Weihnachtskopfnuss

Bitte erstellen Sie für dieses Blatt ein IntelliJ-Modul „**Hypercube**“.

Hypercube

Implementieren Sie die Klasse **Hypercube**. Sie realisiert ein multidimensionales Array, welches Zahlen vom Typ **Integer** speichern kann.

- Package: **hypercube**
- Klassen: **Hypercube**, **TestTheCube**

Der Hypercube hat eine festgelegte Anzahl von Dimensionen und eine festgelegte Kantenlänge. Dimension und Kantenlänge werden im Konstruktor als Parameter übergeben.

Beispiel: Ein dreidimensionaler Hypercube mit einer Kantenlänge von 7

- Wird erzeugt mit: `Hypercube hypercube3D = new Hypercube(3, 7);`
- Ist ein Würfel mit dem Format 7 * 7 * 7.
- Besitzt die Koordinaten [0, 0, 0], [0, 0, 1], [0, 0, 2], …, [6, 6, 4], [6, 6, 5] [6, 6, 6]

Der **Hypercube** hat folgende Methoden

- Eine Methode um eine Zahl an einer bestimmten Koordinate einzutragen
  `public void set(int[] coordinate, Integer value) {…}`
- Eine Methode um eine Zahl an einer bestimmten Koordinate abzufragen
  `public Integer get(int[] coordinate) {…}`
- Eine **`toString()`**-Methode um sich selbst darzustellen

Hinweise

- Es gibt mehrere Möglichkeiten, den Hypercube zu implementieren.
  - Eine einfache Möglichkeit besteht darin, ein **`Object[]`** als Datentyp zu verwenden und es rekursiv mit weiteren **`Object[]`** zu füllen. Das funktioniert deshalb, weil auch ein Array in Java vom Typ **`Object`** ist. Bei dieser Lösung müssen mehrere Casts angewendet werden.
  - Sie dürfen sich aber auch eine andere Lösung ausdenken.
- Damit die **`toString()`**-Methode funktioniert, sollte man folgende zwei Hilfsmethoden implementieren
  - Eine Methode, die einen einzelnen Eintrag (eine Zeile) ausgeben kann
    `private String entryToString(int[] coordinate) {…}`
  - Eine Methode, die die nächsthöhere Koordinate erzeugen kann also z.B. [2, 2, 0] → [2, 2, 1]
    `private int[] retrieveNextCoordinateIfAvailable(int[] coordinate) {…}`

Die Klasse **TestTheCube gibt die drei Hypercubes aus, die auf der nächsten Seite als Beispiel dargestellt sind.**

Vorgehensweise

- Erzeugen des Hypercube,
- Füllen mit Zahlen (aufsteigend)
  - Beginnend bei **0**
  - Sie brauchen dazu mehrfach verschachtelte Schleifen, für jede Dimension eine Schleife.
  - Ausgeben mit dann mit der `toString()`-Methode, als z.B.
    `System.out.println(hypercube3D);`

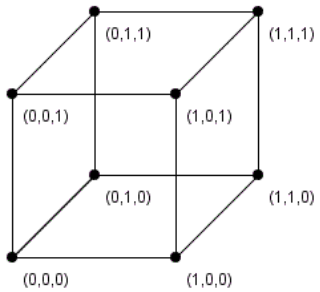
Beispiel 1
Eine Tabelle mit Kantenlänge 4
(**dimension = 2, length = 4**)

```
[0, 0] = 0
[0, 1] = 1
[0, 2] = 2
[0, 3] = 3
[1, 0] = 4
[1, 1] = 5
[1, 2] = 6
[1, 3] = 7
[2, 0] = 8
[2, 1] = 9
[2, 2] = 10
[2, 3] = 11
[3, 0] = 12
[3, 1] = 13
[3, 2] = 14
[3, 3] = 15
```

[0, 0]	[0, 1]		[0, 2]	[0, 3]
[3, 0]	[3, 1]		[3, 2]	[3, 3]

Beispiel 2
Ein Würfel mit Kantenlänge 2
(**dimension = 3, length = 2**)

```
[0, 0, 0]: 0
[0, 0, 1]: 1
[0, 1, 0]: 2
[0, 1, 1]: 3
[1, 0, 0]: 4
[1, 0, 1]: 5
[1, 1, 0]: 6
[1, 1, 1]: 7
```

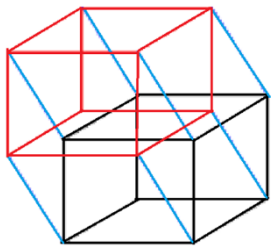


Beispiel 3
Ein 4D-Hyperwürfel mit Kantenlänge 12
(**dimension = 4, length = 12**)

```
[0, 0, 0, 0] = 0
[0, 0, 0, 1] = 1
[0, 0, 0, 2] = 2
[0, 0, 0, 3] = 3
[0, 0, 0, 4] = 4
[0, 0, 0, 5] = 5
[0, 0, 0, 6] = 6
[0, 0, 0, 7] = 7
[0, 0, 0, 8] = 8
[0, 0, 0, 9] = 9
[0, 0, 0, 10] = 10
[0, 0, 0, 11] = 11
[0, 0, 1, 0] = 12

...

[11, 11, 10, 11] = 20723
[11, 11, 11, 0] = 20724
[11, 11, 11, 1] = 20725
[11, 11, 11, 2] = 20726
[11, 11, 11, 3] = 20727
[11, 11, 11, 4] = 20728
[11, 11, 11, 5] = 20729
[11, 11, 11, 6] = 20730
[11, 11, 11, 7] = 20731
[11, 11, 11, 8] = 20732
[11, 11, 11, 9] = 20733
[11, 11, 11, 10] = 20734
[11, 11, 11, 11] = 20735
```



Viel Spaß beim Programmieren!