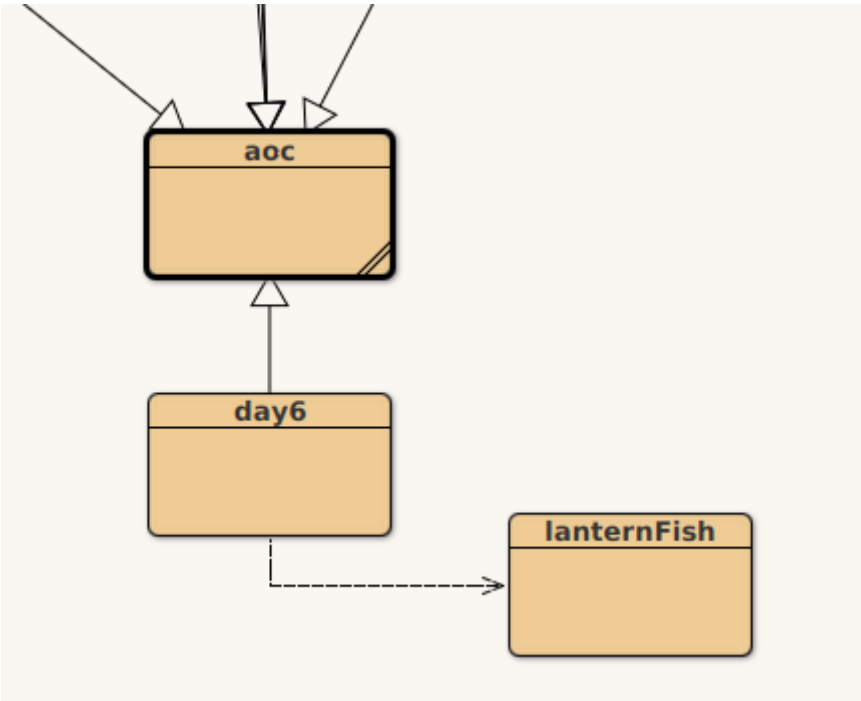
Tag 6: Laternenfische, viele Laternenfische

Aufgabenteil 1

Für den Teil 1 kann man es mit einer Modellierung wie der folgenden versuchen:



Einige Tipps für diesen Ansatz:

- Die `lanternFish`-Objekte können innerhalb von `day6` in einer ArrayList verwaltet werden: `ArrayList<lanternFish> fishList = new ArrayList<>();`
- Die `lanternFish`-Objekte können eine Methode wie `makeTimeStep` haben, die einen Zeitschritt auf dem Fish implementiert. Hier fließen die Rahmenbedingungen der Aufgabenstellung ein. Neue Fische könnten an eine ArrayList mit neuen Fischen angehängt werden, die nachdem alle Objekte der `fishList` bearbeitet wurden an die `fishList` angehängt werden.

Damit kann man Teil 1 lösen, der Ansatz fällt einem aber in Teil 2 laufzeitmäßig auf die Füße, da die Zahl der `lanternFish`-Objekte in der Array List sehr schnell wächst.

Codegerüst für diesen Ansatz

Die `lanternFish`-Klasse und Methoden wie `parseInputToFishList()` um den Input zu verarbeiten müssen passend implementiert werden.

```
public int partOne(int daysToSimulate) {
        int numFish = 0;
        parseInputToFishList();
        ArrayList<lanternFish> tempfishList = new ArrayList<>(fishList);
```

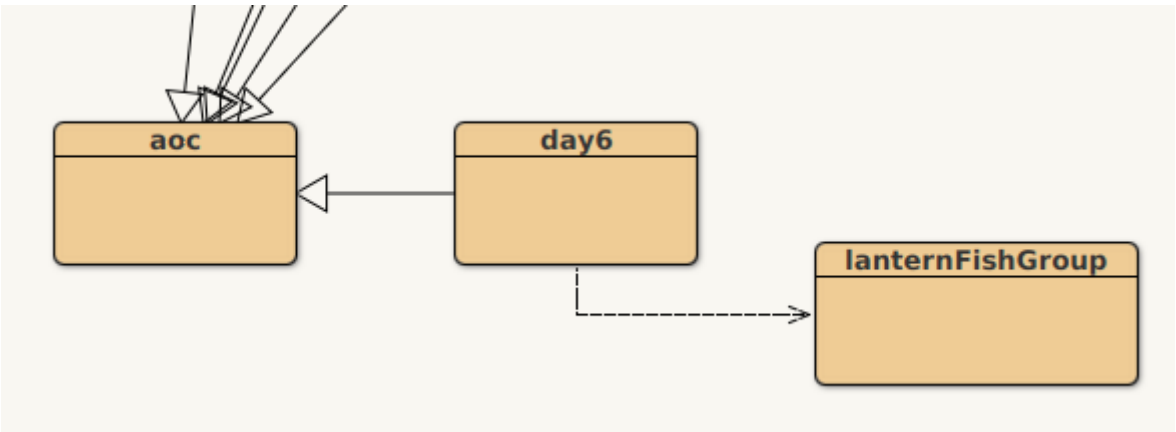
```
        for(int day=0;day<daysToSimulate; day++) {
            int dayNum=day+1;
            System.out.print("Day " + dayNum + ":");

            Iterator<lanternFish> fishIterator = fishList.iterator();
            while(fishIterator.hasNext()) {
                lanternFish f = fishIterator.next();
                // Laesst den Fisch altern und fügt ein mögliches Kind
                // an die tempfishList an
                f.makeTimeStep(tempfishList);
            }

            fishList.clear();
            fishList = (ArrayList<lanternFish>) tempfishList.clone();
            //System.out.print(fishList);
            numFish = fishList.size();
            System.out.println(" There are " + numFish + " fish in the
sea");
        }
        return numFish;
    }
```

Aufgabenteil 2

Wenn man die neuen Fische einzeln in die Liste der Fische einfügt, wir diese sehr schnell sehr lang. Bei genauerem nachdenken fällt auf, dass das auch gar nicht nötig ist, denn bei allen in einem Zeitschritt angefügten Fische gehen die inneren Uhren synchron - sie beginnen alle bei 8 und werden dann runtergezählt. Man kann das ganze also anstatt mit einzelnen Fischen mit Fischgruppen modellieren:



- Die `lanternFishGroup` hat ein weiteres Attribut `numFish`, das angibt, wie groß die Gruppe ist.
- Die Zeitschritt Methode der `lanternFishGroup` gibt einen ganzzahligen Wert zurück, wieviele Fische im Fall, dass der Gruppentimer auf 0 steht neu erzeugt werden.
- wenn man alle Gruppen in einem Zeitschritt durchläuft, kann man ermittelln, wieviele neue

Fische in diesem Zeitschritt erzeugt werden. Das kann man sich in der Gesamtsumme der Fische merken - und außerdem eine neue Gruppe der entsprechenden Größe anfügen. In `day6` gibt es nun also eine ArrayList für die Fischgruppen: `ArrayList<lanternFishGroup> fishGroupList = new ArrayList<>();'`
- Außerdem wird man feststellen, dass beim realen Input `int` nicht ausreicht, da der Wertebereich hier überläuft.

From:
https://info-bw.de/ -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:aoc:aoc2021:day6:start?rev=1638801955

Last update: **06.12.2021 14:45**