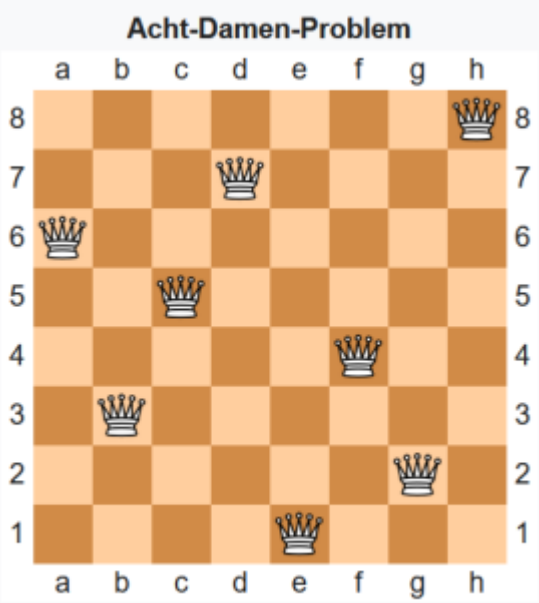
8-Damen-Problem

Das 8-Damen-Problem ist ein mathematisches Rätsel auf Basis eines Schachbretts. Die Aufgabe ist es, 8 Damen auf dem Schachbrett so zu platzieren, dass sich keine zwei Damen gegenseitig schlagen können. Als Erinnerung: eine Dame kann sich von ihrem aktuellen Feld ausgehend senkrecht, waagrecht und auch diagonal beliebig weit bewegen! Anders ausgedrückt besteht die Aufgabe also darin, die Damen so zu platzieren, dass sich keine zwei Damen in einer Reihe, einer Spalte oder einer Diagonalen befinden.



eine der Lösungen

Für dieses Rätsel gibt es tatsächlich ganze 92 Lösungen (schließt man symmetrisch identische Lösungen durch Drehung und Spiegelung aus, so verbleiben 12 Lösungen).

Algorithmus unplugged

Wie würde man das Problem algorithmisch lösen? Am besten probiert man es einmal ohne PC an einem normalen Schachbrett aus - wenn du keines zur Hand hast, kannst du diese Aktivität im Browser verwenden, um die 8 Damen auf einem virtuellen Schachbrett zu platzieren.

Lies hier erst weiter, nachdem du dir selbst Gedanken gemacht hast, wie du vorgehen könntest.

Wichtig ist die Erkenntnis aus der Einleitung: In keiner Reihe, Spalte oder Diagonalen dürfen sich zwei Damen befinden. Es ist daher sinnvoll mit der **ersten Reihe** zu beginnen und dort eine Dame zu platzieren. Um systematisch vorzugehen, werden wir in jeder Reihe die Damen zuerst ganz links platzieren und diese dann stückchenweise nach rechts verschieben. Als Tipp zur Vorstellung: In der Informatik beginnt das Koordinatensystem immer links oben. Die erste Reihe ist also die oberste! Bezogen auf das dargestellte Schachbrett steht die Dame in A8.

Wenn die erste Reihe mit einer Dame gefüllt ist, ist klar, dass es danach mit der zweiten Reihe weitergehen muss. Auch dort versucht man zunächst, die Dame ganz links zu platzieren (A7). Dort kommt es natürlich direkt zu einer Kollision mit A8. Daher verschiebt man die Dame der zweiten Reihe eins nach rechts und probiert B7. Auch dort gibt es wegen der Diagonalen eine Kollision. Also noch eins weiter verschieben auf C7 - nun ist die Dame "sicher".

Die dritte Dame wird in der dritten Reihe auf A6 platziert und du kannst dir sicher schon denken, wie

es weitergeht!

Pseudocode - Wo ist die Rekursion?

Was hat das ganze nun mit Rekursion zu tun? Die Rekursion ist im Befüllen der einzelnen Reihen zu finden. Tatsächlich ist der Algorithmus so stark reduziert, dass es praktisch nur eine Methode gibt, deren Aufgabe es ist, eine einzelne Reihe korrekt/fehlerfrei zu befüllen. Welche Reihe das ist, wird als Parameter übergeben. Damit kann die Methode rekursiv wieder aufgerufen werden um die erste, dann die zweite, dann die dritte Reihe [...] zu füllen.

```
// die Methode befuelleZeile(zeile) hat nur die Aufgabe,
// die übergebene Zeile fehlerfrei zu befüllen
befuelleZeile(zeile):
    wenn zeile < 8:
        für spalte von 0 bis 7:
            wenn istPositionErlaubt(zeile, spalte):
                setze Dame ins Feld [zeile][spalte]
                befuelleZeile(zeile+1)              // Nächste Zeile
befüllen
                loesche Dame im Feld [zeile][spalte] // Warum?
    sonst:
        druckeFeld()  // Lösung gefunden - gib das komplette Feld aus!
```

Man startet diese Methode mit dem Aufruf von `reihe(0)`.

(A1)

Arbeite mit der folgenden Vorlage: https://codeberg.org/qg-info-unterricht/bluej-8-Damen-console

(A) Modelliere das Feld mit einem zweidimensionalen Array des Typs `boolean`. Definiere das Feld als Attribut der Klasse, initialisiere es im Konstrunktor.

(B) Implementiere eine Methode `druckeFeld()`, die den gesamten Zustand des Schachbretts ausdruckt. Du kannst bei der Ausgabe z.B. einen Unterstrich für ein leeres Feld des Bretts, X für ein Feld mit einer Dame verwenden[1)]. Teste deine Methode, indem du im Konstruktor das leere Feld ausgeben lässt, eine oder mehrere Damen setzt und dann das Feld mit den Damen ausgeben lässt.

(C) Implementiere eine Testmethode, die zurück gibt, ob an der aktuellen Position eine Dame gesetzt werden darf.

Um zu testen, ob eine Dame an eine bestimmte Position gesetzt werden darf, implementiert man die Methode

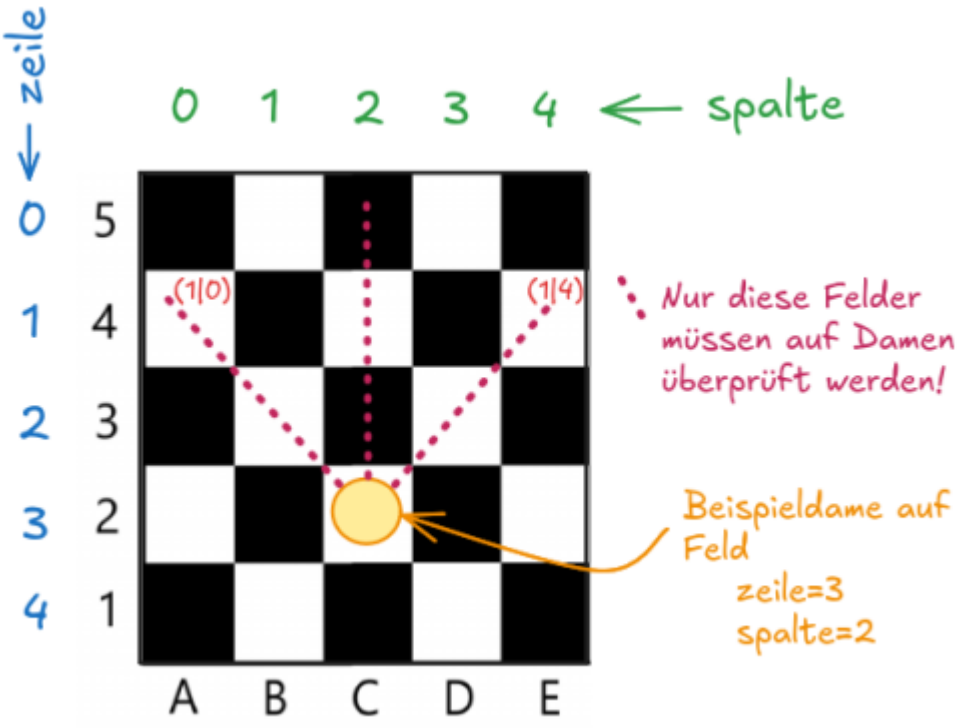
```
boolean istPositionErlaubt(int zeile, int spalte)
```

Diese gibt `true` zurück, wenn die Dame platziert werden darf, andernfalls `false`.

Einige Überlegungen zu dieser Methode:

1. Zeilen unterhalb der aktuellen Zeile muss man bei der Lösung des N-Damen-Problems nicht betrachten, da wir die Damen immer zeilenweise von oben platzieren. Unterhalb der aktuellen Zeile befinden sich also niemals Damen auf dem Brett.
2. Die aktuelle Zeile muss ebenfalls nicht überprüft werden, da hier noch keine Dame platziert ist.
3. Wir müssen also nur die Diagonalen oberhalb und die Spalte oberhalb der Position überprüfen, auf die wir gerne eine Dame platzieren würden.

Die Skizze stellt die Situation dar. Jetzt kann man sich überlegen, wie man an die Koordinaten der "gestrichelten" Felder kommt, die man überprüfen muss:



- Die Felder in der Zeile oberhalb der aktuellen Position haben die Koordinaten `O(0…zeile-1|spalte)`
- Wenn man mit der Variablen `zeileAktuell` zeilenweise von oben nach unten über das Feld iteriert, haben die Felder der "linken" Diagonalen die Koordinaten `L(zeileAktuell|spalte-(zeile-zeileAktuell)`. Die Felder der rechten Diagonalen haben die Koordinaten `R(zeileAktuell|spalte+(zeile-ZeileAktuell))`. In beiden Fällen muss man jedoch darauf achten, dass die berechnete Spalte sich innerhalb des erlaubten Bereichs für die Spaltenindex befindet: 0 bis N-1. Im Beispiel gibt es in Zeile 0 nämlich weder links noch rechts ein Feld auf der Diagonalen, da sowohl `2-(3-0)=-1` als auch `2+(3-0)=5` sich außerhalb des Feldes befinden. In Zeile 1 erhalten wir aber die Koordinaten der beiden Diagonalenfelder: `L(1|2-(3-1)=L(1|0)` und `R(1|2+(3-1))=R(1|4)`.

(D) Teste deine Methode `istPositionErlaubt`:

- Erzeuge ein Schachbrett 8x8
- Setze eine Dame auf das Feld `(0|2)`
- Überprüfe, ob deine Methode beim Aufruf aus der Objektleiste die korrekten Antworten für mögliche Damen auf den Feldern `(1|0)`, `(1|1)`, `(1|2)`, `(1|3)` und `(1|6)` gibt.

Hilfestellung 1 zu "istPositionErlaubt":

```
public boolean istPositionErlaubt(int zeile, int spalte) {
        // Füge hier deinen Code für die Kollisionserkennung ein.
        // Einen Lösungsvorschlag findest du im Wiki

        // Alle Zeilen von 0 bis zeile-1 untersuchen:
        for(int zeileAktuell=0; zeileAktuell<zeile; zeileAktuell++) {
            // Felder oberhalb untersuchen, wenn Dame:
            // return false
            if( FIXME ) {
                return false;
            }
            // Felder der Diagonalen untersuchen, wenn Dame:
            // return false
            // Linke Diagonale
            if( FIXME ) {
                return false;
            }
            // Rechte Diagonale
            if( FIXME ) {
                return false;
            }
        }
        return true;

    }
```

Lösungsvorschlag zu "istPositionErlaubt":

```
public boolean istPositionErlaubt(int zeile, int spalte) {
        // Füge hier deinen Code für die Kollisionserkennung ein.
        // Einen Lösungsvorschlag findest du im Wiki

        // Alle Zeilen von 0 bis zeile-1 untersuchen:
        for(int zeileAktuell=0; zeileAktuell<zeile; zeileAktuell++) {
            // Felder oberhalb untersuchen, wenn Dame:
            // return false
            if(feld[zeileAktuell][spalte]) {
                return false;
            }
            // Felder der Diagonalen untersuchen, wenn Dame:
            // return false
            if(spalte - (zeile-zeileAktuell) >= 0 &&
feld[zeileAktuell][spalte - (zeile-zeileAktuell)] ) {
                return false;
            }
            if( spalte + (zeile-zeileAktuell) < SIZE &&
```

```
feld[zeileAktuell][spalte + (zeile-zeileAktuell)] ) {
                return false;
            }
        }
        return true;

    }
```

(E) Wenn du geprüft hast, dass deine `istPositionErlaubt`-Methode korrekt funktioniert, kannst du den Pseudocode für `befuelleZeile` in deinem BlueJ-Projekt implementieren.

(A2)

Was ist der Basisfall dieser Rekursion?

(A3) - Für die Schnellen

Erweitere das Programm: Lass dir ausgeben, wie viele Lösungen gefunden wurden.

(A4) - Für die Schnellen

Wenn du früh fertig bist, dann kannst du überlegen, ob du selbst eine andere/eigene Lösung für die Kollisionsprüfung auf der Diagonalen findest (das ist ziemlich knifflig, wenn man nicht den mathematischen Kniff der Vorlage nutzt).

Auf der folgenden Seite siehst du visualisiert dieselbe Vorgehensweise bei der Lösungssuche: https://www.mathematik.ch/spiele/N_Damenproblem/

1)
Du kannst auch Unicode Zeichen wie ⬚ und ♕ verwenden