Universität Stuttgart

FIUS

PSE Vorkurs Tag 4

Heute Abend LAN-Party und Spieleabend!

Feedback

Bitte füllt den Feedbackbogen aus, damit wir den Vorkurs für die nächsten Jahre verbessern können. Der Bogen ist anonym und dauert nur 2-3 Minuten.
Ihr findet ihn unter: `https://forms.gle/wVJ9VRoirpk31vVu9`

Musterlösungen

Tag 3 Lösungen

Präsentations-Cheatsheet

Klassensyntax:

```java
public class Student {
    String name;
    int matrikelnummer;

    void lernen() {
        System.out.println(name + " lernt fleißig!");
    }
}
```

Konstruktorsyntax:
Konstruktoren werden immer bei der Erstellung eines Objekts (also bei der Instanziierung) ausgeführt

```java
public class Student {
    private final String name;
    public int matrikelnummer;

    public Student(String name, int matrikelnummer) {
        this.name = name;
        this.matrikelnummer = matrikelnummer;
    }
}
```

Instanzen von Klassen:

```java
public Student(String name, int matrikelnummer) {
    this.name = name;
    this.matrikelnummer = matrikelnummer;
}

Student paul = new Student("Paul", 12345);
paul.lernen(); // → Paul lernt fleißig!

Student anna = new Student("Anna", 67890);

System.out.println(anna.matrikelnummer); // → 67890
anna.lernen(); // → Anna lernt fleißig!
```

Aufgabe 1: Klassen

Heute werden wir eine Klassenstruktur umsetzen, die ein Modell für eine Grillung sein soll.

1. In Java haben Klassen im Allgemeinen jeweils eine eigene Datei. Erstelle jeweils eine neue `Main`-Klasse, sowie eine `Grill`-Klasse. In der `Main` kann jetzt auf die Grillklasse und ihre Funktionen zugegriffen werden.

2. Gib deiner Grillklasse einen Konstruktor, der ausgibt, dass ein neuer Grill erstellt wird. Teste den Konstruktor der Grillklasse in dem du ein Grillobjekt in der Main-*Funktion* der Main-*Klasse* erstellst und in einer Variable mit dem Typ `Grill` speicherst.

3. Füge dem Grill Attribute hinzu. Jeder Grill sollte einen Grillmeister (Name als `String`), eine maximale Würstchenanzahl (`int`), eine maximale Hitze (`int`), sowie eine momentane Hitze (`int`) haben. Erweitere nun den Konstruktor so, dass er den Namen des Grillmeisters, die maximale Würstchenanzahl und die maximale Hitze als Parameter entgegennimmt und anschließend die Attribute entsprechend setzt. Momentane Hitze soll bei der Erstellung des Grills noch 0 sein.

4. Gib dem Grill eine oder mehrere Funktionen zur Kontrolle der Hitze. Gestalte sie so wie du es für sinnvoll hältst. Beachte, dass der Grill eine maximale Hitze hat und Hitze nicht negativ sein kann.

5. Füge dem Grill ein Array der Größe `maximaleWuerstchen` hinzu, dass die Würstchen symbolisieren soll. An Stellen an denen kein Würstchen liegt, soll -1 im Array stehen. Bei der Initialisierung sollen keine Würstchen auf dem Grill liegen.

6. Erstelle eine Funktion, um Würstchen auf den Grill zu legen. Ein Würstchen wird mit Garstufe 0 auf den Grill gelegt.

7. Füge dem Grill eine `Grillen` Funktion hinzu, die die momentane Hitze auf die Garstufe jedes Würstchen addiert.

8. Erstelle eine `runternehmen`-Funktion, die ein Würstchen vom Grill nimmt und ausgibt, ob es noch roh, gut durch oder verbrannt ist. Ein Würstchen ist durch, wenn seine Garstufe zwischen 10 und 15 ist. Unter 10 ist ein Würstchen noch roh, über 15 ist es verbrannt.

9. Nutze die main um einen beispielhaften Grillablauf einer mehr oder weniger erfolgreichen Grillerei darzustellen.

Aufgabe 2: Klassen in Klassen

Eine Sache, die dir vielleicht schon aufgefallen ist, ist dass wir momentan leider nur Würstchen Grillen können und auch bei denen nur genau eine Sorte. Dafür wird jetzt eine neue Grillgut-Klasse erstellt, die viel flexibler sein soll und unserer (leider digitalen) Grillung viel mehr ermöglichen wird.

1. Erstelle eine neue Klasse (wieder mit neuer Datei) namens `Grillgut`. Sie soll einen Grillguttyp (`String`, z.B. Currywurst), eine minimale Garstufe (`int`), eine maximale Garstufe (`int`), eine momentane Garstufe (`int`), eine Hitzetoleranz (`int`) und einen verbrannt (`Boolean`) Status haben. `Hitzetoleranz` stellt dar, dass ein Grillgut es nicht aushält über einer bestimmten Temperatur gegrillt zu werden (man kann Würstchen nicht statt fünf Minuten auf 200 Grad eine Minute auf 1000 Grad grillen).

2. Implementiere einen Konstruktor, der manche der obigen Attribute als Parameter entgegennimmt (die bei denen es dir sinnvoll erscheint), aber alle Attribute initialisiert, d.h. nach der Ausführung des Konstruktors sollten alle Attribute einen Wert haben.

3. Erstelle eine `grillen` Funktion in der `Grillgut`-Klasse, die eine Hitze als Parameter entgegennimmt und das Grillgut entsprechend grillt. Beachte, dass ein Grillgut sofort verbrennt, wenn es mit einer Hitze gegrillt wird, die höher ist als seine Hitzetoleranz.

4. Erstelle zudem eine `vomGrillNehmen` Funktion die ausgibt was gerade vom Grill genommen wurde und auch ausgibt wie durch es ist.

5. Gehe zurück zu deiner Grillklasse und ersetze das `int[] aufDemGrill` Array mit einem `Grillgut[] aufDemGrill`. Mache als Folge dessen sinnvolle Anpassungen in der Grillklasse, die die Funktionen aus Grillgut verwenden.

6. Nutze die main-Funktion um einen neuen beispielhaften Grillablauf einer mehr oder weniger erfolgreichen Grillung darzustellen, aber jetzt mit vielen unterschiedlichen Grillgütern.

HIGHPERFORMER: Optimale Grillerei

Stellt euch vor, es war ein langer Tag im PSE-Vorkurs und ihr habt euch eine Grillung verdient, doch der Grill ist klein und es gibt viele unterschiedliche Grillgüter, sowie viele hungrige Studis. Jetzt muss eine Methode her um möglichst schnell alles gegrillt zu bekommen und das natürlich ohne, dass ein einziges Stück verbrennt oder roh bleiben muss.

1. Füge dem Grill einen Zähler hinzu, der bei jedem Aufruf der `grillen`-Funktion hochgezählt wird und ausgegeben wird. So kann die Dauer einer Grillerei gemessen werden.
2. Programmiere in der Grillklasse eine Funktion die eine zufällige Anzahl (zwischen 50 und 500) an Grillgütern (genannt `Mysterywurst`) mit zufälligen minimalen (zwischen 5 und 20) und maximalen Garstufen (zwischen 10 und 40) initialisiert. Achte jedoch drauf, dass die minimale Garstufe nie größer ist als die maximale. Die Hitzetoleranz soll genauso zufällig gesetzt werden (zwischen 1 und 10). Diese sollen alle in einem `Grillgut[]` Array gesammelt werden und von der Funktion zurückgegeben werden.
3. Erstelle wieder in der Grillklasse eine Funktion, die eins von den in der Highperformer-Aufgabe 1. generierten Arrays von `Grillgut`-Objekten entgegennimmt und in möglichst wenigen aufrufen der `grillen`-Funktion alle grillt. Nutze aus, dass nur die `grillen`-Funktion in der Zeit gezählt wird und "kostenlos"die Hitze umgestellt werden kann, sowie auf den Grill gelegt und wieder heruntergenommen werden kann. Es darf natürlich am Ende keine verbrannten oder rohen Würstchen geben. Wir haben einen beispielhaften (nicht-optimalen) Algorithmus Codeanhang gebastelt. Vielleicht kriegt ihr es hin, dass eurer im Durchschnitt schneller wird.

Feedback

Bitte füllt den Feedbackbogen aus, damit wir den Vorkurs für die nächsten Jahre verbessern können. Der Bogen ist anonym und dauert nur 2-3 Minuten.
Ihr findet ihn unter: `https://forms.gle/wVJ9VRoirpk31vVu9`

Codeanhang

```
public void optimaleGrillerei(Grillgut[] zuGrillen) {
    int anzahlDurch = 0;
    int draufGelegt = 0;
    while (anzahlDurch < zuGrillen.length) {
        int momentaneMaxHitze = maxHitze;
        for (int i = 0; i < aufDemGrill.length; i++) {
            if (aufDemGrill[i] == null && draufGelegt <zuGrillen.length) {
                aufDemGrill[i] = zuGrillen[draufGelegt];
                draufGelegt++;
            }
            if (aufDemGrill[i] != null) {
                momentaneMaxHitze = Math.mi(momentaneMaxHitze,
                ↪  aufDemGrill[i]hitzetoleranz);
                momentaneMaxHitze = Math.mi(momentaneMaxHitze,
                ↪  aufDemGrill[i]maxDurchheit - aufDemGrill[i].durchheit);
            }
        }
        setzeHitze(momentaneMaxHitze);
        grillen();
        for (int i = 0; i < aufDemGrill.length; i++) {
            if (aufDemGrill[i] != null && aufDemGrill[i]durchheit <=
            ↪  aufDemGrill[i].maxDurchheit &&aufDemGrill[i].durchheit >=
            ↪  aufDemGrill[i]minDurchheit && !aufDemGrill[i].verbrannt) {
                anzahlDurch++;
                runternehmen(i);
            }
        }
    }
    System.out.println("Diese Grillung hat für " +zuGrillen.length + " Grillgüter
    ↪  " + grillZaehler + "Grillzyklen gebraucht, ich hoff das war nicht
    ↪  zulang");
}
```