Saarland
Ministerium
für Bildung und Kultur

## Lösungshinweise: Nur für die Hand der Lehrperson

# Schriftliche Abiturprüfung 2013

| | |
|---|---|
| Fach: | **Informatiksysteme (Fachrichtung Technik)** |
| Kurstyp: | E-Kurs |
| Bearbeitungszeit: | 5 Zeitstunden |
| Hilfsmittel: | Java-IDE Netbeans, Umlet (Tool zum Zeichnen des Klassendiagramms), XAMPP (MySQL-DBMS, SQL-Client phpmyadmin ) |
| Seitenzahl: | Die Lösungshinweise umfassen mit Deckblatt, Bewertungsrichtlinien, Lösungsvorschlägen und Bewertungstabelle 17 Seiten. |

# Bewertungsrichtlinien

| Aufgabe | Beschreibung | Punkte |
|---|---|---|
| **1** | **Fragen zur Anwendungsentwicklung** | |
| | **a)** | |
| | • Konstruktor | 4 |
| | • Definition Klassenattribut | 2 |
| | • Definition Klassenmethoden | 2 |
| | b) | |
| | • Vererbung | 3 |
| | • Ableitung von Arzt und Patient von der Klasse Person | 3 |
| | • Sichtbarkeit | 5 |
| | c) | |
| | • Bubblesort | 6 |
| | | |
| | **Σ** | **25** |
| **3** | **Objektorientierte Programmentwicklung: Praktische Aufgabe** | |
| | • Klasse für Patient | |
| | ◦ Attribute | 2 |
| | ◦ Getter-, Setter-Methoden | 2 |
| | ◦ Referenz : Liste mit Behandlungen | 2 |
| | **Σ** | **6** |
| | • Klasse für Behandlungen | |
| | ◦ Attribute | 2 |
| | ◦ Getter- , Setter Methoden mit Anpassung Datum auch im Konstruktor | 6 |
| | ◦ Referenz auf den Patienten mit Konstruktor herstellen | 4 |
| | ◦ getinfo() | 2 |
| | **Σ** | **14** |
| | • Containerklasse CManager | |
| | ◦ Liste für Patienten | 1 |
| | ◦ Singelton | 4 |
| | ◦ Patient erzeugen | 2 |
| | ◦ Patienten anzeigen | 3 |

| | | |
|---|---|---|
| | ◦ Behandlung erzeugen | 2 |
| | ◦ Patient suchen | 7 |
| | ◦ Behandlungen suchen, Schlüssel Datum | 7 |
| | **Σ** | **26** |
| | | |
| | • Testklasse | |
| | ◦ Managerobjekt anlegen | 1 |
| | ◦ Patienten erzeugen und anzeigen | 2 |
| | ◦ Behandlungen erzeugen und zuordnen | 2 |
| | ◦ Suche nach Patient | 1 |
| | ◦ Suche nach Behandlung | 1 |
| | **Σ** | **7** |
| **4** | **Datenbankentwicklung** | |
| | a) ER-Diagramm erstellen | |
| | • Entitäten und ihre Zuordnung | 5 |
| | • Kardinalitäten | 2 |
| | • Schlüssel (Primär- und Fremdschlüssel) | 3 |
| | • Begründung | 2 |
| | b) SQL | 10 |
| | | |
| | **Σ** | **22** |
| | | |
| | **Gesamtpunktzahl** | **100** |

# Lösungsvorschläge:

## *Aufgabe 1*

a) Grundlagen

i. Der Konstruktor ist eine Methode, die bei Erzeugung bzw. Instanzbildung von Objekten aufgerufen wird. Neben der Zuweisung von Speicher (Je nach Programmiersprache) ist seine Hauptaufgabe die Initialisierung des Objektes. Der Standardkonstruktor ist eine parameterlose Methode, die auch ohne explizite Definition existiert. Der Konstruktor kann aber auch selbst definiert werden und über die Parameterliste Werte zur Initialisierung der Objekte einlesen. So kann die Methode, die die gleiche Bezeichnung wie die zugehörige Klasse besitzt überladen werden. Überladene Methoden haben den gleichen Namen, unterscheiden sich jedoch in Anzahl und Typ der Variablen in der Parameterliste.

ii. Ein Klassenattribut beschreibt die Eigenschaften einer Klasse, im Gegensatz zu den üblichen Objektattributen, die Eigenschaften eines einzelnen Objektes beschreiben. Der Attributwert dieses Klassenattributs ist für alle Objekte einer Klasse gleich. Klassenattribute existieren auch, wenn es zu einer Klasse keine Objekte gibt. Sie werden mit dem Schlüsselwort ***static*** gekennzeichnet.

iii. Klassenmethoden sind Operationen, die der jeweiligen Klasse zugeordnet sind und nicht auf ein einzelnes Objekt angewendet werden. Sie manipulieren in der Regel Klassenattribute der eigenen Klasse, ohne die Beteiligung eines Objektes, an das sie gebunden sein müssen.

.

b) Paradigmen

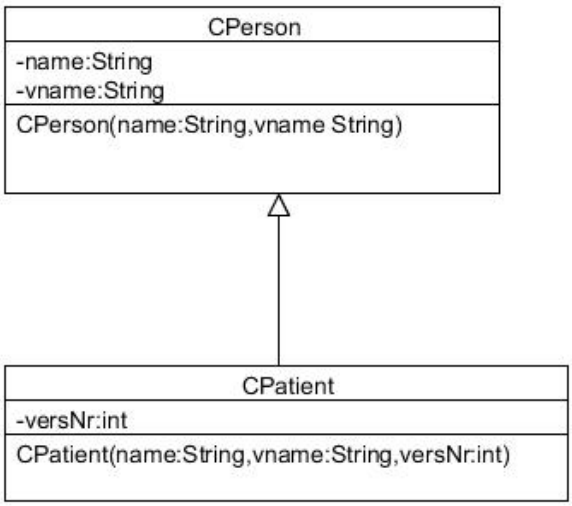
i. Vererbung

Grundlage der Vererbung ist die Überlegung der Generalisierung (Verallgemeinerung) und Spezialisierung. Verallgemeinerbare Eigenschaften werden in sogenannten Oberklassen zusammengefasst und an spezialisierte Klassen, sogenannte Unterklassen "vererbt", indem die Unterklasse nach bestimmten Regeln von der Oberklasse abgeleitet wird. Diese hat alle Eigenschaften der Oberklasse und bekommt neue, die Spezialisierung ausmachende Eigenschaften hinzu. So können sich Vererbungshierarchien bilden.

ii. Beispiel

```
public class CPerson {
    private String name;
    private String vname;
```

```
    public CPerson(String name, String vname) {
        this.name = name;
        this.vname = vname;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getVname() {
        return vname;
    }
    public void setVname(String vname) {
        this.vname = vname;
    }
    }

public class CPatient extends CPerson{
    //zusätzliches Attribut
    private int versNr;
    //Konstruktor

    public CPatient(String name, String vname, int versNr) {
        super(name, vname);
        this.versNr = versNr;
    }

    public int getVersNr() {
        return versNr;
    }

    public void setVersNr(int versNr) {
        this.versNr = versNr;
    }

}
```




iii. Sichtbarkeit

Will man in einer ArrayList von Objekten der von Cperson alle abgeleiteten Objekte wie Patienten, Ärzte oder sonstige verwalten, wird durch das Hinzufügen zur Arraylist eine Referenz vom Typ CPerson auf das abgeleitete und eingefügte Element erzeugt. Somit sind über die Liste die Attribute der abgeleiteten Objekte nicht direkt sichtbar. Es ist eine explizite Typenkonvertierung nötig.

Beispiel:

```
static ArrayList <CPerson> alle=new ArrayList<CPerson>();


    public static void main(String[] args) {
        // TODO code application logic here
        CPatient neuerPatient= new CPatient("Muster", "Fritz",
234561);
        alle.add(neuerPatient);
/*Der Rückgabewert der Cast-Operation wird der CPatient-
Referenz hilf zugewiesen. Diese sieht wieder alle Attribute*/
        CPatient hilf= (CPatient)alle.get(0);
        System.out.printf("%nName: %s, VersNr.:
%d",alle.get(0).getName(), hilf.getVersNr());
    }
}
```

c) Aufsteigende Sortierung:

In einer Liste von n Elementen

1 Man vergleicht beginnend mit dem 1. Element ein Element mit dem folgenden Element.

2 Ist es größer, tauscht man die Plätze

3 Ist es kleiner vergleiche das größere mit dem nächsten Element

4 Fahre fort bis Du beim n-ten Element angelangt bist.

5 Nun ist der Größte Wert an der n-ten Position.

6 Man beginnt wieder bei Punkt 1und führt den Wechsel von Vergleich und Tausch bis zum Platz n-1durch, dann bis zum Platz n-2und so weiter, bis man am Anfang angelangt ist

Beispiel: Zahlenfolge 12, 78, 34, 5,19, 36

1. Durchgang:12, 34, 5,19, 36, 78

2. Durchgang:12, 5, 19, 34, 36, 78

3. Durchgang: 5, 12, 19, 34, 36,78

Folge ist sortiert

## *Aufgabe 2*

**CPatient.java**

```
package pv;
import java.util.ArrayList;

public class CPatient {
   private int pid;
   private String name;
   private String vorname;
   private int vers;
   private ArrayList <CBehandlung> meineBehandlungen;

   public CPatient()   {
      //Instanz von meineBehandlungen bilden
      meineBehandlungen=new ArrayList<CBehandlung>();   }

   public CPatient(int pid, String name, String vorname, int vers )   {
     this.pid = pid;
     this.name=name;
     this.vorname=vorname;
     this.vers=vers;
     meineBehandlungen=new ArrayList<CBehandlung>();
```

```
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getVorname() {
        return vorname;
    }

    public void setVorname(String vorname) {
        this.vorname = vorname;
    }

    public ArrayList<CBehandlung> getMeineBehandlungen() {
        return meineBehandlungen;
    }

    public void setMeineBehandlungen(ArrayList<CBehandlung> meineBehandlungen) {
        this.meineBehandlungen = meineBehandlungen;
    }

    public int getPid() {
        return pid;
    }

    public void setPid(int pid) {
        this.pid = pid;
    }

    public String getVers() {
        String aus="";
        if(vers==1){
         aus= "privat" ;
```

```
        }else
        {
            aus="gesetzlich";
        }
        return aus;
    }

    public void setVers(int vers) {
        this.vers = vers;
    }
}
```

**CBehandlung.java**

```
package pv;
import java.text.DateFormat;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CBehandlung {
    private int bid;
    private String betreff;
    private Calendar datum;
    private String diagnose;

    public CBehandlung( CPatient patient, int bid,String betreff, String datum, String diagnose){
        //Auf den Behandlungskontainer des Patienten zugreifen
        ArrayList meinContainer=patient.getMeineBehandlungen();
        //dieses Behandlungsobjekt hinzufügen
        meinContainer.add(this);
        this.bid=bid;
        this.betreff=betreff;
        //Zeit vom String in Calendar wandeln
```

```
    Calendar d =Calendar.getInstance();
     DateFormat myFormat=DateFormat.getDateInstance(DateFormat.SHORT,Locale.GERMAN);
     Date date = null;
     try {
        date=myFormat.parse(datum);
     } catch (ParseException ex) {
        Logger.getLogger(CBehandlung.class.getName()).log(Level.SEVERE, null, ex);
     }
     d.setTime(date);
     this.datum=d;
     //diagnose zuweisen
     this.diagnose=diagnose;
  }

  public int getBid() {
     return bid;
  }

  public void setBid(int bid) {
     this.bid = bid;
  }

  public String getDiagnose() {
     return diagnose;
  }

  public void setDiagnose(String diagnose) {
     this.diagnose = diagnose;
  }

  public String getBetreff() {
     return betreff;
  }

  public void setBetreff(String betreff) {
     this.betreff = betreff;
  }
```

```
    public Calendar getDatum() {
        return datum;
    }
    public String getDatumStr(){
        String datumString;
        DateFormat meinAusgabeFormat= DateFormat.getDateInstance(DateFormat.MEDIUM);
        datumString=meinAusgabeFormat.format(datum.getTime());
        return datumString;
    }

    public void setDatum(String datum) {//tt.mm.jjjj
        Calendar d=Calendar.getInstance();
        DateFormat myFormat=DateFormat.getDateInstance(DateFormat.FULL);
        myFormat.setTimeZone(TimeZone.getDefault());
        Date date = null;
        try {
            date=myFormat.parse(datum);
        } catch (ParseException ex) {
            Logger.getLogger(CBehandlung.class.getName()).log(Level.SEVERE, null, ex);
        }
        d.setTime(date);
        this.datum=d;
    }

    public String getInfo(){
        String aus=String.format("%n BID: %d %n Betreff: %s %n Datum: %s %n Diagnose: %s "
                , bid,betreff,getDatumStr(), diagnose);
        return aus;
    }
        }
```

**CManager.java**

```
package pv;

import java.util.ArrayList;



public class CManager {
```

```
    private static CManager manager = null;//Singlton
    private ArrayList <CPatient> meinePatienten =new ArrayList <CPatient>();
    //Wegen Singlton-Muster Konstruktor private
    private CManager() {
    }
    public static CManager getManager() {

        if (manager == null) {
            manager = new CManager();
        }else{
        System.out.println("%nManager-Objekt wurde bereits erzeugt!");

        }
        return manager;
    }
//Getter- und Setter-Methoden

    public ArrayList<CPatient> getMeinePatienten() {
        return meinePatienten;
    }

    public void setMeinePatienten(ArrayList<CPatient> meinePatienten) {
        this.meinePatienten = meinePatienten;
    }
//Funktionen allgemein
    public void neuerPatient(int pid,
            String name, String vorname, int vers){
        CPatient neu = new CPatient(pid, name, vorname,vers);
        meinePatienten.add(neu);
    }

    public void info(CPatient p){
        System.out.printf("%n%d, %s, %s, Vers: %s"
                ,p.getPid(), p.getName(),p.getVorname(),p.getVers());
    }

    public void allePatienten(){
        System.out.printf("%nFolgende Patienten sind gespeichert:");
```

```
        for(CPatient p: meinePatienten){


            System.out.printf("%nPID: %d, Name: %s, %s, Vers: %s"
               ,p.getPid(), p.getName(),p.getVorname(),p.getVers());
        }
        System.out.println();
    }
    //Behandlung hinzufügen
    public void neueBehandlung(CPatient p,int bid, String betreff, String datum, String diagnose){
        CBehandlung neu= new CBehandlung( p, bid,betreff, datum, diagnose);


    }


    //nach einem Patienten suchen , Schlüssel: Name
    public void suchePatient(String name){
        int i=0;
        for(CPatient p: meinePatienten){
            if(name.equals(p.getName())){
            i++;
              if(i==1){
                 System.out.printf("%nDaten von Patient %s wurden gefunden:"
                       ,p.getName());
                 }
            info(p);
                 ArrayList <CBehandlung> container=p.getMeineBehandlungen();
                 for(CBehandlung b:container ){
                    System.out.printf("%n-------%nBID: %d %nBetreff: %s %nDatum: %s %nDiagnose: %s
"
                          ,b.getBid(),b.getBetreff(),b.getDatumStr(),b.getDiagnose());
                 }


            }
        }
        if(i==0){
        System.out.println("%n Unter dem Suchbegriff wurden keine Patienten gefunden,"
              + " überprüfen Sie gegebenfalls die Schreibweise");
        }
    }
    //nach Behandlungen suchendie am patient x erfolgt sind
```

```
   //Schlüssel->Datum
   public void sucheBehandlung(String datum){
       //Alternative String in Calender umformen
      /*Calendar einDatum= Calendar.getInstance();
      DateFormat meinEingabeFormat=DateFormat.getDateInstance(DateFormat.SHORT,
Locale.GERMAN);
      Date eingabeDatum=null;
      eingabeDatum=meinEingabeFormat.parse(datum);
      einDatum.setTime(eingabeDatum);*/

      ArrayList <CBehandlung> hilf ;
      for(CPatient p:meinePatienten){
         hilf=p.getMeineBehandlungen();

         for(CBehandlung b:hilf ){
            if(b.getDatumStr().equals(datum)){
            System.out.printf("%nPID: %d, Name: %s", p.getPid(),p.getName());
            System.out.println(b.getInfo());
         }
         }
      }
   }
}
```

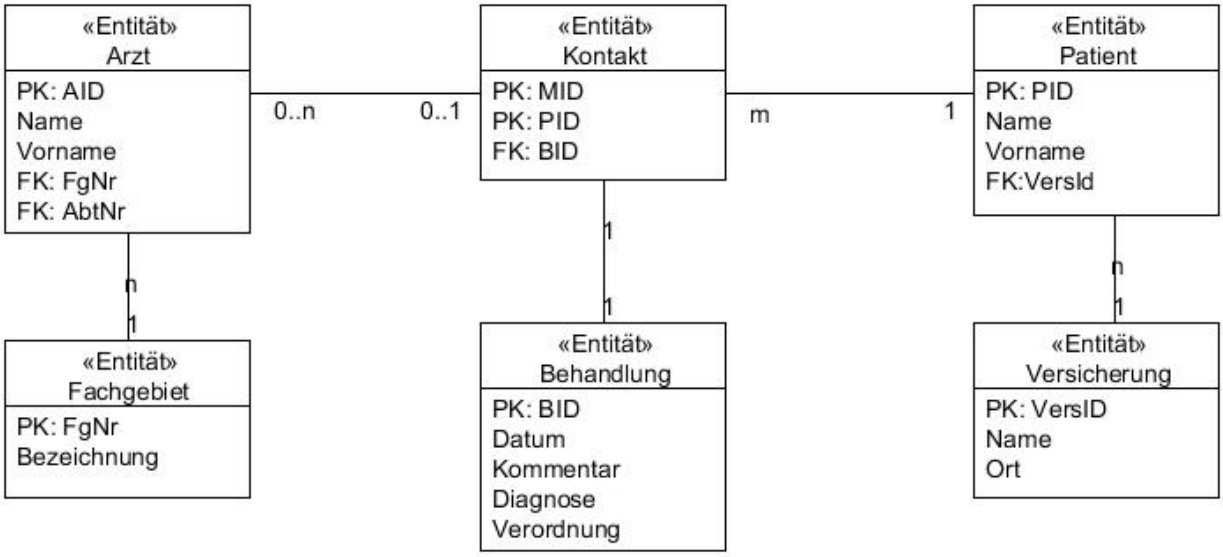
**CPVUI.java**

```
package pv;



public class CPVUI {
   public static void main(String[] args) {

       CManager manager=CManager.getManager();//Managerobjekt anlegen
       //Drei Kontakte anlegen
       manager.neuerPatient(12314, "Meier", "Kurt", 1);
       manager.neuerPatient(12794, "Müller", "Egon", 2);
       manager.neuerPatient(12973, "Schmitt", "Fritz",2);
       //Alle Kontakte ausgeben
       manager.allePatienten();
       //Behandlungen für Patient 1 anlegen
```

```
manager.neueBehandlung(manager.getMeinePatienten().get(0),134,"Herzbeschwerden","24.06.2003"
,"s6w4" );
        manager.neueBehandlung(manager.getMeinePatienten().get(0),234,"Herz,
EKG","30.06.2003","s6w4" );
        manager.neueBehandlung(manager.getMeinePatienten().get(0),357,"Herz,
Kontrolle","12.07.2003","s6w4" );
        manager.neueBehandlung(manager.getMeinePatienten().get(0),435,"Belastungs-
EKG","19.07.2004","s6w4" );

manager.neueBehandlung(manager.getMeinePatienten().get(1),635,"Lungenfunktion","19.07.2004","l2
3z" );
        manager.neueBehandlung(manager.getMeinePatienten().get(1),725,"Kontrolle
LFT","24.10.2004","l23z" );

manager.neueBehandlung(manager.getMeinePatienten().get(2),934,"Erkältung","17.04.2005","erk2" );
        //Alle Behandlungen eines Patienten
        System.out.printf("%n---------Patient suchen----------");
        manager.suchePatient("Meier");
        //nach einer Behandlung suchen
        System.out.printf("%n---------Behandlung suchen----------");
        manager.sucheBehandlung("19.07.2004");

    }
}
```

## *Aufgabe 3*

a) ER-Diagramm



PK: Primärschlüssel

FK: Fremdschlüssel

Beim Entwurf ist darauf zu achten, dass Einfügeanomalien und Redundanzen zu vermeiden sind. Einfügeanomalie entsteht, wenn ein Wert für ein Attribut nicht eindeutig ist. Dies kann bei Bezeichnungen für die Fachgebiete oder der Versicherung leicht geschehen. Zudem ändern sich die Bezeichnungen oder müssen erweitert werden. Dem wird durch die eigene Entität Fachgebiet oder Versicherung Rechnung getragen.

b) SQL

i. select * from mitarbeiter where Name like 'm%';

ii. select PID, Bezeichnung, Name from Projekt, Auftraggeber where Projekt.AID =Aufgtraggeber.AID;

iii. select Name, Vorname, Kosten from Mitarbeiter, Bearbeitung where (Mitarbeiter.MID = Bearbeitung.MID) and Bearbeitung.Kosten >150000;

iv. select DISTINCT (Auftraggeber.Name) from Auftraggeber, Bearbeitung, Projekt where (Bearbeitung.MID = '134') and (Bearbeitung.PID = Projekt.PID) and (Projekt.AID = Auftraggeber.AID);

v. select Bezeichnung, sum(Bearbeitung.Kosten) from Projekt, Bearbeitung where (Projekt.PID = Bearbeitung.PID) group by Bezeichnung having Bezeichnung like 'Pavillon 4';

## Bewertungstabelle

| Prozent der maximal erreichbaren Roh-punktzahl | Note | Punktzahl |
|---|---|---|
| ab 97% bis 100% der max. Punktzahl | sehr gut | 15 P |
| ab 93% bis weniger als 97% | | 14 P |
| ab 90% bis weniger als 93% | | 13 P |
| ab 85% bis weniger als 90% | gut | 12 P |
| ab 80% bis weniger als 85% | | 11 P |
| ab 75% bis weniger als 80% | | 10 P |
| ab 70% bis weniger als 75% | befriedigend | 09 P |
| ab 65% bis weniger als 70% | | 08 P |
| ab 60% bis weniger als 65% | | 07 P |
| ab 55% bis weniger als 60% | ausreichend | 06 P |
| ab 50% bis weniger als 55% | | 05 P |
| ab 45% bis weniger als 50% | | 04 P |
| ab 38% bis weniger als 45% | mangelhaft | 03 P |
| ab 32% bis weniger als 38% | | 02 P |
| ab 25% bis weniger als 32% | | 01 P |
| weniger als 25% der max. Punktzahl | ungenügend | 00 P |

**- Ende der Lösungshinweise -**