Parameter“ erzeugt.

c) Falls für die Methode aus a) die Eingabeparameter i oder j unzulässige Werte darstellen, soll die Methode eine `FalscheParameterException` auslösen.
   Unzulässige Werte sind solche, die Array-Grenzen überschreiten oder für die $i \geq j$ gilt.

d) Schreiben Sie eine (statische) Methode `testTeilSumme`, in der die Methode `teilSumme` für das Array $\{1.0, 2.0, 3.0\}$ und beliebige Parameter i, j aufgerufen wird. Gegebenenfalls auftretende Exceptions sollen durch einen Exception-Handler abgefangen werden.

(Sie *können* - müssen aber nicht - für die Aufgabenteile a) und c) eine gemeinsame Lösung abgeben.)

Aufgabe 4 (3 + 12 P)
Gegeben sei das folgende Interface

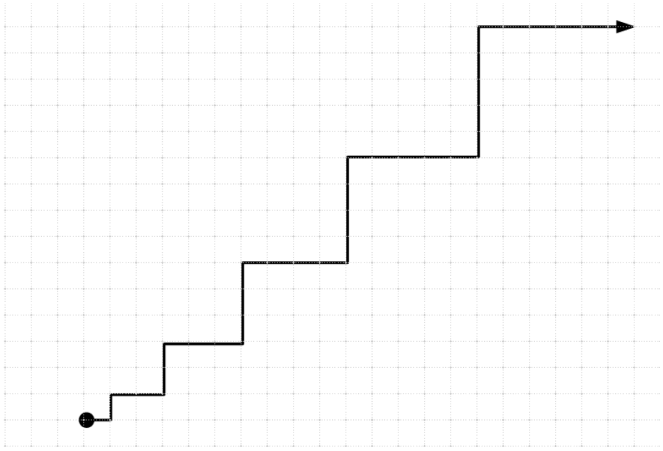
```
public interface Turtle {

        void step(); // ein Schritt in Blickrichung

        void step(int n); // n Schritte in Blickrichtung

        void turnR(); // Vierteldrehung gegen den Uhrzeiger

        void turnL(); // Vierteldrehung mit dem Uhrzeiger
}
```

Nehmen Sie an, die Klasse `Roboter` implementiere dieses Interface.
Die Klasse verfüge über einen parameterlosen Konstruktor, der einen Roboter mit Startposition $(0, 0)$ und Blickrichtung „rechts“ erzeugt.

a) Definieren Sie ein Interface `TreppenTurtle`, das das Interface `Turtle` um eine Methode `void treppe(int n)` erweitert.

b) Die Methode `treppe(int n)` ermöglicht den Objekten, treppenförmige Wege zu laufen (siehe unten). Der Parameter n gibt die Anzahl der Stufen an.
Implementieren Sie eine Klasse `TreppenRoboter`, die von `Roboter` abgeleitet ist und zusätzlich das Interface `TreppenTurtle` implementiert.
(Beachten Sie bei der Implementierung, dass die Treppe eine rechts-Bewegung mehr erfordert als nach-oben-Bewegungen.)

Beispiel:
Für den Wert $n = 5$ sollte die Treppe die unten gezeigte Gestalt haben.



Turtleweg Seite 1

Implementieren Sie „gegen das Interface“! Sie brauchen an keiner Stelle weitere Implementierungsdetails der Klasse `Roboter` zu kennen oder nutzen.

Aufgabe 5 (4 + 6 P)
Durch die Corona-Epidemie haben wir alle viel über den epidemiologischen Verlauf von Infektionskrankheiten gelernt.

Es sei

- p die Bevölkerungszahl eines Landes
- s die Anzahl der für eine Ansteckung empfänglichen Personen (diese Zahl sinkt im Laufe der Zeit, da einmal infizierte Personen nicht noch einmal angesteckt werden können)
- r die Basisreproduktionszahl, dh die Anzahl an Personen, die durchschnittlich von einer infizierten Person angesteckt werden, wenn (noch) niemand immun ist
- $r_e = r \cdot \frac{s}{p}$ die effektive Reproduktionszahl
- n_t die Zahl der Neuinfektionen am Tag t nach Ausbruch
- z_t die Zahl der insgesamt infizierten Personen nach t Tagen

Ein (sehr stark vereinfachtes) Modell für die Berechnung der Zahl der Infizierten führt auf die 2-Term-Rekursion:

$$z_t = z_{t-1} + n_{t-1}$$
$$n_t = n_{t-1} \cdot r_e = n_{t-1} \cdot r \cdot \frac{s}{p} = n_{t-1} \cdot r \cdot \frac{p - z_{t-1}}{p}$$

Diese beiden Formeln lassen sich auch so ausdrücken:

$$z(t) = z(t-1) + n(t-1)$$
$$n(t) = n(t-1) \cdot r \cdot \frac{p - z(t-1)}{p}$$

Die Startwerte für n und z sind $n_0 = n(0) = 1$ und $z_0 = z(0) = 0$.

a) Berechnen Sie die Werte $n_t = n(t)$ und $z_t = z(t)$ für $t = 1$ und $t = 2$ „von Hand“ .
(Nehmen Sie $r = 2$ und $p = 100$ an.)

b) Implementieren Sie statische Methode(n), die die Anzahl der infizierten Personen nach t Tagen berechnet. Verwenden Sie die Werte $p = 80000000$ und $r = 2.0$, während t als Parameter übergeben wird.

Bemerkungen:

1. Sie können, aber Sie müssen die Rekursionsformel nicht *rekursiv* implementieren!
   Wie bei den Fibonacci-Zahlen gibt es auch hier eine sehr gute *iterative* Lösung.

2. Der Parameter t ist natürlich vom Typ `int` (eine *Anzahl* von Tagen);
   alle anderen Variablen sollten Sie als vom Typ `double` deklarieren (weil es sich hier um *durchschnittliche* Anzahlen handelt, in denen Brüche auftreten).