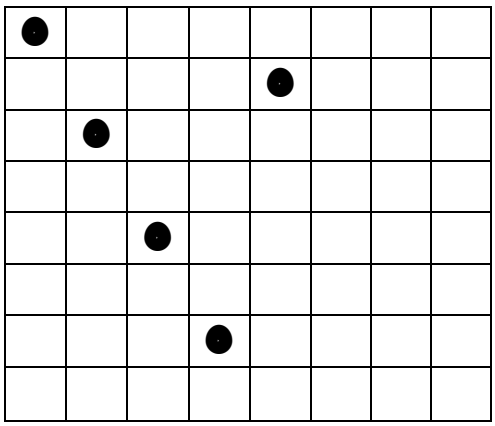
DAS N-DAMEN-PROBLEM



Die Verwaltung der Damen geschieht in einem Array für die acht Spalten: `damen: int[0..7]`. Jeder Eintrag `damen[i]` entspricht der Zeilenposition der Dame in Spalte i.

Umgangssprachlich kann man den Lösungsalgorithmus wie folgt formulieren:

- Starte mit der ersten Dame in der ersten Spalte.
- Mache hier für jede Zeilenposition folgendes:
  - Setze die Dame an die entsprechende Stelle
  - Probiere nun, die restlichen Damen in den übrigen Spalten zu setzen.
  - Falls es geklappt hat,
    dann haben wir eine Lösung und sind fertig
    sonst nehme die Dame wieder vom Spielbrett.

Bei dem umgangsprachlichen Befehl „Probiere nun, die restlichen Damen in den übrigen Spalten zu setzen“ kann man genau den gleichen Lösungsalgorithmus verwenden – mit dem Unterschied, dass man nun mit der zweiten Dame in der zweiten Spalte anfängt. Das gleiche gilt schließlich auch für die 3., 4. ... und 8. Dame. Wir erhalten also einen rekursiven Algorithmus, einen sogenannten Backtrackingalgorithmus:

ALGORITHMUS	***versucheDameZuSetzen***	
Input:	spalte:	int
	damen:	int[0..7]
Lokal.	zeile:	int

- Für zeile von 0 bis 7 tue folgendes:
  - Falls NICHT d*ameBedroht(spalte, zeile)*
    dann
    - damen[spalte] ← zeile
    - s*etzeBedrohung(spalte, zeile)*
    - Falls spalte = 7
      dann
      - Zeige die Lösung an

      sonst
      - ***versucheDameZuSetzen (spalte + 1, damen)***
    - *loescheBedrohung(spalte, zeile)*
    - damen[spalte] ← 0

Die Frage, ob *dameBedroht(spalte, zeile)* ist und die Routinen für s*etzeBedrohung(spalte, zeile)* und *loescheBedrohung(spalte, zeile)* müssen nun ein wenig konkretisiert werden.

Eine Dame bedroht die Spalte und die Zeile, in der sie steht. Da wir die Datenstruktur so gewählt haben, dass in jedem Fall nur maximal eine Dame in einer Spalte gesetzt ist, erübrigt sich die Frage, ob eine neu zu setzende Dame durch eine andere Dame in der gleichen Spalte bedroht sein könnte. Anders verhält es sich mit der Zeile. Hierfür wollen wir uns ein Feld anlegen, in dem festgehalten wird, welche Zeilen bereits von Damen besetzt sind und somit andere Damen in diesen Zeilen bedrohen würden. Sinnvollerweise wählen wir

```
boolean[] zeileBedroht = new boolean[8]
```

Neben der Zeile und Spalte werden durch eine Dame auch die Diagonalen (Aufwärtsdiagonale und Abwärtsdiagonale) bedroht. Schauen wir uns dazu zwei Beispiele an:

	0	1	2	3	4	5	6	7
0								
1								
2								●
3							●	
4						●		
5					●			
6				●				
7			●					

	0	1	2	3	4	5	6	7
0			●					
1				●				
2					●			
3						●		
4							●	
5								●
6								
7								

Im ersten Beispiel liegen alle Damen auf einer Aufwärtsdiagonalen – bedrohen sich also gegenseitig. Die Besonderheit hieran ist, dass für alle Damen gilt: Zeile + Spalte = Konstant. In diesem Fall ist Zeile + Spalte = 9, d. h. die 9. Aufwärtsdiagonale wird von den eingetragenen Damen belegt. Insgesamt gibt es 15 Aufwärtsdiagonalen, deren eventuelle Bedrohung durch eine Dame festgehalten werden muss. Dazu wählen wir ein Feld

```
boolean[] aufDiagBedroht = new boolean[15]
```

in dem beim setzen einer Dame das Feldelement mit dem Index Zeile+Spalte auf true gesetzt wird. Die Nummerierung ergibt sich daraus, dass Zeile + Spalte minimal 0 (0+0) und maximal 14 (7+7) ergeben kann.

Im zweiten Beispiel liegen alle Damen auf einer Abwärtsdiagonalen Die Besonderheit an dieser Situation ist, dass für alle Damen gilt: Spalte – Zeile = Konstant. In diesem Fall ist Spalte – Zeile = 2, d. h. die 2. Abwärtsdiagonale wird von den eingetragenen Damen belegt. Insgesamt gibt es auch hier 15 Abwärtsdiagonalen, deren Bedrohung durch eine Dame festgehalten werden muss. Wählen wir auch hier wieder ein Feld mit 15 Einträgen, so sollten wir die Nummerierung umstellen. Spalte – Zeile kann minimal den Wert –7 (0–7) und maximal den Wert +7 (7–0) annehmen. Daraus ergibt sich die Feldvariable

```
boolean[] abDiagBedroht = new boolean[15]
```

in der beim setzen einer Dame das Feldelement mit dem Index Spalte – Zeile + 7 auf true gesetzt wird.

Der Algorithmus für das setzen und löschen einer Bedrohung sieht damit relativ einfach aus:

ALGORITHMUS	***setzeBedrohung***	***loescheBedrohung***
Input:	spalte, zeile: int	
• zeileBedroht[zeile] ← true		..... ← false
• aufDiagBedroht [zeile + spalte] ← true		..... ← false
• abDiagBedroht [spalte – zeile + 7] ← true		..... ← false

Der Algorithmus zur Feststellung, ob *dameBedroht(spalte, zeile)* ist, ist dann nur noch eine Formsache:

ALGORITHMUS	***dameBedroht***
Output:	boolean
Input:	spalte, zeile: int
• return	zeileBedroht[zeile] ODER aufDiagBedroht [zeile + spalte] ODER abDiagBedroht [spalte – zeile + 7]