Bewertungsrichtlinien

Aufgabe	Beschreibung	Punkte
	Klassendiagramm	
	• Zuordnung der Klassen zu den drei Schichten	3
	• Assoziationen, Rollennamen, Multiplizitäten, Benutztpfeile	4
	• Vererbung	3
	• Design der Klassen in UML-Notation	4
	Σ	**14**
	Programm Stammdaten Fussballmanager	
	• Klasse zur Herstellung der Verbindung mit Datenbanksystem	2
	Σ	**2**
	• Klasse mit SQL-Funktionalität (Persistenzschicht)	
	• Datensätze aus Tabelle Spieler auslesen	3
	• Datensätze aus Tabelle Trainer auslesen	3
	• Datensätze aus Tabelle Torwart auslesen	3
	• Datensätze aus Tabelle Mannschaft auslesen	3
	Σ	**12**
	• Abstrakte Klasse für allgemeine Personendaten	
	• Abstrakt	2
	• Attribute	1
	• Methoden	2
	Σ	**5**
	• Klasse Spieler abgeleitet von abstrakter Klasse	
	• Ableitung von abstrakter Klasse	2
	• Attribute	2
	• Methoden	2
	Σ	**6**
	• Klasse für Trainer abgeleitet von abstrakter Klasse	
	• Ableitung von abstrakter Klasse	2
	• Attribute	1
	• Methoden	2
	Σ	**5**

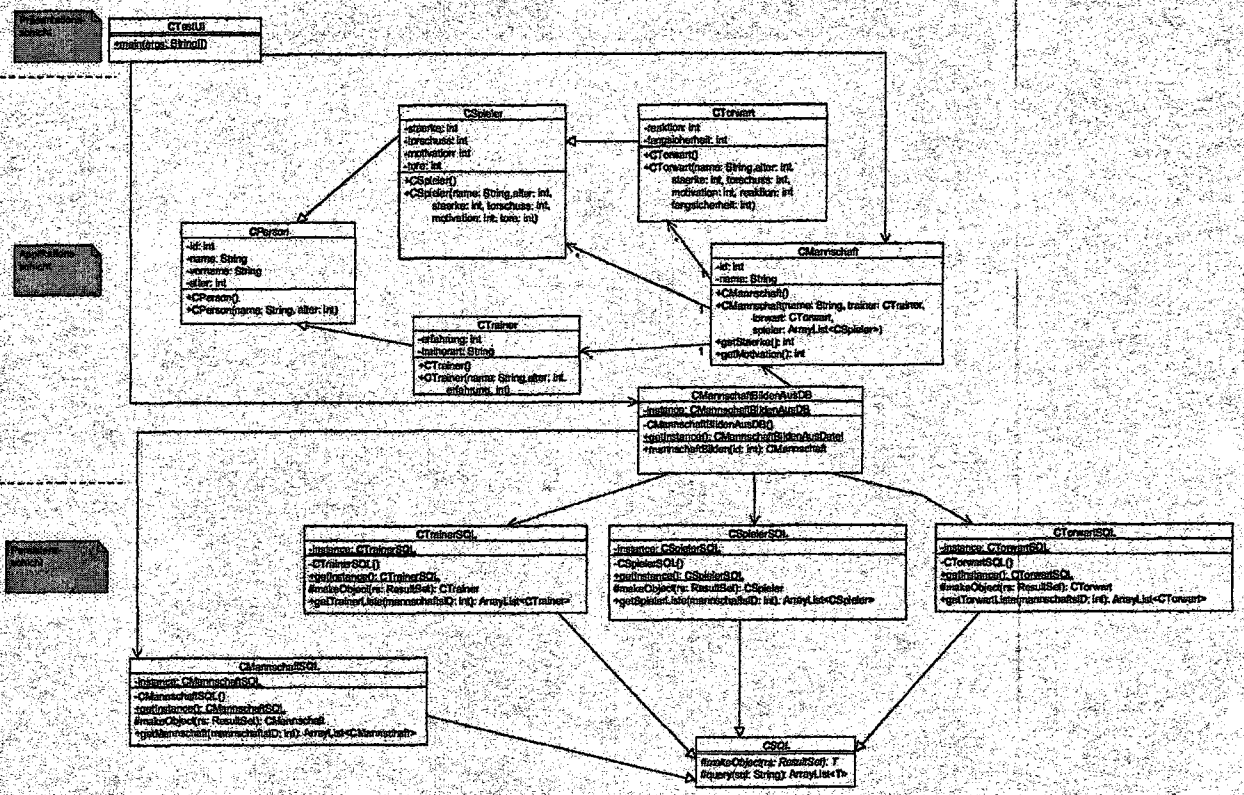
	• Klasse für Torwart abgeleitet von Klasse Spieler	
	• Ableitung von Klasse Spieler	2
	• Attribute	2
	• Methoden	2
	Σ	**6**
	• Mittlerklasse zwischen Applikations- und Persistenzschicht	
	• Attribute	1
	• Methoden	2
	Σ	**3**
	• Testklasse	
	• Selektion der Spieler über die mannschaftsID	1
	• Selektion der Trainer über die mannschaftsID	1
	• Selektion der übrigen Mannschaftsangaben über die mannschaftsID	1
	• Bilden einer Mannschafts-Instanz	1
	• Ausgabe des Mannschaftsnamens (Eigenschaft der Mannschafts-Instanz)	1
	• Anzeige von Name und Vorname der Spieler und des Trainers (ebenfalls über die zuvor gebildete Mannschafts-Instanz).	2
	Σ	**7**
	Σ	**60**
	Vergleich zweier Sortieralgorithmen	**Punkte**
	• Implementierung eines weiteren Sortieralgorithmus	5
	Σ	**5**
	• Effizienzanalyse	
	• Zahl der Vergleiche Bubble Sort	1
	• Zahl der Vergleiche Sortieralgorithmus der Wahl	1
	• Zahl der Bewegungen Bubble Sort	1
	• Zahl der Bewegungen Sortieralgorithmus der Wahl	1
	• Laufzeitverhalten Bubble Sort	1
	• Laufzeitverhalten Sortieralgorithmus der Wahl	1
	Σ	**6**
	• Testklasse	
	• Generierung eines Arrays aus unsortierten Zufallszahlen	3
	• Test Bubble Sort	2

	• Test 2. Sortieralgorithmus	2
	• Ausgabe der Ergebnisse der Effizienzanalyse	2
	Σ	**9**
	Σ	**20**
	Datenbankentwicklung	**Punkte**
	• ER-Diagramm	
	• Die Entität Student	1
	• Die Entität Veranstaltung	1
	• Die Entität Dozent	1
	• Die Entität Modul	1
	• Assoziationen	1,5
	• Kardinalitätstypen der Assoziationen	1,5
	Σ	**7**
	• Relationales Datenmodell	
	• Die Relationen	3
	• Auflösung der 2 Komplex-Komplex-Beziehungen	2
	• Kardinalitätstypen der Assoziationen	2
	• Angabe der Primärschlüsselattribute	3
	• Angabe der Fremdschlüsselattribute	3
	Σ	**13**
	Σ	**20**
	Gesamtpunktzahl	**100**

Lösungvorschläge:

Aufgabe 1

Aufgabe 1.1



Aufgabe1.2

Die Klasse CPerson

```
1 package fussballmanager;
2
3 public abstract class CPerson {
4   // Eigenschaften einer Person
5   private int id;
6   private String name;
7   private String vorname;
8   private int alter;
9
10  // Konstruktoren
11  public CPerson() {}
12  public CPerson (String n, String vn, int a) {
13    name = n;
14    vorname = vn;
15    alter = a;
16  }
17
18  // Funktionen (get und set)
19  public String getName() {
```

```
20      return name;
21    }
22
23    public void setName(String n) {
24      name = n;
25    }
26
27    public String getVorname() {
28        return vorname;
29    }
30
31    public void setVorname(String vn) {
32        vorname = vn;
33    }
34
35    public int getAlter() {
36      return alter;
37    }
38
39    public void setAlter (int a) {
40      alter = a;
41    }
42
43    public int getID() {
44        return id;
45    }
46
47    public void setID(int id) {
48        this.id = id;
49    }
50  }
```

Die Klasse CTrainer

```
 1  package fussballmanager;
 2
 3  public class CTrainer extends CPerson {
 4    // zusätzliche Eigenschaften eines Trainers:
 5    private int erfahrung;
 6    private String trainerart;
 7
 8    // Konstruktoren
 9    public CTrainer() {}
10
11    public CTrainer(String n, String vn, int a, int e, String ta) {
12      super(n, vn, a);
13      erfahrung = e;
14      trainerart = ta;
15    }
16
17    // Funktionen (get und set):
18    public int getErfahrung() {
19      return erfahrung;
20    }
21
22    public void setErfahrung (int e) {
23      erfahrung = e;
24    }
25
26    public String getTrainerart() {
```

```
27        return trainerart;
28    }
29
30    public void setTrainerart(String ta) {
31        trainerart = ta;
32    }
33 }
```

Die Klasse CSpieler

```
 1 package fussballmanager;
 2
 3 public class CSpieler extends CPerson{
 4    // Zusätzliche Eigenschaften eines Spielers:
 5    private int staerke;      // von 1 (schlecht) bis 10 (super)
 6    private int torschuss;    // von 1 (schlecht) bis 10 (super)
 7    private int motivation;   // von 1 (schlecht) bis 10 (super)
 8    private int tore;
 9
10    // Konstruktoren
11    public CSpieler() {}
12
13    public CSpieler(String n, String vn, int a, int s, int ts, int m, int t){
14       super(n, vn, a);
15       staerke     = s;
16       torschuss   = t;
17       motivation  = m;
18       tore        = t;
19    }
20
21    // Funktionen (get und set):
22    public int getStaerke(){
23       return staerke;
24    }
25
26    public void setStaerke(int s){
27       staerke = s;
28    }
29
30    public int getTorschuss(){
31       return torschuss;
32    }
33
34    public void setTorschuss(int t){
35       torschuss = t;
36    }
37
38    public int getMotivation(){
39       return motivation;
40    }
41
42    public void setMotivation(int m){
43       motivation = m;
44    }
45
46    public int getTore(){
47       return tore;
48    }
49
50    public void setTore(int t){
```

```
51        tore = t;
52     }
53 }
```

Die Klasse CTorwart

```
 1 package fussballmanager;
 2
 3 public class CTorwart extends CSpieler{
 4    // Zusätzliche Eigenschaften eines Torwarts:
 5    private int reaktion;
 6    private int fangsicherheit;
 7
 8    // Konstruktoren
 9    public CTorwart() {}
10
11    public CTorwart(String n, String vn, int a, int s, int ts, int m, int t,
12            int r, int f){
13       super(n, vn, a, s, ts, m, t);
14       reaktion    = r;
15       fangsicherheit = f;
16    }
17
18    // Funktionen (get und set):
19    public int getReaktion(){
20       return reaktion;
21    }
22
23    public void setReaktion(int r){
24       reaktion = r;
25    }
26
27    public int getFangsicherheit() {
28        return fangsicherheit;
29    }
30
31    public void setFangsicherheit(int f) {
32        fangsicherheit = f;
33    }
34 }
```

Die Klasse CMannschaft

```
 1 package fussballmanager;
 2
 3 import java.util.ArrayList;
 4
 5 public class CMannschaft{
 6    // Eigenschaften einer Mannschaft:
 7    private int id;
 8    private String name;
 9    private ArrayList<CTrainer> trainer;
10    private ArrayList<CTorwart> torwart;
11    private ArrayList<CSpieler> kader;
12
13    // Konstruktoren
14    public CMannschaft() {}
15
16    public CMannschaft(String n, ArrayList<CTrainer> t,
17            ArrayList<CTorwart> tw, ArrayList<CSpieler> sp){
```

```
18        name     = n;
19        trainer  = t;
20        torwart  = tw;
21        kader    = sp;
22     }
23
24     public CMannschaft(int id, String n, ArrayList<CTrainer> t,
25             ArrayList<CTorwart> tw, ArrayList<CSpieler> sp){
26        this.id       = id;
27        name          = n;
28        trainer       = t;
29        torwart       = tw;
30        kader         = sp;
31     }
32
33     // Funktionen (get und set):
34
35     public int getID() {
36         return id;
37     }
38
39     public void setID(int id) {
40         this.id = id;
41     }
42
43     public String getName(){
44        return name;
45     }
46
47     public void setName(String n){
48        name = n;
49     }
50
51     public ArrayList<CTrainer> getTrainer(){
52        return trainer;
53     }
54
55     public void setTrainer(ArrayList<CTrainer> t){
56        trainer = t;
57     }
58
59     public ArrayList<CTorwart> getTorwart(){
60        return torwart;
61     }
62
63     public void setTorwart(ArrayList<CTorwart> tw){
64        torwart = tw;
65     }
66
67     public ArrayList<CSpieler> getKader(){
68        return kader;
69     }
70
71     public void setSpieler(ArrayList<CSpieler> sp){
72        kader = sp;
73     }
74
75  }
```

Die Klasse CConnectionManager

```
 1 package fussballmanager;
 2
 3 import java.sql.*;
 4
 5 public class CConnectionManager {
 6     private static String DRIVER = "com.mysql.jdbc.Driver";
 7     private static String URL = "jdbc:mysql://localhost/fussballmanager";
 8     private static String USER = "root";
 9     private static String PASSWORD = "";
10     private static Connection con;
11
12     // Verbindung zur DB herstellen
13     public static Connection getConnection() throws ClassNotFoundException,
14         SQLException {
15         if (con == null) {
16             con = DriverManager.getConnection(URL, USER, PASSWORD);
17         }
18         return con;
19     }
20
21     // Verbindung zur DB schließen
22     public static void closeConnection() {
23         try {
24             if (con != null) {
25                 con.close();
26                 con = null;
27             }
28         }
29         catch (SQLException e) {}
30     }
31 }
```

Die Klasse CSQL

```
 1 package fussballmanager;
 2
 3 import java.sql.*;
 4 import java.sql.ResultSet;
 5 import java.util.ArrayList;
 6
 7 public abstract class CSQL<T> {
 8
 9     protected abstract T makeObject(ResultSet rs) throws SQLException;
10
11     protected ArrayList<T> query(String sql) throws ClassNotFoundException,
12             SQLException {
13         ArrayList<T> result = new ArrayList<T>();
14
15         Connection con = null;
16         Statement s = null;
17         ResultSet rs = null;
18         try {
19             con = CConnectionManager.getConnection();
20             s = con.createStatement();
21             rs = s.executeQuery(sql);
22             while (rs.next()) {
23                 result.add(makeObject(rs));
```

```
24              }
25              return result;
26          } finally {
27              if (rs != null) {
28                  rs.close();
29              }
30              if (s != null) {
31                  s.close();
32              }
33          }
34      }
35
36      // SQL-Insert, -Update oder -Delete
37      protected int update(String sql) throws ClassNotFoundException,
38              SQLException {
39          Connection con = null;
40          Statement s = null;
41          int count = 0;
42
43          try {
44              con = CConnectionManager.getConnection();
45              s = con.createStatement();
46              count = s.executeUpdate(sql);
47              return count;
48          } finally {
49              if (s != null) {
50                  s.close();
51              }
52          }
53      }
54  }
```

Die Klasse CTrainerSQL

```
 1  package fussballmanager;
 2
 3  import java.sql.*;
 4  import java.util.ArrayList;
 5
 6  public class CTrainerSQL extends CSQL<CTrainer> {
 7      private static CTrainerSQL instance;
 8
 9      private CTrainerSQL() {
10      }
11
12      public static CTrainerSQL getInstance() {
13          if (instance == null) {
14              instance = new CTrainerSQL();
15          }
16          return instance;
17      }
18
19      // Einschub-Methode für CSQL
20      protected CTrainer makeObject(ResultSet rs) throws SQLException {
21          CTrainer t = new CTrainer();
22          t.setID(rs.getInt("trainerid"));
23          t.setName(rs.getString("name"));
24          t.setVorname(rs.getString("vorname"));
25          t.setAlter(rs.getInt("alter"));
```

```
26          t.setErfahrung(rs.getInt("erfahrung"));
27          t.setTrainerart(rs.getString("trainerart"));
28
29          return t;
30      }
31
32      // Alle Trainer einer Mannschaft holen
33      public ArrayList<CTrainer> getTrainerListe(int mannschaftsID)
34              throws ClassNotFoundException, SQLException {
35          String sql = "select * from trainer "
36                  + "where mannschaftsid = " + mannschaftsID;
37          ArrayList<CTrainer> result = query(sql);
38          return result;
39      }
40  }
```

Die Klasse CSpielerSQL

```
 1  package fussballmanager;
 2
 3  import java.sql.*;
 4  import java.util.*;
 5
 6  public class CSpielerSQL extends CSQL<CSpieler> {
 7
 8      private static CSpielerSQL instance;
 9
10      private CSpielerSQL() {
11      }
12
13      public static CSpielerSQL getInstance() {
14          if (instance == null) {
15              instance = new CSpielerSQL();
16          }
17          return instance;
18      }
19
20      // Einschub-Methode für CSQL
21      protected CSpieler makeObject(ResultSet rs) throws SQLException {
22          CSpieler s = new CSpieler();
23          s.setID(rs.getInt("spielerid"));
24          s.setName(rs.getString("name"));
25          s.setVorname(rs.getString("vorname"));
26          s.setAlter(rs.getInt("alter"));
27          s.setStaerke(rs.getInt("staerke"));
28          s.setMotivation(rs.getInt("motivation"));
29          s.setTorschuss(rs.getInt("torschuss"));
30
31          return s;
32      }
33
34      // Alle Feldspieler einer Mannschaft holen
35      public ArrayList<CSpieler> getSpielerListe(int mannschaftsID)
36              throws ClassNotFoundException, SQLException {
37          String sql = "select * from spieler "
38                  + "where mannschaftsid = " + mannschaftsID;
39          ArrayList<CSpieler> result = query(sql);
40          return result;
41      }
```

```
42 }
```

Die Klasse CMannschaftSQL

```
 1 package fussballmanager;
 2
 3 import java.sql.*;
 4 import java.util.ArrayList;
 5
 6 public class CMannschaftSQL extends CSQL<CMannschaft> {
 7     private static CMannschaftSQL instance;
 8
 9     private CMannschaftSQL() {
10     }
11
12     public static CMannschaftSQL getInstance() {
13         if (instance == null) {
14             instance = new CMannschaftSQL();
15         }
16         return instance;
17     }
18
19     // Einschub-Methode für CSQL
20     protected CMannschaft makeObject(ResultSet rs) throws SQLException {
21         CMannschaft m = new CMannschaft();
22         m.setID(rs.getInt("mannschaftsid"));
23         m.setName(rs.getString("name"));
24
25         return m;
26     }
27
28     //  Mannschaft holen
29     public ArrayList <CMannschaft> getMannschaft(int mannschaftsID)
30             throws ClassNotFoundException, SQLException {
31         String sql = "select * from mannschaft "
32                 + "where mannschaftsid = " + mannschaftsID;
33         ArrayList<CMannschaft> result = query(sql);
34         return result;
35     }
36 }
```

Die Klasse MannschaftBildenAusDB

```
 1 package fussballmanager;
 2
 3 import java.sql.SQLException;
 4 import java.util.ArrayList;
 5
 6 public class CMannschaftBildenAusDB {
 7     private static CMannschaftBildenAusDB instance;
 8
 9     private CMannschaftBildenAusDB() {
10     }
11
12     public static CMannschaftBildenAusDB getInstance() {
13         if (instance == null) {
14             instance = new CMannschaftBildenAusDB();
15         }
16         return instance;
17     }
```

```
18
19      public CMannschaft mannschaftBilden(int id)
20              throws ClassNotFoundException, SQLException  {
21          ArrayList<CTrainer> trainer = new ArrayList<CTrainer>();
22          ArrayList<CTorwart> torwart = new ArrayList<CTorwart>();
23          CMannschaft mannschaft = new CMannschaft();
24          ArrayList<CSpieler> feldspieler = new ArrayList<CSpieler>();
25          mannschaft = CMannschaftSQL.getInstance().getMannschaft(id).get(0);
26          trainer = CTrainerSQL.getInstance().getTrainerListe(id);
27          torwart = CTorwartSQL.getInstance().getTorwartListe(id);
28          feldspieler = CSpielerSQL.getInstance().getSpielerListe(id);
29          mannschaft.setSpieler(feldspieler);
30          mannschaft.setTorwart(torwart);
31          mannschaft.setTrainer(trainer);
32          return mannschaft;
33      }
34  }
```

Die Klasse CFussballManagerUI

```
 1  package fussballmanager;
 2
 3  import java.io.FileNotFoundException;
 4  import java.io.IOException;
 5  import java.sql.SQLException;
 6  import java.util.ArrayList;
 7
 8  public class CFussballManagerUI{
 9
10     public static void main(String[] args)
11             throws FileNotFoundException, IOException,
12             ClassNotFoundException, SQLException{
13
14         CMannschaft m =
15                 CMannschaftBildenAusDB.getInstance().mannschaftBilden(1);
16         ArrayList<CTrainer> tr = m.getTrainer();
17         ArrayList<CTorwart> tw = m.getTorwart();
18         ArrayList<CSpieler> sp = m.getKader();
19
20         System.out.println("Mannschaft: " + m.getName());
21         for (CTrainer t : tr) {
22             System.out.println("Trainer: " + t.getName() + ", " +
23                     t.getVorname());
24         }
25
26         for (CTorwart t : tw) {
27             System.out.println("Torwart: " + t.getName() + ", " +
28                     t.getVorname());
29         }
30
31         for (CSpieler s : sp) {
32             System.out.println("Spieler: " + s.getName() + ", " +
33                     s.getVorname());
34         }
35     }
36
37  }
```

Aufgabe 2

Die Klasse CSort

Alternativ sind entweder
- Insertion Sort oder
- Selection Sort oder
- Quicksort

zu implementieren.

```
 1 package sortieralgorithmen;
 2
 3 public class CSort {
 4     private int vergleiche = 0;
 5     private int bewegungen = 0;
 6
 7     public int getVergleiche() {
 8         return vergleiche;
 9     }
10
11     public int getBewegungen() {
12         return bewegungen;
13     }
14
15      public void insertionSort(int[] arr) {
16         int i, j, newValue;
17         vergleiche = 0;
18         bewegungen = 0;
19         for (i = 1; i < arr.length; i++) {
20             newValue = arr[i];
21             // bewegungen++;
22             j = i;
23             while (j > 0 && arr[j - 1] > newValue) {
24                 vergleiche++;
25                 bewegungen++;
26                 arr[j] = arr[j - 1];
27                 j--;
28             }
29             arr[j] = newValue;
30         }
31     }
32
33     public void bubbleSort(int[] arr) {
34         boolean swapped = true;
35         int j = 0;
36         int tmp;
37         vergleiche = 0;
38         bewegungen = 0;
39
40         while (swapped) {
41             swapped = false;
42             j++;
43             for (int i = 0; i < arr.length - j; i++) {
44                 vergleiche++;
45                 if (arr[i] > arr[i + 1]) {
46                     bewegungen++;
47                     tmp = arr[i];
```

```
48                      arr[i] = arr[i + 1];
49                      arr[i + 1] = tmp;
50                      swapped = true;
51                  }
52              }
53          }
54      }
55
56      public void selectionSort(int[] arr) {
57          vergleiche = 0;
58          bewegungen = 0;
59          int i, j, minIndex, tmp;
60          int n = arr.length;
61
62          for (i = 0; i < n - 1; i++) {
63              minIndex = i;
64              for (j = i + 1; j < n; j++) {
65                  vergleiche++;
66                  if (arr[j] < arr[minIndex]) {
67                      minIndex = j;
68                  }
69              }
70              vergleiche++;
71              if (minIndex != i) {
72                  bewegungen++;
73                  tmp = arr[i];
74                  arr[i] = arr[minIndex];
75                  arr[minIndex] = tmp;
76              }
77          }
78      }
79
80      public int partition(int arr[], int left, int right) {
81
82          int i = left, j = right;
83          int tmp;
84          int pivot = arr[(left + right) / 2];
85
86          while (i <= j) {
87              vergleiche++;
88              while (arr[i] < pivot) {
89                  i++;
90              }
91
92              while (arr[j] > pivot) {
93                  j--;
94              }
95
96              if (i <= j) {
97                  bewegungen++;
98                  tmp = arr[i];
99                  arr[i] = arr[j];
100                 arr[j] = tmp;
101                 i++;
102                 j--;
103             }
104         };
105         return i;
106     }
107
108     public void quickSort(int arr[], int left, int right) {
```

```
109         int index = partition(arr, left, right);
110
111         if (left < index - 1) {
112             quickSort(arr, left, index - 1);
113         }
114
115         if (index < right) {
116             quickSort(arr, index, right);
117         }
118     }
119 }
```

Die Klasse CSortTest

Ein Test ist durchzuführen mit Bubble Sort und alternativ

- Selection Sort oder
- Insertion Sort oder
- Quicksort

```
 1 package sortieralgorithmen;
 2
 3 import java.util.Random;
 4
 5 // Datei: SortTest.java
 6 //
 7 // Testen der Sortierfunktionen:
 8 //   - Selection Sort
 9 //   - Insertion Sort
10 //   - Bubble Sort
11 //   - Quicksort
12 //
13 // Arbeitsweise: Es wird ein unsortiertes Feld
14 // erzeugt (mit Hilfe von Zufallszahlen).
15 // Die Sortierfunktion sortiert das Feld und
16 // es wird auf korrekte Sortierung geprueft.
17 //
18 public class CSortTest {
19
20     //
21     // init erzeugt ein zufaelliges Feld
22     // der Laenge n. Dabei wird die Funktion
23     // Math.random benutzt.
24     //
25     public static int[] init(int n) {
26         int a[] = new int[n];
27         Random rd = new Random();
28
29         for (int i = 0; i < n; i++) {
30             a[i] = rd.nextInt(100000);
31         }
32         return a;
33     }
34
35     //
36     // test prueft die korrekte Reihenfolge
37     // in einem sortierten Feld.
38     //
39     public static void test(int a[]) {
```

```
40
41          for (int i = 1; i < a.length; i++) {
42              if (a[i - 1] > a[i]) {
43                  System.out.println("Falsche Reihenfolge!");
44                  return;
45              }
46          }
47          System.out.println("ok.");
48      }
49
50      // Ausgabe zu Testzwecken
51      public static void ausgabe(int a[]) {
52          for (int i = 1; i <= a.length; i++) {
53              if (i % 20 == 0 && i != 0) {
54                  System.out.printf("%4d", a[i - 1]);
55                  System.out.println();
56              } else {
57                  System.out.printf("%4d", a[i - 1]);
58              }
59          }
60      }
61
62      //
63      // main ist der Einstiegspunkt des Programms.
64      // Es werden alle implementierten Sortierverfahren
65      // getestet.
66      //
67      public static void main(String args[]) {
68
69          //
70          // n ist die Laenge des zu testenden Feldes
71          //
72          int n = 10000;
73          CSort sort = new CSort();
74
75          // Test Selection Sort
76          int a[] = init(n);
77          int arrGesichert[] = a;
78          CStoppuhr t = new CStoppuhr();
79          t.starte();
80          sort.selectionSort(a);
81          t.stoppe();
82          // ausgabe(a);
83           System.out.println("Zeit Selection Sort: " + t.lies() + " ms" +
84                  " Vergleiche: " + sort.getVergleiche() +
85                  " Bewegungen: " + sort.getBewegungen());
86
87          // Test Insertion Sort
88          a = init(n);
89          t = new CStoppuhr();
90          t.starte();
91          sort.insertionSort(a);
92          t.stoppe();
93          // ausgabe(a);
94           System.out.println("Zeit Insertion Sort: " + t.lies() + " ms" +
95                  " Vergleiche: " + sort.getVergleiche() +
96                  " Bewegungen: " + sort.getBewegungen());
97
98          // Test Bubble Sort
99          a = init(n);
100         t = new CStoppuhr();
```

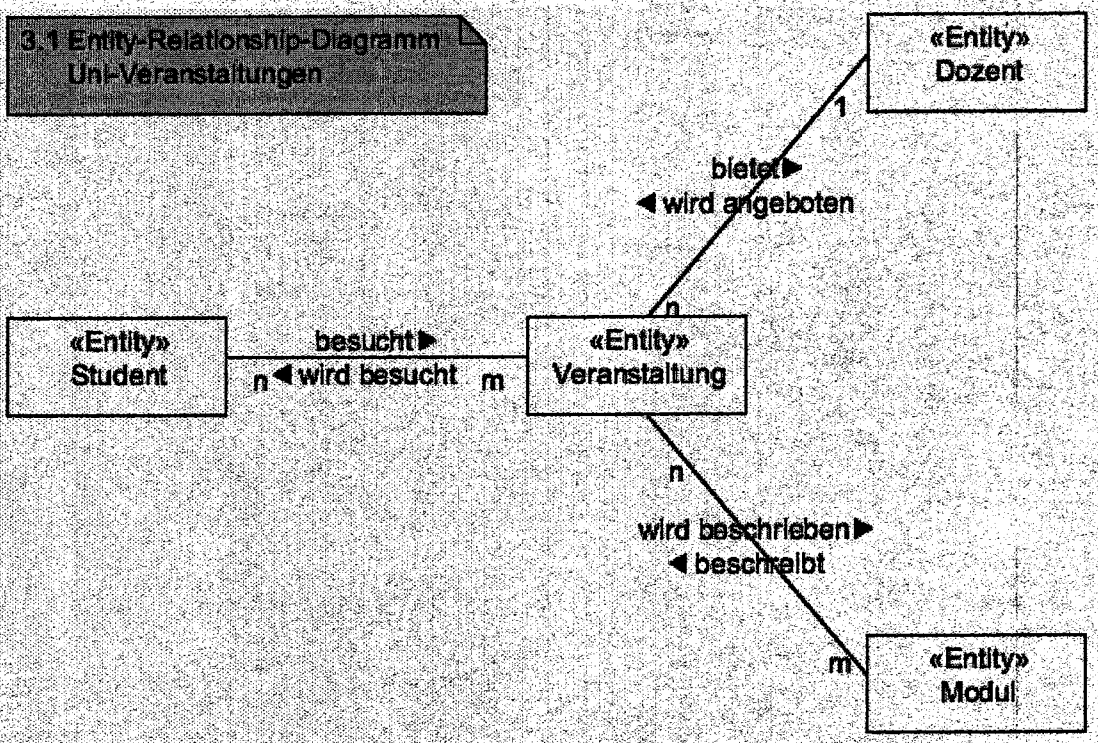
```
101         t.starte();
102         sort.bubbleSort(a);
103         t.stoppe();
104         // ausgabe(a);
105          System.out.println("Zeit Bubble Sort: " + t.lies() + " ms" +
106                  " Vergleiche: " + sort.getVergleiche() +
107                  " Bewegungen: " + sort.getBewegungen());
108
109         // Test Quicksort
110         CSort qsort = new CSort();
111         a = init(n);
112         t = new CStoppuhr();
113         t.starte();
114         qsort.quickSort(a, 0, n - 1);
115         t.stoppe();
116         // ausgabe(a);
117          System.out.println("Zeit Quicksort optimiert: " + t.lies() + " ms" +
118                  " Vergleiche: " + qsort.getVergleiche() +
119                  " Bewegungen: " + qsort.getBewegungen());
120     }
121 }
```

Hilfklasse CStoppuhr() für Zeitmessung

```
 1 package sortieralgorithmen;
 2
 3 public class CStoppuhr
 4 {
 5   private long millis;
 6
 7   public void starte()
 8   {
 9     millis = System.currentTimeMillis();
10   }
11
12   public void stoppe()
13   {
14     millis = System.currentTimeMillis() - millis;
15   }
16
17   public long lies()
18   {
19     return millis;
20   }
21 } // class Stoppuhr
```

Aufgabe 3

Entity-Relationship-Diagramm



Relationales Datenbankmodell