Exercises

W07H02 - Rekursionsraetsel

Points	Submission due date	Status	Difficulty	Categories
0 / 10	in 5 days	No graded result		Hausaufgabe

Tasks:

Rekursionsrätsel - Schneeballturnier der Pinguine

In der Antarktis hat es geschneit, und die Pinguine bereiten sich auf ihr traditionelles Schneeballturnier vor. Deine Aufgabe ist es, den Pinguinen bei der Organisation ihrer Startpositionen zu helfen. Damit die Plätze so schnell wie möglich belegt werden und das Turnier losgehen kann, kannst du die rekursiven Methoden benutzen.

Insgesamt gibt es in der Antarktis 9 Pingu-Familien. Jeder Pingu trägt dabei die Nummer der Familie, also von 1 bis 9. Das gesamte Turnier teilt sich dabei in zwei Runden auf. In der ersten Runde des Turniers wollen die Pinguine natürlich nicht gegen ihre eigene Familie, sondern gegen die anderen Familien spielen. In die zweite Runde kommen nur bestimmt Pinguine. Wie viele das schaffen, steht natürlich nicht fest. Deine Implementierung soll aber für jede Zahl eine mögliche Belegung liefern können.

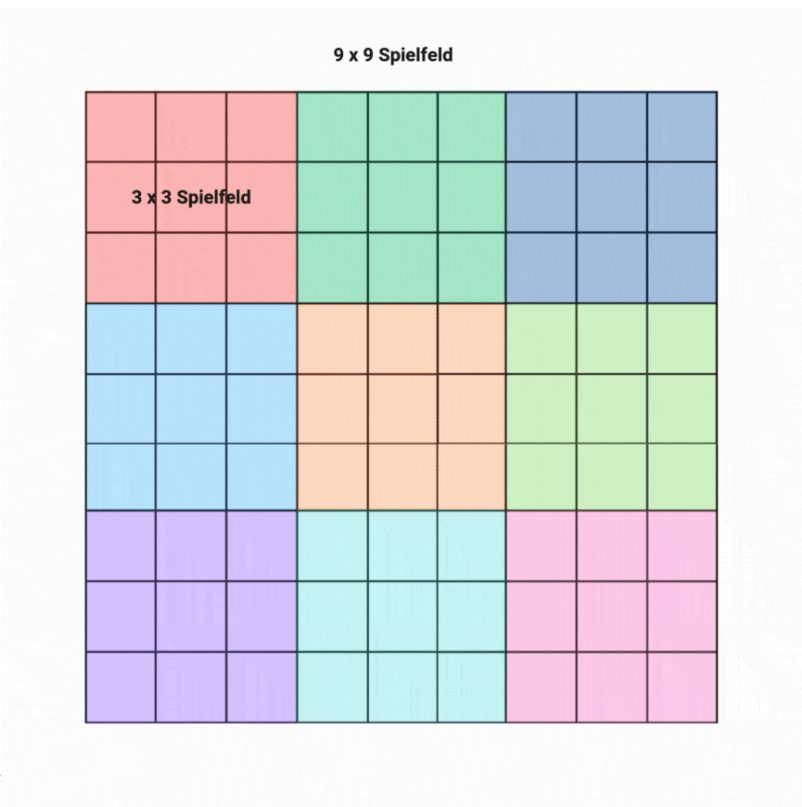
Hinweise 💡

- Die beiden Runden sind von der Geschichte her verbunden. Jedoch kannst du die beiden Teile unabhängig voneinander bearbeiten. Bedeutet: Die Tests sind unabhängig voneinander und mocken vorherige Teilaufgaben.
- Allerdings soll der Gedankengang der ersten Runde dich für die zweite Runde vorbereiten. Empfehlenswert ist es, dass du diese der Reihe nach bearbeitest.
- Deine Abgabe wird mit beliebig großen Feldern getestet. Damit du keine Time Outs bei unseren Tests bekommst, sollst du deine Methoden rekursiv schreiben, besonders die `place..` Methoden. Eine effiziente und iterative Implementierung ist Möglich erfordert jedoch die Simulation des Callstacks der Rekursion. Wir raten davon ab dies zu versuchen.
- Die Lösung des Problems erfordert systematisches Ausprobieren aller Möglichkeiten, wobei ungültige Konfigurationen verworfen werden.
- Beachte, dass das Problem mehrere Lösungen haben kann. Die Methode sollte jedoch stets mindestens eine Lösung finden.
- Du darfst die Zugriffsmodifier auf deinem Template auf keinen Fall ändern. Ansonsten wirst du von unseren Tests einen Build Fail bekommen.
- Es werden nur Eingaben getestet die auch eine Lösung für das Problem erlauben.

Im Template findest du die folgenden Klassen: `BigGameField`, `FirstRoundPositions`,`SecondRoundPositions` & `Penguin`.

Jeder Pinguin hat, wie du in der Klasse `Penguin` sehen kannst, folgende Attribute: `familyNumber`, `isBest`. In der `Penguin` Klasse musst du nichts ergänzen. Wir fangen nun mit der Initialisierung des Spielfelds an.

Das Spiel findet auf einem 9x9-Spielfeld statt, das in 9 kleinere 3x3-Felder unterteilt ist.



Runde 1: Startposition der Pinguine

In dieser Runde gibt es insgesamt 9 Spiele, die jeweils auf den 3x3 Spielfeldern stattfinden. Auf diesen kleineren Spielfeldern sollst du die Pinguine nach ihrer `familyNumber` platzieren. Wie oben erwähnt wurde, gibt es insgesamt 9 Pinguin-Familien, die durch Zahlen (1 bis 9) gekennzeichnet sind. D.h, jeder Pinguin aus Familie 1 hat die `familyNumber` 1.

Ziel: Platziere die Pinguine so, dass folgende Regeln eingehalten werden:

- Zeilenregel: In jeder Reihe darf nur ein Pinguin derselben Familie sein.
- Spaltenregel: In jeder Spalte darf nur ein Pinguin derselben Familie sein.
- Blockregel: In jedem der 9 kleineren 3x3-Felder darf nur ein Pinguin derselben Familie sein.

Deine Aufgabe ist es, die Pinguine aller Familien so zu platzieren, dass die Regeln nicht verletzt werden. Manche Pinguine sind ungeduldig und können nicht mehr auf das Spiel warten. Es kann deswegen sein, dass manche Pinguine sich schon einen Platz gesucht haben, jedoch nicht alle. Gehe davon aus, dass diese Pinguine sich schon an die Regeln gehalten haben. Deine Implementierung soll also das Feld korrekt belegen, auch wenn am Anfang manche Positionen schon durch die Pinguine belegt sind. Nutze dafür die `boolean isFixed(int, int)` Methode aus `BigGameField`. Sie gibt zurück ob der Platz von einem der ungeduldigen Pinguine besetzt wurde.

BigGameField

Diese Klasse verwaltet das 9x9-Spielfeld, das aus 9 3x3-Spielfeldern besteht und überprüft, ob eine Position sicher für einen Pinguin ist.

Konstruktor No results

Der Konstruktor soll ein BigGameField-Objekt erstellen. Es nimmt ein zweidimensionales Array `initialBoard` als Parameter, das die Anfangswerte des Fields enthält. Das Board besteht aus 9 Zeilen und 9 Spalten, wobei jedes Element entweder eine Zahl von 1 bis 9 oder 0 ist (0 steht für ein leeres Feld). Die anfangs bereits belegte Felder sind als `isFixed` bezeichnet. Implementiere den Konstruktor und setze dabei die belegten Felder von `initialBoard` als fixed.

isSafe() No results

Um die Implementierung dieser Methode einfacher zu machen, implementieren wir zunächst die folgenden Hilfsmethoden. Du kannst natürlich auch selber weitere Hilfsmethoden hinzufügen.

- **isColSafe()** No results
  Überprüfe, ob die `col` für den Pinguin mit Nummer `num` sicher ist.
- **isRowSafe()** No results
  Überprüfe, ob die `row` für den Pinguin mit Nummer `num` sicher ist.
- **isBlockSafe()** No results
  Überprüfe, ob der 3x3 Block mit `startRow`und `startCol`für den Pinguin mit Nummer `num` sicher ist. Nutze nun die drei Hilfsmethoden um zu überprüfen, ob ein Pinguin eienr bestimmten Familie an der Position platziert werden kann.

Die Methode `boolean isSafe(int, int)` überprüft ob ein Pinguin einer bestimmten Familie `num` an der Position (`row`, `col`) platziert werden kann, ohne die Regeln zu verletzen. Eine Position ist dann sicher, wenn die Spalte, Reihe und Block, in der sie sich befindet sicher ist.

placePenguin No results

Platziert einen Penguin an der gegebenen Position im Spielfeld. Die Methode `placePenguin(int row, int col, int num)` nimmt die Zeilen-/Spaltennummer von der Position und die Nummer der Pinguin entgegen.

removePenguin No results

Entfernt den Penguin an der gegebenen Position im Spielfeld. Die Methode `removePenguin(int row, int col)` nimmt die Position entgegen, auf der der Pinguin entfernt werden muss. Nach der Entfernung soll diese Position wie am Anfang wieder leer sein.

FirstRoundPositions

Die Klasse `FirstRoundPositions` organisiert die erste Runde des Turniers und implementiert die Logik für die Platzierung der Pinguine.

placeAllPenguins No results

- Entwickle eine rekursive Methode, die alle Pinguine auf dem Spielfeld platziert. Überprüfe dabei, ob alle Regeln eingehalten werden. Falls eine gültige Lösung gefunden wird, gibt die Methode `true` zurück, andernfalls `false`. Denk hier daran, dass dir viele Methoden aus der Klasse `BigGameField`zur Hilfe stehen.

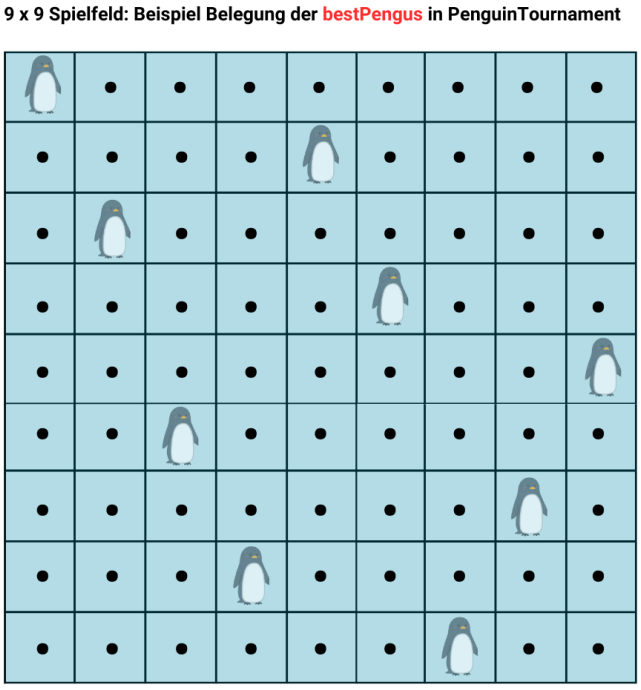
Durch das Ausführen der `main` -Methode könnt ihr eure Implementierung testen. Die Methode `solvePuzzle()` startet den Prozess zum Lösen des Pinguin-Puzzles. Es versucht, Pinguine beginnend bei der Zelle (0,0) durch den Aufruf `placePenguins()` zu platzieren. Deine Implementierung soll bei den vorgegebenen Beispiele, auf die leeren Plätze Pinguine setzen können, ohne dass die oben gennannten Regeln der ersten Runde verletzt werden.

▶ Eine richtige Ausgabe vom 9x9-Spielfeld kann z.B. wie folgt aussehen:

Runde 2: Turnier der Queens

Nach der ersten Runde treten die besten Pinguine als `Queens` in einem neuen Turnier an.

9 x 9 Spielfeld: Beispiel Belegung der bestPengus in PenguinTournament



Regeln für Runde 2

- Zeilenregel: Jede Zeile darf nur eine *Queen* enthalten.
- Spaltenregel: Jede Spalte darf nur eine *Queen* enthalten.
- Diagonalenregel: Keine Diagonale darf mehr als eine *Queen* enthalten.
  Das Ziel ist es, alle Queens gemäß diesen Regeln auf dem Feld zu platzieren. Am Anfang steht nicht fest, wie viele Queens in der zweiten Runde spielen werden. Deine Implementierung soll für beliebige Größen des Feldes und die Anzahl der Queens die richtige Belegung finden. Das Feld ist immer quadratisch und die Kanten sind genauso lang wie die Anzahl der Queens.

SecondRoundPositions

Diese Klasse organisiert die zweite Runde des Turniers und platziert die besten Pinguine gemäß den Regeln.

Am Anfang ist das Feld leer, also mit `'.'` gefüllt, da sich noch keine *Queen* auf dem Feld befindet. Die Liste `bestPenguins` enthält die Pinguine, die du für die zweite Runde auf das Feld platzieren sollst.

Wie in der ersten Runde hast du hier wieder Hilfsmethoden, die du für die Methode `placeQueens` brauchst. Fange am Besten mit den Hilfsmethoden an. In der folgenden Teilaufgabe erklären wir dir genauer wieder Algorithmus funktioniert. Für die folgenden Methoden, die auf Regelverstöße checken, reicht es aus nach Regelverstößen links der aktuellen Spalte zu suchen. Alle Spalten auf der rechten Seite der aktuellen Spalte können ignoriert werden.

◉ **isSafe(int row, int col)** No results
Eine Position ist in der zweiten Runde genau dann safe, wenn die Diagonale, Spalte & Zeile safe sind. Implementiere die einzelnen Methoden, die diese Regeln überprüfen müssen.

- ◉ **isUpperDiagonalSafe(int row, int col)** No results
  Überprüfe, ob auf der oberen Diagonale sich schon eine `Queen` befindet. Es reicht aus die linke-obere Diagonale zu untersuchen.
- ◉ **isLowerDiagonalSafe(int row, int col)** No results
  Überprüfe, ob auf der unteren Diagonale sich schon eine `Queen` befindet. Es reicht aus die linke-untere Diagonale zu untersuchen.
- ◉ **isRowSafe(int row, int col)** No results
  Überprüfe, ob auf der Zeile sich schon eine `Queen` befindet. Es reicht aus die Einträge der Zeile die sich links von der gegebenen Spalte befinden zu untersuchen.

Nachdem du die Hilfsmethoden für die Regeln implentiert hast, kannst du nun die Queens platzieren.

- ◉ **placeQueens(int col)** No results
  Implementiere nun die Methode `placeQueens(int col)`. Um eine *Queen* auf das Feld zu platzieren sollst du diese Position durch `'Q'` ersetzen. Wie du es auf dem Template sehen wirst, sind hier die leeren Positionen durch `'.'` gekennzeichnet.

Der Parameter `col` in der Methode `placeQueens(int col)` gibt die aktuelle Spalte an, in der versucht wird, eine Queen zu platzieren. Der Algorithmus soll von links nach rechts arbeiten, indem er in jeder Spalte (beginnend bei 0) prüft, ob eine sichere Position in einer der Zeilen gefunden werden kann. Sobald eine Queen platziert ist, wird rekursiv zur nächsten Spalte weitergegangen.

Deine Implementierung kannst du innerhalb der `main`-Methode testen. Wenn du es richtig implentiert hast, soll deine Methode für alle Eingaben `tournament.placeQueens(col)` eine Lösung ausgeben.

▶ Die richtige Ausgabe mit `tournament.placeQueens()` mit 9 Queens soll wie folgt aussehen:

Exercise details

Release date	Dec 5, 2025 17:00
Submission due date	Dec 14, 2025 17:00
Complaint possible	No