Spielbrett Informationen

Simon Döring

Inhalt

Die Klasse „FieldType“ 1
Funktionen der Klasse „Board“ 1
public final int getNextFieldByType(FieldType type,int pos) 1
public final int getPreviousFieldByType(FieldType type,int pos) 2
public final FieldType getTypeAt(int pos) 2
Aufgaben 3

Die Klasse „FieldType“

Bevor wir mit der Klasse Board also dem Spielbrett arbeiten können, müssen wir das Enum „FieldType“ kennenlernen. Dieses Enum definiert die einzelnen Felder des Bretts. Es gibt:

CARROT	Karottenfeld
GOAL	Zielfeld
HARE	Hasenfeld
HEDGEHOG	Igelfeld
SALAD	Salatfeld
START	Start
POSITION_1	Positionsfelder
POSITION_2	
INVALID	Ungültig

Die Dokumentation für FieldType ist in /doc/sc/plugin2018/FieldType.html zu finden.

Funktionen der Klasse „Board“

Das Spielbrett kann man sich wie einer Art modifizieren „Array“ [1: Technisch gesehen hat die Klasse eine private Liste von Feldern. Allerdings ähnelt ein Aufruf der Funktion getTypeAt sehr den Aufruf eines Arrays. Auch wenn der Rückgabewert kein Feld sonder ein FieldType ist.] vorstellen. Mithilfe der FieldTypes können wir die Funktionen des Boards verwenden. Zunächst stellt sich allerdings die Frage, wie wir überhaupt das Spielbrett bekommen können. Das Spielbrett bekommen wir durch den GameState. Mit Hilfe der Methode *gameState.getBoard()*.

Unsere Eigene Position können wir mithilfe von **currentPlayer** oder **gameState** herausfinden:

```
// Es wird immer ein Integer zwischen 0 bis 64 zurückgegeben:
currentPlayer.getFieldIndex();
// dasselbe wie oben:
gameState.getCurrentPlayer().getFieldIndex();

if (currentPlayer.getFieldIndex() == gameState.getCurrentPlayer().getFieldIndex()) {
  //wird immer ausgeführt
}
```

public final int getNextFieldByType(FieldType type,int pos)

Dieser Funktion übergeben wir eine Position und ein FieldType. Es gibt die Position des nächsten FieldType an, welches **nach** der angegeben Position liegt. Hat das Feld mit der angegeben Position,

den gleichen FieldType, so wird dennoch die Position des nächsten Felds angegeben (siehe zweites Beispiel). Gibt es diesen FieldType nicht mehr wird -1 zurückgegeben.

```
gameState.getBoard().getNextFieldByType(FieldType.GOAL,
                                        currentPlayer.getFieldIndex());
```

Dies würde in einem normalen Spiel immer 64 zurückgeben, außer man befindet sich auf dem Ziel. Im diesem Fall wäre das Ergebnis -1.

```
// Damit man nicht immer gameState.getBoard() schreiben muss:
Board b = gameState.getBoard();
final int index = currentPlayer.getFieldIndex(); // Index des Spielers

if (index == b.getNextFieldByType(b.getTypeAt(index), index)) {
  System.out.println("Geht nicht");
}
```

Dies liegt daran, dass der verwendete Index nie der Rückgabewert von **getNextFieldByType** sein kann.

```
int next_hedgehog = b.getNextFieldByType(FieldType.HEDGEHOG,
                                         currentPlayer.getFieldIndex());
int next_next_hedgehog = b.getNextFieldByType(FieldType.HEDGEHOG, next_hedgehog);
```

Dies würde die Postion des nächste und vom übernächste Igelfeld bestimmen. Achtung es wird hierbei nicht überprüft, ob es so ein Feld überhaupt gibt. Dies kann zu Fehlern führen. Steht man z.B auf dem letzten Igelfeld, so hat **next_next_hedgehog** den Wert 11, anstelle von -1 (Siehe Übung 2).

public final int getPreviousFieldByType(FieldType type,int pos)

Analog zu getNextFieldByType. Allerdings bezieht sich die Funktion auf das vorherige Feld mit dem entsprechenden FieldType.

public final FieldType getTypeAt(int pos)

Mit dieser Funktionen kann man den FieldType eines bestimmten Feldes ermitteln.

```
FieldType my_field = gameState.getBoard().getTypeAt(currentPlayer.getFieldIndex());
FieldType field_42 = gameState.getBoard().getTypeAt(42);
```

Speichert den aktuelle FieldType auf welchen man steht und den FieldType des Feldes 42.

```
if (gameState.getBoard().getTypeAt(-1) == FieldType.INVALID) {
  // Wird immer ausgeführt, da es das Feld an der Position -1 nicht gibt
}
```

Dieses Beispiel zeigt auf, dass es durch getTypeAt nie zu einer IndexOutOfBoundsException kommen kann. Es gibt nur Felder im Intervall von 0 – 64. Sollte man nach einem Feld außerhalb diese Intervalls fragen, so wird immer INVALID zurückgegeben.

Die komplette API-Dokumentation ist in /doc/sc/plugin2018/Board.html zu finden.

Aufgaben

1. Gib die Entfernung des Gegners zum Startfeld aus. Verwende dabei keine Variablen oder Literale. Die Ausnahme ist der Rückgabewert. Dieser darf eine Literale sein (z.B. return 127;). Tipp: Suche in der API von GameState nach einer Funktion, welchen den anderen Spieler zurückgibt oder lese das Dokument „Hase_Igel_Player“.
2. Erkläre warum **next_next_hedgehog** den Wert 11 hätte, wenn wir auf den letzten Igelfeld stehen würden.

   ```
   int next_hedgehog = b.getNextFieldByType(FieldType.HEDGEHOG,
                                            currentPlayer.getFieldIndex());
   int next_next_hedgehog = b.getNextFieldByType(FieldType.HEDGEHOG, next_hedgehog);
   ```

3. Schreibe eine Funktion, welche das übernächste Igelfeld ausgibt. Gibt es solch ein Feld nicht, so soll immer -1 zurückgegeben werden.