

13.10 Reihungen

Weitere Namen: Feld, Array

- Einstufige Reihungen
- Mehrstufige Reihungen
- Fallstudie: Tic Tac Toe

Beispiel

```

/* powersOfTwo initializes an array to powers of two
 * @param n - size of the array */
public static void powersOfTwo (int n)
{
    int[] r = new int[n];
    int i = 0;
    r[0] = 1;
    while (i != r.length-1)
        { // Invariante: für 0 <= j <= i gilt r[j] == 2^j
          r[i+1] = 2*r[i];
          i = i+1;
        }
}

```

Reihung als Rückgabewert

```

/* vectorAdd adds two double arrays of same size
 * @param v - first summand
 * @param w - second summand
 * @return the sum of v and w, null if v and w have different sizes */
public static double[] vectorAdd (double[] v, double[] w)
{ if (v.length != w.length)
  { return null; }
  else
  { double result[] = new double[v.length];
    int i = 0;
    while (i != result.length)
      { result[i] = v[i] + w[i];
        i = i + 1;
      }
    return result;
  }
}

```

Reihungen von Objekten

```

BankAccount[] account = new BankAccount[100];
account[1] = new BankAccount(20);
account[1].deposit(500);
int i = 0;
while ( i != account.length )
  { if ( account[i] != null )
    { System.out.println( "Balance of account " + i
      + " is " + account[i].balance() );
    }
    i = i+1;
  }

```

• falls $i == 0$ oder $i > 1$ && $i < 100$, so gilt $account[i] == null$ (Startwert)

13.10.1 Einstufige Reihungen

```
int[] r = new int [6];
```

- r ist *kein Objekt*, aber besitzt eigene Identität
- r enthält sechs Werte vom Typ `int`
- Zugriff durch $r[0]$, $r[1]$, ..., $r[5]$
- 0, ..., 5 *Indizes* der Reihung
- Anzahl der Werte: $r.length == 6$
- Startwert: $r[i] == 0$

- nach **Ausführung von** `int[] r = new int[n];`
 $r \rightarrow$

0	0	0	0	0	0
---	---	---	---	---	---
- nach **Ausführung von** `r[0] = 1;`
 $r \rightarrow$

1	0	0	0	0	0
---	---	---	---	---	---
- nach **erstem Schleifendurchlauf**
 $r \rightarrow$

1	2	0	0	0	0
---	---	---	---	---	---
- nach **zweitem Schleifendurchlauf**
 $r \rightarrow$

1	2	4	0	0	0
---	---	---	---	---	---
- nach **letztem (fünftem) Schleifendurchlauf**
 $r \rightarrow$

1	2	4	8	16	32
---	---	---	---	----	----

`vectorAdd (v, w)` mit

$v \rightarrow$

1	2	4
---	---	---

$w \rightarrow$

5	5	5
---	---	---

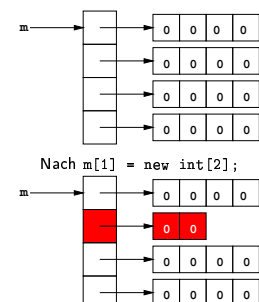
liefert **neue Reihung** `result` mit

`result` \rightarrow

6	7	9
---	---	---

13.10.2 Mehrstufige Reihungen

- Mehr als eine Stufe möglich:
`int[] [] m = new int [4] [4];`
`m[2][3] = 5;`
- Darstellung: Reihung, deren Elemente wieder Reihungen sind
- Zweistufige Reihung: Matrix
- Achtung:
`m[1] = new int[2];`
 ist erlaubt!



Beispiel

```

/* transpose transposes a rectangular matrix
 * @param a - input matrix
 * @return the transposed matrix */
public static double[][] transpose (double[][] a)
{ int m = a.length;
  int n = a[0].length;
  double[][] answer = new double[n][m];
  int i = 0;
  while (i != m)
  { int j = 0;
    while (j != n)
    { answer[j][i] = a[i][j];
      j = j + 1;
    }
    i = i + 1;
  }
}

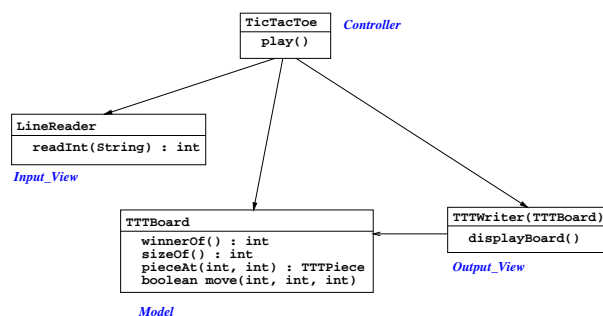
```

Vorläufige Version

181

© 2001 Peter Thiemann

13.10.3 Fallstudie: TicTacToe



Vorläufige Version

182

© 2001 Peter Thiemann

```

public class TTTBoard
{
  private int SIZE = 3;
  private int winner;
  private int nrOfMoves;
  private TTTPiece[][] board;

  public TTTBoard()
  {
    board = new TTTPiece[SIZE][SIZE];
    winner = 0;
    nrOfMoves = 0;
  }
}

```

Vorläufige Version

183

© 2001 Peter Thiemann

```

/** winnerOf reports the state of the game
 * @return 0 if the game is still going,
 *         -1 if it's a draw,
 *         the number of the winning player, otherwise */
public int winnerOf()
{ return winner; }

public int sizeOf()
{ return SIZE; }

public TTTPiece pieceAt(int x, int y)
{ if (x < 0 || x >= SIZE || y < 0 || y >= SIZE)
  { return null; }
  else
  { return board[x][y]; }
}

```

Vorläufige Version

184

© 2001 Peter Thiemann

```

public boolean move(int player, int x, int y)
{ if (x < 0 || x >= SIZE
     || y < 0 || y >= SIZE // illegal move: outside board
     || board[x][y] != null // illegal move: square occupied
     || winner != 0) // illegal move: game over
  { return false; }
  else
  { nrOfMoves = nrOfMoves + 1;
    board[x][y] = new TTTPiece (player);
    if (winning(player, x, y))
    { winner = player; }
    else if (nrOfMoves == SIZE*SIZE)
    { winner = -1; }
    return true;
  }
}

```

Vorläufige Version

185

© 2001 Peter Thiemann

```

private boolean winning(int player, int x, int y)
{ return winningInDirection(player, x, y, 0, 1)
  || winningInDirection(player, x, y, 1, 0)
  || x == y && winningInDirection(player, x, y, 1, 1)
  || x + y == SIZE - 1 && winningInDirection(player, x, y, 1, SIZE-1);
}

private boolean winningInDirection(int player, int x, int y, int dx, int dy)
{ int i = 1;
  boolean answer = true;
  while (i != SIZE)
  { answer = answer && checkFor(player, x+i*dx, y+i*dy);
    i = i + 1;
  }
  return answer;
}

```

Vorläufige Version

186

© 2001 Peter Thiemann

```

/** checkFor tests if a piece from player is present at (x,y)
 * @param player - an integer > 0
 * @param x - x position on game board, x>=0, treated %SIZE
 * @param y - y position on game board, y>=0, treated %SIZE
 * @return true iff player's piece is at (x,y) */
private boolean checkFor(int player, int x, int y)
{ int ix = x % SIZE;
  int iy = y % SIZE;
  TTTPiece atXY = board[ix][iy];
  return atXY != null && atXY.ownerOf() == player;
}

```

Vorläufige Version

187

© 2001 Peter Thiemann

```

public class TTTPiece
{
  private int owner;

  public TTTPiece (int who)
  { owner = who; }

  public int ownerOf ()
  { return owner; }
}

```

Vorläufige Version

188

© 2001 Peter Thiemann

```

public class TTTTextWriter
{ private TTTBoard board;

  public TTTTextWriter (TTTBoard b)
  { board = b; }

  /** displayBoard prints the TicTacToe board on the terminal */
  public void displayBoard ()
  { System.out.println("Current Board");
    int size = board.sizeOf();
    int x = 0;
    while (x != size)
      { System.out.print(" " + x + " ");
        x = x + 1;
      }
    System.out.println();
  }
}

```

```

int y = 0;
while (y != size)
  { x = 0;
    while (x != size)
      { TTPiece atXY = board.pieceAt(x, y);
        if (atXY == null)
          { System.out.print (" _ "); }
        else
          { System.out.print (" " + atXY.ownerOf() + " "); }
          x = x + 1;
        }
      System.out.println();
      y = y + 1;
    }
  }
}

```

Syntax (Reihungen)

```

⟨expr⟩ ::= ⟨ArrayAccess⟩
        | new ⟨Type⟩ [ ⟨expr⟩ ]
⟨ArrayAccess⟩ ::= ⟨ident⟩ [ ⟨expr⟩ ]
                | ⟨ArrayAccess⟩ [ ⟨expr⟩ ]
⟨statement⟩ ::= ⟨ArrayAccess⟩ = ⟨expr⟩

```