I Erläuterungen

Aufgabenart
materialgebundene Aufgabenstellung

Voraussetzungen gemäß Lehrplan und Erlass „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/schwerpunktbezogene Fächer)“ in der für den Abiturjahrgang geltenden Fassung

Implementierung
Die Implementierung soll grundsätzlich in einer objektorientierten Programmiersprache erfolgen. Die Musterlösungen sind in die Programmiersprache zu übertragen, die an der jeweiligen Schule verwendet wird.

Entity-Relationship-Modell
Die Notation für Entity-Relationship-Diagramme hat sich in den zurückliegenden Jahren stark verändert. Neben der ursprünglichen Darstellungsform, die auf P. P. Chen (1976) zurückgeht, unterstützen Softwaretools vielfältige Formen, bis hin zur Krähenfuß-Notation. Aus didaktischen Gründen wird die Notation nach Chen verwendet, die übersichtlich und einfach auch von Hand zu erstellen ist. Die Entitätsmengen werden durch Substantive, die Beziehungen durch Verben beschrieben und die Kardinalitäten in Min-Max-Schreibweise angegeben.

Relationales Modell
Die Primärschlüssel sind in den Musterlösungen unterstrichen, Fremdschlüssel mit einer Raute („#“) versehen.

SQL-Syntax
Die verwendete SQL-Syntax orientiert sich an dem Standard SQL-92 (SQL2). Abweichungen aufgrund der im Unterricht verwendeten Datenbanksysteme sind möglich. Abweichende Lösungen aufgrund anderer Tabellen aus vorangegangenen Aufgabenteilen sind entsprechend zu bewerten.

Aufgabe 1: Ein vorliegendes ER-Diagramm ist in das relationale Modell zu überführen. Auf der Grundlage des relationalen Modells sind SQL-Statements zur Datenmanipulation und -abfrage zu entwickeln. Am Beispiel einer gegebenen Tabelle sind mögliche Anomalien zu beschreiben und es soll eine Normalisierung durchgeführt werden. Darüber hinaus ist zu weiteren Anforderungen an die Datenbank ein entsprechendes ER-Modell zu entwickeln und als ER-Diagramm darzustellen.
Die Aufgabe bezieht sich auf den Kurs „Datenbanken“ aus Q3. Von den verbindlichen Inhalten des Lehrplans werden zur Lösung der Aufgabe benötigt: die Modellierung von Mini-Welten mit ER-Diagrammen, die Überführung eines ER-Diagramms in das Relationenmodell, Datenmanipulation und Abfragen mittels SQL-Anweisungen sowie die Normalisierung von Tabellen.

Aufgabe 2: Eine Vererbungshierarchie ist zu erläutern und auf Grundlage eines UML-Klassenddiagramms zu implementieren. Eine Verwaltungsklasse ist mit zwei Methoden zu implementieren. Ein komplexer Algorithmus ist zu entwerfen und als Struktogramm darzustellen. Eine Client-Server-Lösung ist zu modellieren und ein Anwendungsfall in Form eines Sequenzdiagramm darzustellen.
Um eine Client-Server-Lösung zu realisieren, werden Inhalte aus Q2 „Datenkommunikation“ benötigt. Darüber hinaus enthält die Aufgabe Teile aus Q1 „Objektorientierte Softwareentwicklung“. Zur Bearbeitung der Aufgabe werden folgende verbindlichen Inhalte des Lehrplans gefordert: Objektorientierte Modellierung in UML (Klassen-, Objekt- und Sequenzdiagramm), Klassen, Kollektionen, Standardoperationen und die Visualisierung von strukturierten Abläufen. Des Weiteren das Client-Server-Prinzip, Nebenläufigkeit mit Threads sowie die Anwendung der Klassen zur Netzwerk-Kommunikation.

II Lösungshinweise

In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.1	überführen Kunde(kid, name, vorname, plz, ort, strasse) Mitarbeiter(mid, name, vorname, telefon, gehalt) Art(aid, bezeichnung, stundensatz) Befaehigung(aid#, mid#) Dienstleistung(did, datum, von, stunden, kid#, aid#, mid#) begründen Die Entitätstypen Kunde, Dienstleistung, Mitarbeiter und Art werden zu Tabellen. Bei 1:n-Beziehungen wie beispielsweise zwischen Mitarbeiter und Dienstleistung wird der PK der 1-Relation FK in der n-Relation. Die n:m-Beziehung „befähigt“ wird zur Beziehungsmengenrelation. Die PK der beteiligten Relationen Art und Mitarbeiter werden FK in der Beziehungsmengenrelation. Die Relationen befinden sich in der 3. Normalform, weil die Attribute Name und Adresse atomar sind, alle Nicht-Schlüsselattribute vom Primärschlüssel voll funktional abhängig sind und keine transitiven Abhängigkeiten, das heißt kein Nicht-Schlüsselattribut ist über den Umweg über ein anderes Nicht-Schlüsselattribut abhängig, existieren	3	2 1	 1

1.2.1	formulieren
	`SELECT name, vorname, bezeichnung,`
	`  SUM(stunden) AS Gesamtstunden`
	`FROM Mitarbeiter, Dienstleistung, Art`
	`WHERE Mitarbeiter.mid = Dienstleistung.mid`
	`AND Dienstleistung.aid = Art.aid`
	`AND datum LIKE '2018-01-%'`
	`GROUP BY name, vorname, bezeichnung;`
			4	

1.2.2	entwickeln
	`UPDATE Dienstleistung SET mid =`
	`  (SELECT mid FROM Mitarbeiter`
	`  WHERE name = 'Weiss' AND vorname = 'Tim')`
	`WHERE mid = 23`
	`AND datum >= NOW();`
	
	`DELETE FROM Befaehigung`
	`WHERE mid = 23;`
			1	3

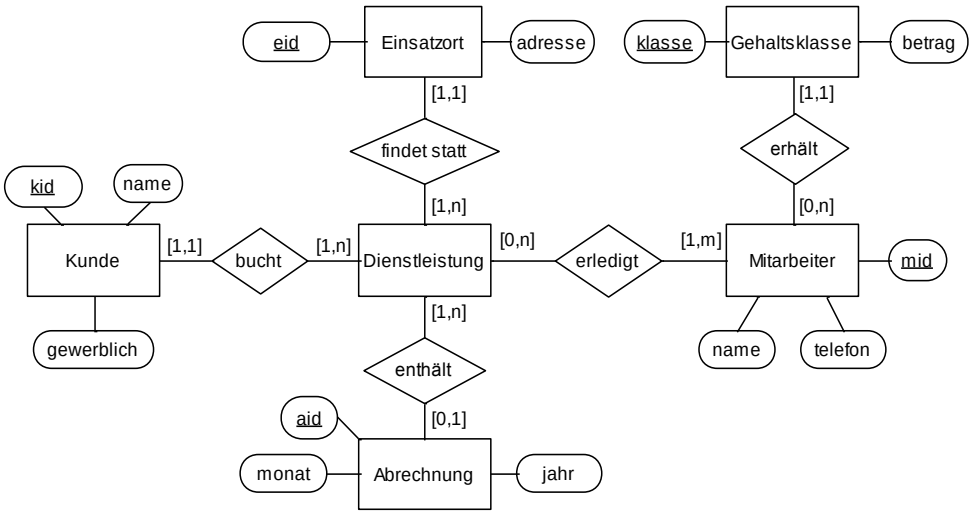
1.2.3	implementieren
	`SELECT name, vorname`
	`FROM Mitarbeiter, Befaehigung`
	`WHERE Mitarbeiter.mid = Befaehigung.mid`
	`AND aid = 17`
	`AND Mitarbeiter.mid NOT IN`
	`  (SELECT mid FROM Dienstleistung`
	`  WHERE datum = '2018-04-24');`
			2	2

Aufg.	erwartete Leistungen	BE		
		I	II	III

1.2.4	implementieren
	`SELECT datum, stunden, bezeichnung,`
	`  stundensatz*stunden AS betrag`
	`FROM Dienstleistung, Art`
	`WHERE Dienstleistung.aid = Art.aid`
	`AND kid = 4711`
	`AND datum LIKE '2017-12-%'`
	`ORDER BY datum;`
	
	`SELECT SUM(stunden*stundensatz) AS Gesamt`
	`FROM Dienstleistung, Art`
	`WHERE Dienstleistung.aid = Art.aid`
	`AND kid = 4711`
	`AND datum LIKE '2017-12-%';`
			2	3

1.3.1	zeigen Die Tabelle weist Redundanzen, beispielsweise Fensterreiniger in der Spalte Gerätetyp oder Kärcher in der Spalte Hersteller, auf. Das kann zu Inkonsistenzen führen, was bereits bei dem Hersteller Husqvarna-Husquarna in der vorliegenden Tabelle der Fall ist. angeben Mögliche Anomalien sind: Änderungsanomalie: Der Bestand des Reinigungsmittels „Konzentrat 100 ml“ wird an mehreren Stellen geführt. Wird Reinigungsmittel verbraucht und der Ist-Bestand nur an einer Stelle aktualisiert, entsteht Inkonsistenz. Einfügeanomalie: Ein neues Gerät kann nur eingefügt werden, wenn schon eine entsprechende Dienstleistungsart besteht. Löschanomalie: Wird die Dienstleistungsart „Fensterreinigung“ gelöscht, weil sie in einer anderen Art enthalten ist, gehen die Daten zu den Geräten und Verbrauchsmitteln ebenfalls verloren, obwohl sie noch vorhanden sind.	3	1	

1.3.2	überführen
	`Art(aid, Bezeichnung)`
	`Gerätetyp(tid, Bezeichnung)`
	`Gerät(gid, Modell, Anzahl_Geräte, tid#, Hersteller#)`
	`Hersteller(hid, Name)`
	`Verbrauchsmittel(vid, Bezeichnung, Soll, Ist)`
	`Art_Gerät(aid#, gid#)`
	`Gerät_Verbrauchsmittel(gid#, vid#)`
		4	2	

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.4	entwickeln, zeichnen  entwickeln zeichnen	2	2	2
	Summe 40	**12**	**17**	**11**

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.1.1	erläutern Vererbung ist ein wichtiges Konzept der objektorientierten Programmierung und ist deshalb in allen objektorientierten Sprachen enthalten. Ihr Hauptzweck liegt in der Nutzung der Ähnlichkeit von neu zu schaffenden Klassen mit bereits vorhandenen Klassen. Bei der Spezialisierung erweitert die Subklasse (Unterklasse) die Oberklasse (Superklasse) um spezifische Attribute und Methoden. Die Generalisierung abstrahiert mehrere Klassen, indem allgemeingültige Attribute und Methoden in eine Superklasse ausgelagert werden. Durch Vererbung kann man sich Entwicklungsarbeit sparen. Die verwendeten Klassen sind bereits ausgetestet. Programmierfehler können dann nur noch in den Erweiterungen auftreten.		3	
	beschreiben In dem vorliegenden Klassendiagramm ist die Klasse Person die Superklasse und die Klassen Kunde und Reinigungskraft sind Subklassen von Person. Die Subklassen erben die Attribute und Methoden der Superklasse. Auf die privaten Attribute darf allerdings auch in den Subklassen nur über entsprechende get- und set-Methoden zugegriffen werden. Das Zugriffsrecht protected (geschützt) bei dem Container für Buchungen erlaubt den Zugriff innerhalb der Vererbungshierarchie bzw. aus Klassen desselben Pakets. Die Klasse Person ist `abstract`, das heißt, sie stellt eine Schnittstelle für weitere Klassen dar. Es lassen sich keine Objekte von einer abstrakten Klasse erzeugen. Die abstrakte Methode `abrechnenMonat()` erzwingt, dass sie in den Unterklassen Kunde und Reinigungskraft überschrieben werden muss. Eine Methode ist polymorph, wenn sie in verschiedenen Klassen einer Vererbungshierarchie die gleiche Signatur hat, jedoch erneut implementiert ist wie beispielsweise die Methoden `abrechnenMonat()` und `toString()` der Klassen Kunde und Reinigungskraft. Welche der Methoden für ein konkretes Objekt verwendet wird, wird erst zur Laufzeit bestimmt (Dynamisches Binden).	3		

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.1.2	überführen, implementieren			

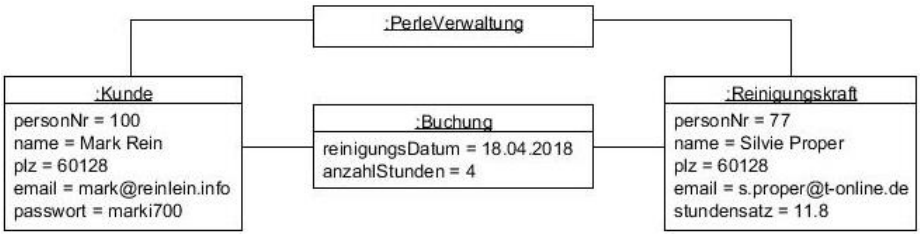
```
public abstract class Person {
  private int personNr;
  private String name;
  private String plz;
  private String email;
  private static int autoWert = 1;
  protected List<Buchung> buchungen;

  public Person(String name, String plz, String email) {
    this.personNr = autoWert++;
    this.name = name;
    this.plz = plz;
    this.email = email;
    buchungen = new List<Buchung>();
  }
  public void hinzufuegenBuchung(Buchung b) {
    buchungen.add(b);
  }
  public abstract double abrechnenMonat(int monat,
                                        int jahr);
  public String toString() {
    return name + " (Personennummer: " + personNr + "), " +
                  email + ", PLZ " + plz;
  }
}

public class Reinigungskraft extends Person {
  private double stundenSatz;
  public Reinigungskraft(String name, String plz,
                         String email, double satz) {
    super(name, plz, email);
    this.stundenSatz = satz;
  }
  public double abrechnenMonat(int monat, int jahr) {
    double betrag = 0;
    for(Buchung b : buchungen) {
      if(b.getReinigungsDatum().getMonth() == monat &&
         b.getReinigungsDatum().getYear() == jahr) {
        betrag += b.getAnzahlStunden()*stundenSatz*0.9;
        if(b.getAnzahlStunden() < 3) {
          betrag += 10;
        }
      }
    }
    return betrag;
  }

  public String toString() {
    return "Reinigungskraft " + super.toString() +
           ", Stundensatz " + stundenSatz + " EURO";
  }
}
```

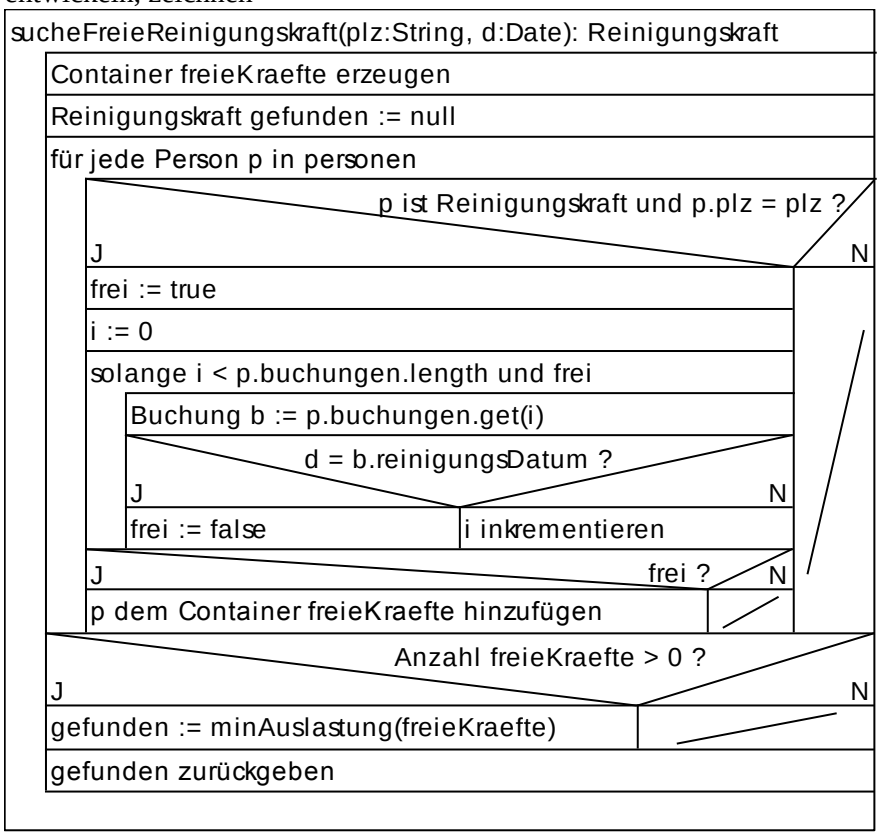
	überführen	4	2	
	implementieren		2	4

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.2.1	zeichnen  	3		

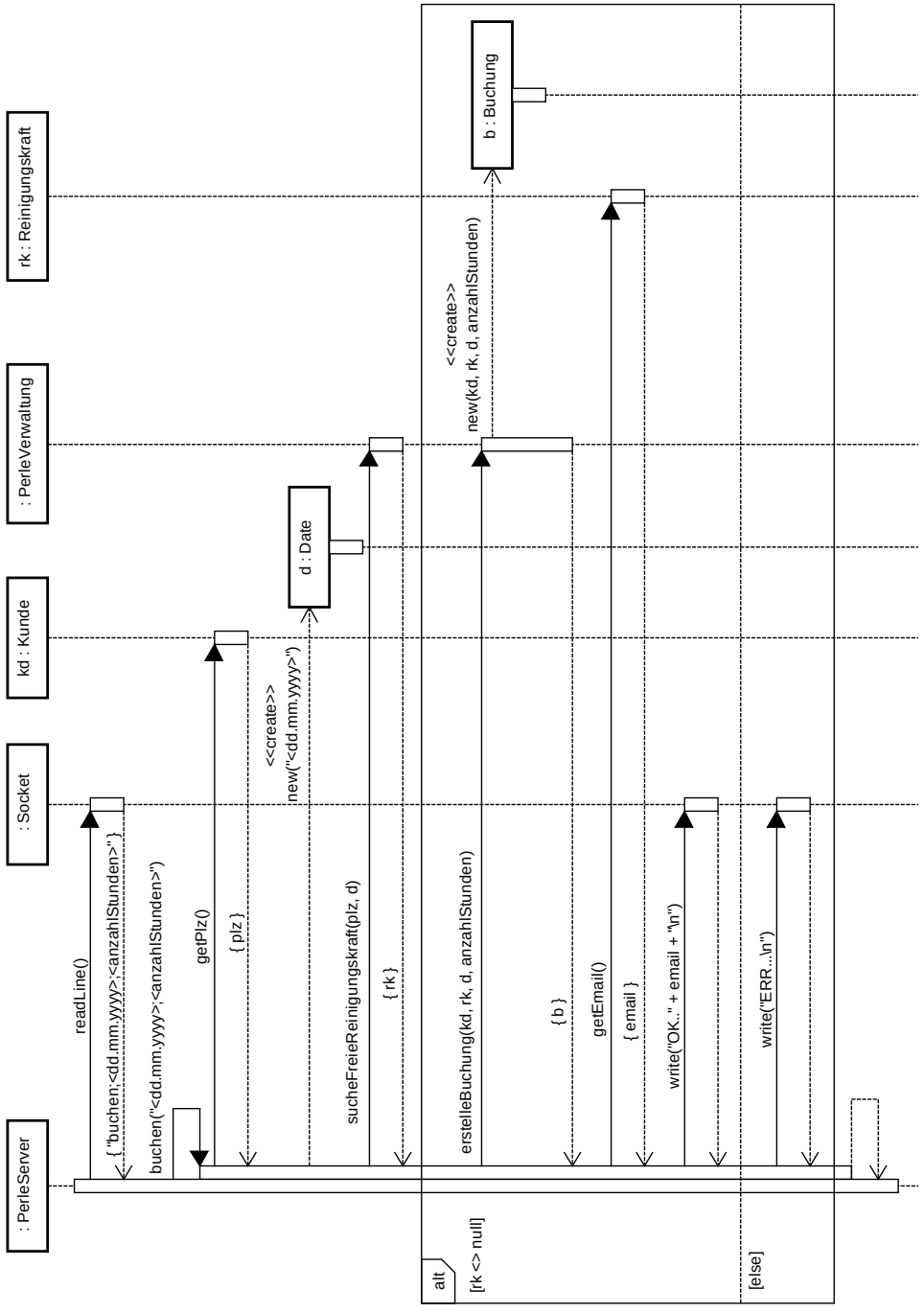
2.2.2	überführen, implementieren
	`public class PerleVerwaltung {`
	`  private List<Person> personen;`
	`  private PerleVerwaltung () {`
	`  personen = new List<Person>();`
	`  }`
	`  public Kunde anmelden(String email, String passwort) {`
	`  Kunde gesucht = null;`
	`  for(Person p : personen) {`
	`  if(p instanceof Kunde){`
	`  Kunde k = (Kunde)p;`
	`  if(k.getEmail().equals(email) &&`
	`  k.getPassword().equals(passwort)) {`
	`  gesucht = k;`
	`  break;`
	`  }`
	`  }`
	`  }`
	`  return gesucht;`
	`  }`
	`  public Kunde registriereKunde(String name, String plz,`
	`  String email, String pw){`
	`  Kunde k = new Kunde(name, plz, email, pw);`
	`  personen.add(k);`
	`  return k;`
	`  }`
	`}`			

	überführen	2		
	implementieren		3	2

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.2.3	entwickeln, zeichnen			

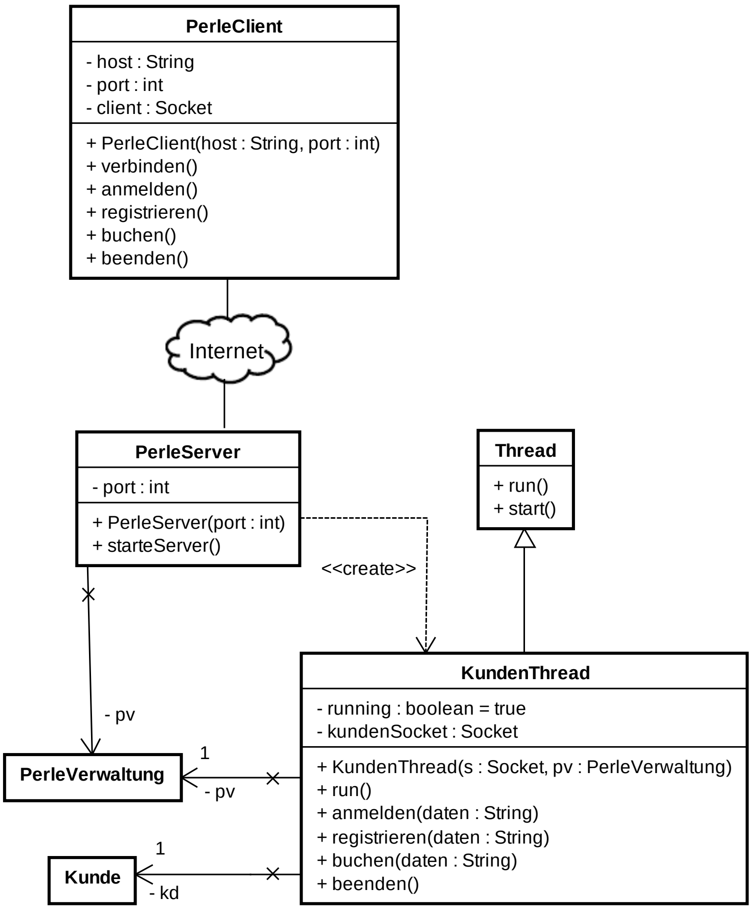


	entwickeln		3	3
	zeichnen	2		

Aufg.	erwartete Leistungen	BE I	BE II	BE III
2.3.1	entwickeln, zeichnen  entwickeln zeichnen	2	5	4

Nicht für den Prüfling bestimmt

<table>
<tr><th rowspan="2">Aufg.</th><th rowspan="2">erwartete Leistungen</th><th colspan="3">BE</th></tr>
<tr><th>I</th><th>II</th><th>III</th></tr>
<tr><td>2.3.2</td><td>modellieren, darstellen, zeichnen



modellieren
darstellen
zeichnen</td><td>

2</td><td>
3</td><td>3</td></tr>
<tr><td>2.3.3</td><td>implementieren

```
public class PerleServer {
  private int port;
  private PerleVerwaltung pv;

  public PerleServer(int port) {
    this.port = port;
    pv = new PerleVerwaltung();
  }

  public void starteServer() {
    ServerSocket server = new ServerSocket(port);
    while(true) {
      Socket client = server.accept();
      new KundenThread(client, pv).start();
    }
  }
}
```

</td><td></td><td>3</td><td>2</td></tr>
<tr><td></td><td>**Summe 60**</td><td>**18**</td><td>**24**</td><td>**18**</td></tr>
</table>

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt gemäß den Bestimmungen in der Oberstufen- und Abiturverordnung (OAVO) in der jeweils geltenden Fassung, insbesondere §33 OAVO in Verbindung mit den Anlagen 9a und ggf. 9b bis 9f, sowie in den Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA). Für die Umrechnung von Prozentanteilen der erbrachten Leistungen in Notenpunkte nach §9 Abs. 12 der OAVO gelten die Werte in der Anlage 9a der OAVO. Darüber hinaus sind die Vorgaben des Erlasses „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/schwerpunktbezogene Fächer)“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Bei der Bewertung und Beurteilung ist auch die Intensität der Bearbeitung zu berücksichtigen. Als Bewertungskriterien dienen über das Inhaltliche hinaus qualitative Merkmale wie Strukturierung, Differenziertheit und Schlüssigkeit der Argumentation.

Im Fach Datenverarbeitungstechnik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass insgesamt 46 BE, ein Prüfungsergebnis von **11 Punkten (gut)**, dass insgesamt 76 BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	12	17	11	**40**
2	18	24	18	**60**
Summe	**30**	**41**	**29**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.