Mehrdimensionale Arrays: Schiffe versenken

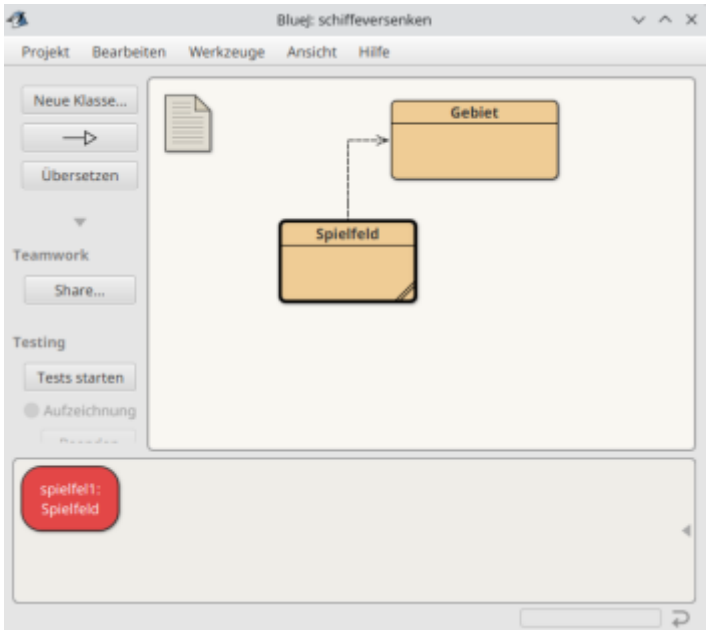
Ein nettes Beispiel für ein zweidimensionales Array ist das Spiel Schiffe versenken. Um das Spielfeld zu modellieren, kann man ein mehrdimensionales Array einsetzen. Dabei werden die Elemente, ähnlich wie beim Koordinatensystem in Mathe die x- und y-Koordinate, durch zwei Indizes angesprochen.

Das Spielfeld als zweidimensionales Array

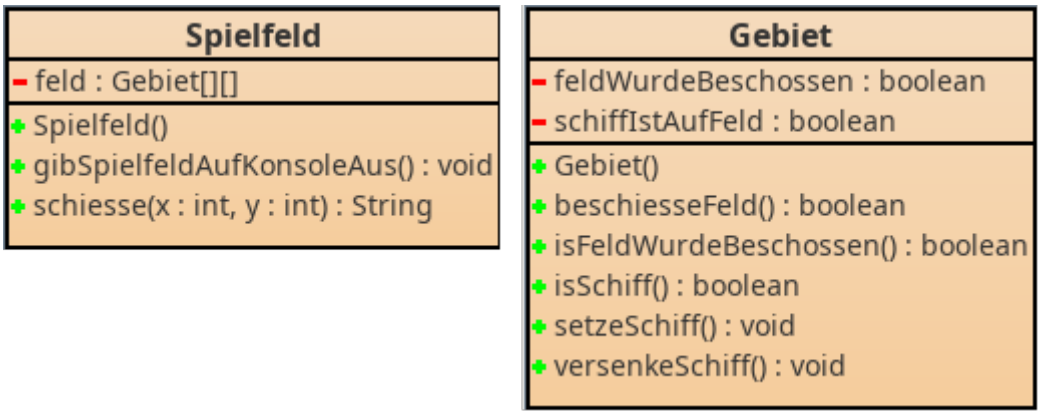
Mehrdimensionale Arrays definiert man folgendermaßen:

```
  private int[][] zahlenquadrat = new int[10][10];
```

Die Modellierung für das Schiffe-Versenken-Spiel sieht so aus:



Die Klassen Gebiet und Spielfeld sind dabei folgendermaßen angelegt:



Das **Spielfeld** hat im Wesentlichen die Aufgabe, die Gebiete zu verwalten. Daher bekommt es nur eine Eigenschaft: `feld` ist ein zweidimensionales Array vom Typ `Gebiet`. Auch hier wird das "zweidimensional" durch zwei leere eckige Klammerpaare realisiert: `Gebiet[][]`.

```
public class Spielfeld {
  // Eigenschaften
  private Gebiet[][] feld;
  // Konstruktor
  public Spielfeld() {
    feld = new Gebiet[10][10];
    // Anfangswerte setzen für alle 100 Felder
    for (int i = 0; i < 10; i++) {
      // senkrecht
      for (int j = 0; j < 10; j++){
        //waagerecht
        feld[j][i]=new Gebiet();
      }
    }
    feld[1][2].setzeSchiff();
    feld[2][2].setzeSchiff();
    feld[3][2].setzeSchiff();
  }
(...)
```

Hinweise zum Beispielcode für das Spielfeld

Das Anlegen eines zweidimensionalen Arrays geschieht ähnlich wie das eines eindimensionalen. mit der Anweisung new `Gebiet[10][10]` legt man die Größe des Arrays fest: 10x10-Felder beim Schiffeversenken. Wie bei eindimensionalen Arrays gehen die Indizes jeweils von 0 bis 9.

Um das Array sinnvoll zu initialisieren müssen mit Hilfe einer verschachtelten Schleife alle Felder abgegangen werden: Erst Feld 0,0 dann 0,1 dann 0,2 …. 0,9 dann 1,0 dann 1,1 dann 1,2…dann 1,9 dann 2,0 usw. bis 9,9).

Dann wird wird beispielhaft ein 3er Schiff gesetzt. Dazu müssen 3 Felder einzeln angesprochen werden.

Aufgaben

- Analysiere auf Grundlage der beiden UML-Klassendiagramme und auf Grundlage deiner Kenntnisse über das Schiffeversenken-Spiel die Klassen `Gebiet` und `Spielfeld`. Überlege dir, was die einzelnen Methoden bewirken (sollen), ohne dass Sie den folgenden Quelltext anschauen. Vergleiche anschließend mit dem Quelltext.
- Nenne weitere Beispiele aus dem Alltag, bei denen man Zwei- oder Mehrdimensionale Arrays benutzen könnte.

- Analysiere die Methoden `schieße(int x, int y)` und `gibSpielfeldAufKonsoleAus()`
- Warum kann Spielfeld nicht direkt die Eigenschaften von Gebietver wenden?
- Implementiere die Methoden `setzeSenkrechtesSchiff()` und `setzeWaagerechtesSchiff()`, die ein Schiff auf das Spielfeld eintragen. Überlege zunächst, welche Übergabeparameter benötigt werden. Entwerfe einen Plan, wie man vermeiden kann, dass der Spielfeldrand überschritten wird.
- Schreibe Methode `pruefeObGewonnen()`, die dann `true` zurückgibt, wenn auf dem gesamten Spielfeld kein Schiff mehr steht.

Quelltexte

```
public class Gebiet
{
   // Eigenschaften
   private boolean feldWurdeBeschossen;
   private boolean schiffIstAufFeld;

    // Konstruktor
   public Gebiet()
   {
      feldWurdeBeschossen=false;
      schiffIstAufFeld=false;
   }

   // Methoden
   public boolean beschiesseFeld(){
        feldWurdeBeschossen=true;
        if (schiffIstAufFeld) {
            versenkeSchiff();
            return true;
        }
        else return false;
   }

   public void setzeSchiff(){
        schiffIstAufFeld=true;
   }

   public void versenkeSchiff(){
        schiffIstAufFeld=false;
   }

   public boolean isFeldWurdeBeschossen(){
        return feldWurdeBeschossen;
   }

   public boolean isSchiff(){
        return schiffIstAufFeld;
   }
```

```
} // Ende der Klasse
```

```
public class Spielfeld
{
    // Eigenschaften
   private Gebiet[][] feld;

    // Konstruktor
    public Spielfeld()
    {
       feld = new Gebiet[10][10];
       // Anfangswerte setzen fuer alle 100 Felder
       for (int i=0;i<10;i++){ // senkrecht
         for (int j=0;j<10;j++){ //waagerecht
           feld[j][i]=new Gebiet();
         }
       }
       feld[1][2].setzeSchiff();
       feld[2][2].setzeSchiff();
       feld[3][2].setzeSchiff();
    }

    // Methoden
    public String schiesse(int x, int y){
       if (feld[x][y].beschiesseFeld()) return "Treffer";
       else return "Daneben";
    }

    public void gibSpielfeldAufKonsoleAus(){
       System.out.println("\n  0123456789"); // Leerzeile + Beschriftung
       for (int i=0;i<10;i++){
         System.out.print(i+" ");
         for (int j=0;j<10;j++){
           if (feld[j][i].isFeldWurdeBeschossen()) System.out.print("*");
           else {
              if (feld[j][i].isSchiff()) System.out.print("X");
              else System.out.print("-");
           }
         }
         System.out.println(); // Zeilenwechsel
       }
    }

} // Ende der Klasse
```