

Name, Vorname:

Hilfsmittel:

zwei DIN A4-Seiten handschriftliche Aufschriebe (keine Kopien).

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I)

50P

Es soll eine Homepage der Firma Greif Bike Gmbh erstellt werden.

Dazu muss folgendes erfüllt sein:

1)

- a) Die Links im linken Fenster müssen linksbündig sein.
- b) Der Text in den rechten Fenstern muss zentriert sein.
- c) Der linke Fensterrahmen muss 212 Pixel breit sein

2)

- a) Beim Anklicken des Links Home muss das Motorrad im mittleren rechten Rahmen ausgestellt werden.
- b) Beim Anklicken des Links Kontakt muss die Adresse und die Telefonnummer der Firma im mittleren rechten Rahmen ausgestellt werden, indem der Inhalt der html-Datei kontakt.html im mittleren rechten Rahmen ausgestellt wird.
- c) Beim Anklicken des Links Händler muss der Text "wir sind registrierte Händler" im mittleren rechten Rahmen ausgestellt werden, indem der Inhalt der html-Datei haendler.html im mittleren rechten Rahmen ausgestellt wird.

3)

Mit Hilfe des Links zurück muss immer wieder in den Dateien kontakt.html und haendler.html die Einstiegsseite (siehe Vorlage auf der Rückseite) angeklickt werden können.

Bitte beachten Sie dazu die Vorlage auf der Rückseite!

<u>Home</u> <u>Kontakt</u> <u>Händler</u>	Greif Bike Gmbh Ihr Motorradhändler in Raschstadt
	Bild eines Motorrads 855 Pixel x 462 Pixel motorrad.jpg
	www.greif.bike.de

Name, Vorname:

Hilfsmittel:

eigene Aufschrieb, Skripte, Bücher

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrückungen der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABE

1) 30P

Es soll ein Taschenrechner (Addition und Subtraktion) mit Hilfe eines HTML-Formulars erzeugt werden.

Die Rechnung wird von einem Programm auf einem Webserver gemacht.

Nur entsprechendes HTML-Formular schreiben (kein Servlet).

2) 4P

Sie wollen einen Link in einer HTML-Datei erzeugen.

Was ist der Unterschied zwischen a href und link

3) 4P

Geben Sie ein konkretes Beispiel an, was man mit CSS und HTML im Gegensatz zu reinem HTML machen kann.

4)
Sie wollen alle ihre HTML-Dateien mit der Hintergrundfarbe rot versehen, indem in alle HTML-Dateien die Datei mycssstyle.css eingebunden wird.

a) 4P

Welchen Text muss die Datei myCssStyle.css enthalten?

Benutzen Sie dazu die Formateigenschaft background und die Farbe red.

In reinem HTML (ohne CSS) würde man es so machen:

```
<body bgcolor=red>
```

b) 4P

Wie (mit welchem Text) wird die Datei myCssStyle.css in eine HTML-Datei eingebunden?

5) a) 4P

Geben Sie ein konkretes Beispiel an, was man mit XML im Gegensatz zu HTML machen kann.

Lösungen:

1) a)

<body>

Taschenrechner

```
<form action="http://localhost:8080//test1.html" method="get">
1. Zahl:
<input type="text" name="Zahl1" size=60 maxlength=60 value=""><br><br>
2. Zahl:
<input type="text" name="Zahl2" size=60 maxlength=60 value=""><br><br>
<input type="submit" value="Addition">
<input type="submit" value="Subtraktion">
<input type="reset" value="Rücksetzen">
</form>
</body>
```

2) Mit a href kann man einen Link erzeugen

Mit link erzeugt man einen Verweis auf eine Ressource (z.B. eine css-Datei), die sich außerhalb der HTML-Datei befindet.

3) Der Überschrift <h1> ein anderes Erscheinungsbild geben, als dies standardmäßig vorgesehen ist.

4)

a) Inhalt der Datei mycssstyle.css:

```
body {background:red;}
```

b) Einbinden in HTML-Datei

```
<link rel=stylesheet type="text/css" href="myCssStyle.css">
```

5) a)

Man kann selbst ein Tag erstellen, das es in HTML nicht gibt.

Z.B. den Tag <mytag>

Name, Vorname:

Hilfsmittel:

zwei DIN A4-Seiten handschriftliche Aufschriebe (keine Kopien).

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I)

Erklärung und Beschreibung

Die schwarz gefüllte Raute mit 1..* am Ende bedeutet eine Komposition, d.h. zu einem Cocktail gehören mehrere Zutaten (hier maximal 20). Wenn der Cocktail stirbt (d.h. ein Objekt zerstört wird), sind auch die Zutaten nicht mehr vorhanden. Das bedeutet, daß die Zutaten (Objekte mit new) müssen innerhalb einer Methode der Klasse Cocktail erstellt werden, konkret der Methode addZutat(...).

Die ungefüllte (weiße) Raute mit 1 am Ende bedeutet eine Aggregation, d.h. zu einem Barmixer gehört genau ein Cocktail. Wenn der Barmixer entlassen wird, (d.h. ein Objekt zerstört wird), ist sein Cocktail noch vorhanden.

Das bedeutet, daß der Cocktail nicht in einer Methode der Klasse Barmixer erzeugt wird, sondern z.B. im Konstruktor bzw. in setCocktail(...), als Parameter (Referenz auf ein Objekt der Klasse Cocktail) übergeben wird.

II) konkrete Aufgaben:

1) Nur Systemintegratoren:

50P

Erstellen Sie die Klassen Vorrat, Zutat und Cocktail mit den jeweiligen (aus dem UML-Diagramm abgelesenen) Attributen und Methoden

2) Nur Anwendungsentwickler:

50P

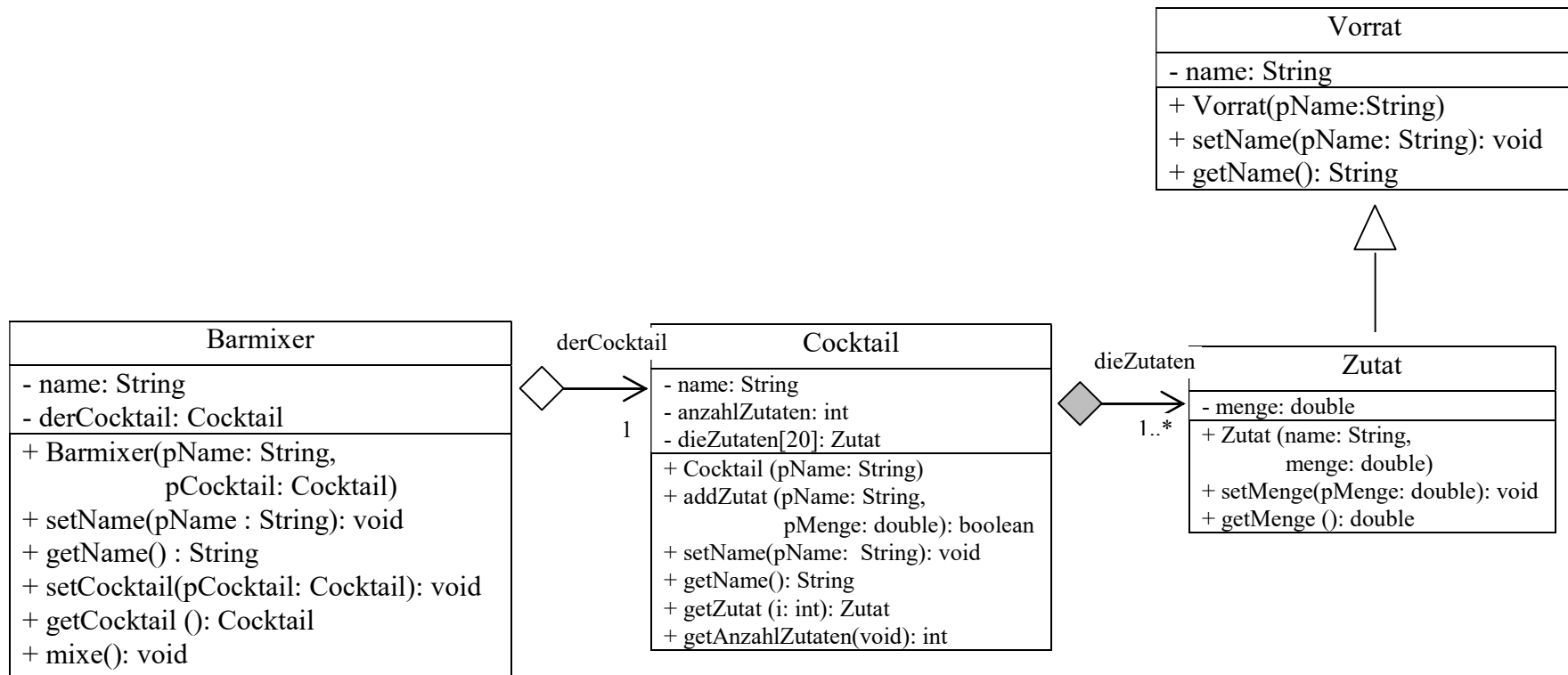
a) Erstellen Sie die Klassen Cocktail und Barmixer mit den jeweiligen (aus dem UML-Diagramm abgelesenen) Attributen und Methoden. Erstellen Sie in der Methode main() jeweils die folgenden zwei Cocktails (die jeweils aus 2 Zutaten bestehen):

Cuba libre: 100 gr Rum mit 10 gr Zucker

Whyski Sour: 200 gr Whyski mit 20gr Eis

Erstellen Sie den Barmixer Johnny, der den Cocktail Whyski Sour herstellt.

Geben Sie mit Hilfe der Methode mixe() den Namen des Barmixers, den Namen des Cocktails mit den entsprechenden Zutaten (Name und Menge) auf dem Bildschirm aus.



Bemerkung:

Die Methode mixe() gibt den Namen des Barmixers und den Namen des Cocktails mit den zugehörigen Zutaten (Name und Menge) auf dem Bildschirm aus.

Lösung:
Systemintegratoren

```
class Vorrat{                                     10P
    private String name;                        1P

    public Vorrat(String pName){ 3P
        name = pName;
    }

    public void setName(String pName){ 3P
        name = pName;
    }

    public String getName(){ 3P
        return name;
    }
}

class Zutat extends Vorrat{ 1P                                     12P
    private double menge; 1P

    public Zutat(String pName, double pMenge){ 4P
        super(pName);
        menge = pMenge;
    }

    public void setMenge(double menge) { 3P
        this.menge = menge;
    }

    public double getMenge() { 3P
        return menge;
    }
}
```

<pre> class Cocktail{ private String name; private int anzahlZutaten; private Zutat[] dieZutaten; public Cocktail(String pName) { name = pName; anzahlZutaten = 0; dieZutaten = new Zutat[20]; } public boolean addZutat(String pName, double pMenge) { if(anzahlZutaten<20){ dieZutaten[anzahlZutaten] = new Zutat(pName, pMenge); anzahlZutaten++; return true; } else{ return false; } } public void setName(String name){ this.name = name; } public String getName(){ return name; } public int getAnzahlZutaten(){ return anzahlZutaten; } public Zutat getZutat(int i){ if(i<anzahlZutaten){ return dieZutaten[i]; } else{ return null; } } } </pre>	<p>30P</p> <p>1P</p> <p>1P</p> <p>1P</p> <p>6P</p> <p>6P</p> <p>3P</p> <p>3P</p> <p>3P</p> <p>6P</p>
---	--

Anwendungsentwickler:

```
class Cocktail{
    private String name;          1P
    private int anzahlZutaten;    1P
    private Zutat[] dieZutaten;  1P

    public Cocktail(String pName) { 2P
        name = pName;
        anzahlZutaten = 0;
        dieZutaten = new Zutat[20];
    }

    public boolean addZutat(String pName, double pMenge) { 4P
        if(anzahlZutaten<20){
            dieZutaten[anzahlZutaten] = new Zutat(pName, pMenge);
            anzahlZutaten++;
            return true;
        }
        else{
            return false;
        }
    }

    public void setName(String name){ 2P
        this.name = name;
    }

    public String getName(){ 2P
        return name;
    }

    public int getAnzahlZutaten(){ 2P
        return anzahlZutaten;
    }

    public Zutat getZutat(int i){ 4P
        if(i<anzahlZutaten){
            return dieZutaten[i];
        }
        else{
            return null;
        }
    }
}
```

```

class Barmixer{
    private String name;          1P
    private Cocktail derCocktail; 1P

    Barmixer(String pName, Cocktail pCocktail){2P
        name = pName;
        derCocktail=pCocktail;
    }

    public void setName(String pName){          2P
        name = pName;
    }

    public String getName(){                    2P
        return name;
    }

    public void setCocktail(Cocktail pCocktail){ 2P
        derCocktail = pCocktail;
    }

    public Cocktail getCocktail() {              2P
        return derCocktail;
    }

    public void mixe(){                          5P
        int i;
        System.out.println("Der Barmixer "+name+" mixt einen "
            +derCocktail.getName());
        System.out.println("Die Zutaten sind: ");
        for(i=0; i<derCocktail.getAnzahlZutaten();i++){
            System.out.print(derCocktail.getZutat(i).getMenge()+" ");
            System.out.println(derCocktail.getZutat(i).getName()
                +" Einheiten ");
        }
    }
}

public static void main(String[] args) {
    Cocktail c1 = new Cocktail("cuba_libre"); 1P
    Cocktail c2 = new Cocktail("WhyskiSour"); 1P
    c1.addZutat("Rum", 100); 2P
    c1.addZutat("Zucker", 10); 2P
    c2.addZutat("Whyski", 200); 2P
    c2.addZutat("Eis", 20); 2P
    Barmixer b1 = new Barmixer("Johnny", c1); 2P
    b1.setCocktail(c2); 2P
    b1.mixe(); 2P
}

```

16P

Name, Vorname:

Hilfsmittel:

selbstgeschriebenes zweiseitiges DIN A4 Blatt

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

1a) 2P

Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!)

```
...
int[] v;
v = new int[3];
v[3] = -3;
...
```

b) 2P

Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!)

```
...
double[] w, zahlen;
zahlen[0] = 12;
...
```

c) 2P

Ist der folgende Java-Programmausschnitt syntaktisch korrekt?

Wenn ja, welche Werte haben die Elemente der Variablen w?

```
...
int[] w;
w = new int[2];
w[0] = -123;
...
```

2) 4 P

Die Klasse Schaf soll schon existieren (mit genau einem zweiparametrischen Konstruktor aus integer Parametern, die das Gewicht und das Alter festlegen).

Erstellen Sie das Feld "schafstall" mit 2 Schafen.

Das erste Schaf wiegt 20 Kilo und ist 2 Jahre alt,

das zweite Schaf wiegt 30 Kilo und ist 3 Jahre alt.

3)

a)

18P

Erzeugen Sie die Klasse Kreis mit genau dem Attribut (und nur dem Attribut) "radius", den zu "radius" gehörigen get-und set-Methoden und 2 Konstruktoren.

Erzeugen Sie die Methode berechneFlaeche und berechneUmfang.

b)

22P

Machen Sie in main() hintereinander Folgendes

b1) Erstellen Sie in dem Feld "kreise" 100 Kreise mit den Radien 0, 1, 2, ...99

b2) Geben Sie die Flächeninhalte dieser Kreise auf dem Bildschirm aus.

b3) Verdoppeln Sie die Radien dieser Kreise

b4) Speichern Sie diese Kreise (mit doppeltem Radius) zur Sicherheit in einem neuen Feld mit dem Namen "kreisSicherung".

b5) Angenommen, jemand fügt an das Ende in main()die 2 folgenden Programmierzeilen:

```
kreisSicherung[10].setRadius(13);  
System.out.println("Radius =" + kreise[10].getRadius());
```

Was wird auf dem Bildschirm ausgegeben?

Begründen Sie!

Lösungen:

1a) 2P

`v[3] = -3;` überschreibt nicht reservierten Speicher

1b) 2P

Für das Feld `zahlen` wurde kein Speicher reserviert.

`zahlen` ist eine lokale Variable, deren Wert undefiniert ist.

c) Die Werte des Felds `w` sind nach dem Erstellen mit 0 vorbelegt, `w[1]` wurde verändert, also: 2P

`w[0] = -123`

`w[1] = 0`

2) 4P

`Schaf [] schafstall = {new Schaf (20, 2), new Schaf (30, 3)};`

3) a)

```
class Kreis{
    private double radius;                // 2P
    public static final double pi = 3.14;

    public Kreis(double pRadius){         // 2P
        radius = pRadius;
    }

    public Kreis(){                       // 2P
    }

    public void setRadius(double pRadius){ // 3P
        radius = pRadius;
    }

    public double getRadius(){            // 3P
        return(radius);
    }

    public double berechneUmfang(){       // 3P
        return(2*pi*radius);
    }

    public double berechneFlaeche(){      // 3P
        return(pi*radius*radius);
    }
}
```

```

public class MainBK11_13_11_10_Nr3 {
    public static void main(String[] args) {
        int i;
        Kreis[] kreise; //1P
        kreise = new Kreis[100]; //1P
        for(i=0;i<100;i++){ //4P
            kreise[i]=new Kreis(i);
        }

        for(i=0;i<100;i++){ //4P
            System.out.println("Radius="+kreise[i].berechneFlaeche());
        }

        for(i=0;i<100;i++){ //4P
            kreise[i].setRadius(kreise[i].getRadius()*2);
        }

        Kreis[] kreisSicherung = new Kreis[100]; //2P
        for(i=0;i<100;i++){ //4P
            kreisSicherung[i]=kreise[i];
        }

        // Radius=20 //1P
        // kreisSicherung[10] und kreise [10] zeigen auf das //1P
        // gleiche Objekt.
    }
}

```

KLAUSUR 3 SAE E3FI 21.3.2011 Zeit: 90 Minuten
ANWENDUNGSENTWICKLER

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

Außendienstmitarbeiter sollen durch ein Programm unterstützt werden, das die häufigen Updates der Produktdaten ermöglicht.

Teile des Programms wurden schon erstellt (siehe Anlagen UML und Programm).

1) Implementieren Sie die Klasse Artikelliste mit den dazugehörigen Attributen und den Methoden:

erzeugeListe(...)

getArtikelAt(...)

Falls es für den Programmierer bequemer wird, kann er auch noch weitere Methoden entwerfen, wie z.B. die Ermittlung der Größe eines ResultSet.

2) Erstellen Sie in main() mittels der Methode erzeugeListe(...) eine Artikelliste aller Artikel, die in der Datenbank vorkommen.

Da MS-Access den letzten Artikel in der Datenbank immer mit 0 vorbelegt (bzw. Strings mit null), soll dieser letzte Artikel in der Artikelliste nicht vorkommen.

3) Geben Sie diese Artikel (mit ihren Eigenschaften, also den Wert der Attribute) auf dem Bildschirm aus. Verwenden Sie dazu u.a. die Methode getArtikelAt(...).

Bemerkungen:

1) In dem UML-Diagramm und in der Beschreibung der Klassen ist nur ein Teil der Attribute und Methoden aufgeführt.

2) Das komplette Projekt befindet sich auf dem Tauschordner

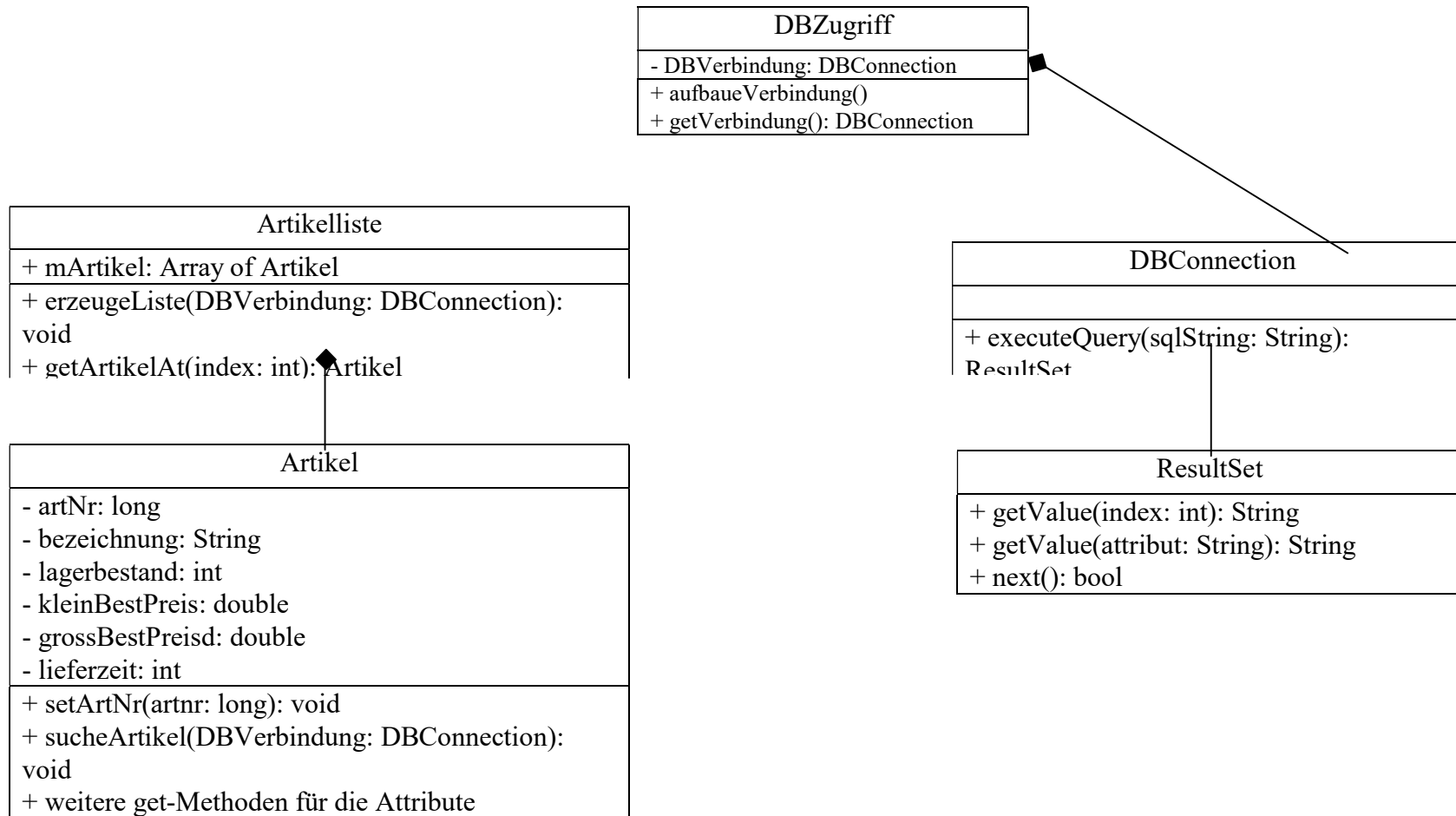
3) Der Pfad zur Datenbank muss noch im Programm angepasst werden

4) Es müssen die Prinzipien der OOP verwendet werden.

Beschreibung der Klassen

Klasse Artikelliste	
Eigenschaften	
mArtikel	enthält eine passende Anzahl von Objekten vom Typ Artikel
Methoden	
erzeugeListe	erzeugt eine Liste mit allen Artikeln, die in der Datenbank vorhanden sind
getArtikelAt	gibt eine Referenz auf den Artikel an der Position 'index' zurück. Bei fehlerhaften Indizee wird eine Referenz auf null zurückgegeben.
Klasse DBConnection	
Methoden	
executeQuery	übergibt den sqlString an die Datenbank und gibt eine Ergebnistabelle vom Typ ResultSet als mehrdimensionales Array zurück.
Klasse Artikel	
Eigenschaften	
artNr	enthält die Artikelnummer
bezeichnung	enthält die Bezeichnung
lagerbestand	enthält den Lagerbestand
kleinBestPreis	enthält den Preis für kleine Bestellmengen
grossBestPreis	enthält den Preis für große Bestellmengen
lieferzeit	Lieferzeit in Tagen, falls Artikel nicht vorrätig
Methoden	
setArtNr	setzt die Artikelnummer
sucheArtikel	sucht die Daten des Artikels anhand der Eigenschaft ArtNr aus der Datenbank und trägt sie in die eigenen Eigenschaften ein
get-Methoden	für jede Eigenschaft ist eine get-Methode vorhanden
Klasse ResultSet	
Methoden	
getValue(index:int)	gibt den Wert der mit 'index' angegebenen Spalte des ResultSets einer Datenbankabfrage als String zurück
getValue(attribut: String)	gibt den Wert der mit 'attribut' angegebenen Spalte des ResultSets einer Datenbankabfrage als String zurück.
next	setzt den Datensatzzeiger um eine Zeile vor. Am Ende des ResultSets wird 'false' zurückgegeben, sonst 'true' Der erste Datensatz wird erst erreicht, wenn 'next()' das erste Mal aufgerufen wird. Ist das ResultSet leer, wird dabei bereits 'false' zurückgegeben.
Klasse DBZugriff	
Eigenschaften	
DBVerbindung	enthält die mit 'aufbaueVerbindung' aufgebaute Verbindung zur Firmendatenbank
Methoden	
aufbaueVerbindung	Diese Methode stellt eine Verbindung zur Firmendatenbank her, unabhängig von der physikalischen Anbindung. Die Verbindungsdaten (wie username etc.) werden statisch vorgehalten. Die Datenbankverbindung wird in der Eigenschaft DB-Verbindung ablegt.
getVerbindung	Diese Methode gibt die Datenbankverbindung 'DB-Verbindung' zurück. Damit können Daten z.B. mit SQL-Befehlen von der Datenbank auf den Client geladen werden.

I) UML-Diagramm



Quellcode

```
package w2006aenr2ver1;

import java.sql.*;

public class MainW2006AENr2Ver1 {
    public static void main(String[] args) throws Throwable {
        // muß implementiert werden
    }
}

/* WICHTIGE BEMERKUNG (siehe API)
By default, only one ResultSet object per Statement object can be open
at the same time. Therefore, if the reading of one ResultSet object
is interleaved with the reading of another, each must have been generated
by different Statement objects. All execution methods in the Statement
interface implicitly close a statment's current ResultSet object if an
open one exists.

class Artikelliste{
    // muß implementiert werden
}
```

```
/*
Enthält neben den elementaren Attributen auch die Methode
sucheArtikel() , die auf die Datenbank zugreift.
Dieses Design ist suboptimal. Diese Methode sollte in eine
andere Klasse aufgenommen werden.
*/
```

```
class Artikel {
    private String artNr;
    private String bezeichnung;
    private int lagerbestand;
    private double kleinBestPreis;
    private double grossBestPreis;
    private int lieferzeit;

    Artikel() {
    }

    public Artikel(String artNr, String bezeichnung, int lagerbestand,
        double kleinBestPreis, double grossBestPreis, int lieferzeit) {
        this.artNr = artNr;
        this.bezeichnung = bezeichnung;
        this.lagerbestand = lagerbestand;
        this.kleinBestPreis = kleinBestPreis;
        this.grossBestPreis = grossBestPreis;
        this.lieferzeit = lieferzeit;
    }

    public void setArtNr(String artNr) {
        this.artNr = artNr;
    }

    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    public void setGrossBestPreis(double grossBestPreis) {
        this.grossBestPreis = grossBestPreis;
    }

    public void setKleinBestPreis(double kleinBestPreis) {
        this.kleinBestPreis = kleinBestPreis;
    }

    public void setLagerbestand(int lagerbestand) {
        this.lagerbestand = lagerbestand;
    }

    public void setLieferzeit(int lieferzeit) {
        this.lieferzeit = lieferzeit;
    }
}
```

```

    public String getArtNr() {
        return artNr;
    }

    public String getBezeichnung() {
        return bezeichnung;
    }

    public double getGrossBestPreis() {
        return grossBestPreis;
    }

    public int getLagerbestand() {
        return lagerbestand;
    }

    public int getLieferzeit() {
        return lieferzeit;
    }

    /*
    Zugriff auf DB (eigentlich schlechtes Design)
    Setzt eine Verbindung zur Datenbank voraus (als Parameter in der Methode).
    */
    public void sucheArtikel(DBConnection pDBConnection) throws Throwable
    {
        // der gefundene Artikel
        String mSQL;
        ResultSet rsA;
        mSQL = "SELECT ArtNr, Bezeichnung, Lagerbestand, KleinBestPreis,
                GrossBestPreis, Lieferzeit FROM tblArtikel ";
        mSQL = mSQL + "WHERE ArtNr = '" + artNr + "';";

        rsA = pDBConnection.executeQuery(mSQL);

        try {
            //System.out.println("mSQL="+mSQL);
            rsA.next();
            //artikelnummer = rsA.getString("Artikelnummer");
            bezeichnung = rsA.getString("Bezeichnung");
            lagerbestand = rsA.getInt("Lagerbestand");
            kleinBestPreis = rsA.getDouble("KleinBestPreis");
            grossBestPreis = rsA.getDouble("GrossBestPreis");
            lieferzeit = rsA.getInt("Lieferzeit");
        } catch (Throwable t) {
            t = new Throwable("Datensatz in sucheArtikel konnte nicht
                               gelesen werden in Artikel.sucheArtikel()");
            throw t;
        }
        rsA.close();
    }
}

/*
Hier wird die Verbindung zur Datenbank aufgebaut.
Die Verbindung wird charakterisiert durch 3 Zustände:
- Datenquelle: Datei
- Connection: Die eigentliche Anbindung
- Statement: Möglichkeit Datenbankabfragen zu stellen
Diese werden in der Klasse DBConnection verwaltet.
*/
class DBZugriff {
    private DBConnection DBVerbindung;

    public DBZugriff() throws Throwable{
        DBVerbindung = new DBConnection();
        // Die o.g. 3 Zustände werden geeignet initialisiert und im Attribut
        // DBVerbindung gespeichert.
        aufbaueVerbindung();
    }
}

```

```

public void aufbaueVerbindung() throws SQLException {
    String datenQuelle;
    Connection connectionDB;
    Statement statementsSQL;
    try {
        // Lädt die Klasse mit dem Namen
        "sun.jdbc.odbc.JdbcOdbcDriver"
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        // Variable für den Treiber und den Pfad zur DB (Deklaration
        // und Wertzuweisung)
        // getConnection(...) benötigt 3 Parameter:
        // eine Datenquelle, einen User, ein Passwort
        // eine Datenquelle ist wie folgt aufgebaut:
        // jdbc:Subprotokoll:Datenquellennamen
        // Für ODBC-Datenquellen ist das Subprotokoll obcd
        //
        // datenQuelle = "jdbc:odbc:Driver={Microsoft Access Driver
        (*.mdb)};DBQ=H:/Daten_Austausch/MEINE_SKRIPTTE/ProgJava/100_Eigenes/PROG/Abs
chlusspruefungenFI_NetBeans/W2006AENr2Ver1/dBMSAccess2.mdb";
        // Dieser Pfad muss am jeweiligen Computer angepasst werden!
        datenQuelle = "jdbc:odbc:Driver={Microsoft Access Driver
        (*.mdb)};DBQ=C:/Daten_Austausch/MEINE_SKRIPTTE/ProgJava/100_Eigenes/PROG/Abs
chlusspruefungenFI_NetBeans/W2006AENr2Ver1/dBMSAccess2.mdb";

        connectionDB = DriverManager.getConnection(datenQuelle, "",
        "");
        statementsSQL = connectionDB.createStatement();
        DBVerbindung.setConnectionDB(connectionDB);
        DBVerbindung.setDatenQuelle(datenQuelle);
        DBVerbindung.setStatementsSQL(statementsSQL);
    } catch (SQLException fehler) {
        fehler = new SQLException("Treiber existiert nicht in in
        DBZugriff.abbauenVerbindung()");
        throw fehler;
    } catch (ClassNotFoundException fehler) {
        System.out.println("Treiber existiert nicht in
        DBZugriff.abbauenVerbindung()");
        fehler.printStackTrace();
    }
}

public void abbauenVerbindung() throws SQLException {
    try {
        DBVerbindung.getConnectionDB().close();
        DBVerbindung.getStatementsSQL().close();
    } catch (SQLException fehler) {
        fehler = new SQLException("DB konnte nicht geschlossen werden
        in DBZugriff.abbauenVerbindung()");
        throw fehler;
    }
}

public DBConnection getVerbindung() throws SQLException {
    aufbaueVerbindung();
    return DBVerbindung;
}
}

```

```

/*
Die Verbindung wird charakterisiert durch 3 Zustände:
- Datenquelle: Datei
- Connection: Die eigentliche Anbindung
- Statement: Möglichkeit Datenbankabfragen zu stellen
*/

class DBConnection {
    private String datenQuelle;
    private Connection connectionDB;
    private Statement statementsSQL;

    public DBConnection() {
    }

    public Connection getConnectionDB() {
        return connectionDB;
    }

    public String getDatenQuelle() {
        return datenQuelle;
    }

    public Statement getStatementsSQL() {
        return statementsSQL;
    }

    public void setConnectionDB(Connection connectionDB) {
        this.connectionDB = connectionDB;
    }

    public void setDatenQuelle(String datenQuelle) {
        this.datenQuelle = datenQuelle;
    }

    public void setStatementsSQL(Statement statementsSQL) {
        this.statementsSQL = statementsSQL;
    }

    public ResultSet executeQuery(String sqlString) throws SQLException {
        ResultSet rs;
        try {
            rs = statementsSQL.executeQuery(sqlString);
        } catch (SQLException fehler) {
            fehler = new SQLException("Fehler in
                                     DBConnection.executeQuery()");
            throw fehler;
        }
        return rs;
    }
}

```

Lösung:

```
package w2006aenr2ver1;

import java.sql.*;

public class MainW2006AENr2Ver1 {

    public static void main(String[] args) throws Throwable {

        int anz;
        int i;
        DBZugriff dbZugriff;
        dbZugriff = new DBZugriff();
        Artikelliste al;
        al = new Artikelliste();
        // Hier gibt es Probleme
        al.erzeugeListe(dbZugriff.getVerbindung());
        anz = al.getAnzahlArtikel();

        /*
            anz = al.getMArtikel().length;
            for(i=0;i<anz;i++){
                System.out.println("Artnr="+al.getMArtikel()[i].getArtNr());
            }
        */

        for(i=0;i<anz-1;i++){
            System.out.println("Artnr="+al.getArtikelAt(i).getArtNr());
            System.out.println("Bezeichnung="+al.getArtikelAt(i).getBezeichnung());
            System.out.println("Grossbestellpreis="+al.getArtikelAt(i).getGrossBestPreis());
            System.out.println("Kleinbestellpreis="+al.getArtikelAt(i).getLagerbestand());
            System.out.println("Lieferzeit="+al.getArtikelAt(i).getLieferzeit());
        }

    }

    /*
        Enthält neben dem Array auch die Methode
        erzeugeListe() , die auf die Datenbank zugreift.
        Dieses Design ist suboptimal. Diese Methode sollte in eine
        andere Klasse aufgenommen werden.
    */

    class Artikelliste {
        // enthält alle Artikel aus der Datenbank
        private Artikel[] mArtikel;
        private int anzahlArtikel;

        public Artikel[] getMArtikel() {
            return mArtikel;
        }

        public Artikelliste() {
        }

        public void erzeugeListe(DBConnection pDBVerbindung) throws Throwable{
            boolean b;
            int i;
            String mySQL;
            ResultSet rsA;

            String artNr;
```



```

String bezeichnung;
int lagerbestand;
double kleinBestPreis;
double grossBestPreis;
int lieferzeit;

anzahlArtikel = berechneAnzahlArtikel(pDBVerbindung);
mySQL = "SELECT ArtNr, Bezeichnung, Lagerbestand, KleinBestPreis,
GrossBestPreis, Lieferzeit FROM tblArtikel ";
rsA = pDBVerbindung.executeQuery(mySQL);
mArtikel = new Artikel[anzahlArtikel];
for(i=0;i<anzahlArtikel-1;i++){
    mArtikel[i]=new Artikel();
}

for(i=0;i<anzahlArtikel-1;i++){
    try {
        b=rsA.next();
        if(b==false){
            break;
        }
        // ArtNr dem ResultSet entnehmen
        // Hier gibt es Probleme
        artNr = rsA.getString("ArtNr");
        // Setze ArtNr im i-ten Element
        mArtikel[i].setArtNr(artNr);
    }
}

/* WICHTIGE BEMERKUNG (siehe API)
By default, only one ResultSet object per Statement object can be open
at the same time. Therefore, if the reading of one ResultSet object
is interleaved with the reading of another, each must have been generated
by different Statement objects. All execution methods in the Statement
interface implicitly close a statment's current ResultSet object if an
open one exists.

DESWGEN geht die folgende Suche nicht,
mArtikel[i].sucheArtikel(pDBVerbindung);
weil man dann konkurrierende Result-Sets bekommt.
*/
        //mArtikel[i].sucheArtikel(pDBVerbindung);
    } catch (Throwable t) {
        t = new Throwable("Datensatz in erzeugeListe konnte nicht
gelesen werden in Artikelliste.erzeugeListe");
        throw t;
    }
}
rsA.close();

for(i=0;i<anzahlArtikel-1;i++){
    try {
        // Suche nach diesem Artikel
        mArtikel[i].sucheArtikel(pDBVerbindung);
    } catch (Throwable t) {
        t = new Throwable("Datensatz in erzeugeListe konnte nicht
gelesen werden");
        throw t;
    }
}

}

/*
Größe des Result-Sets bestimmen
*/
private int berechneAnzahlArtikel(DBConnection pDBVerbindung) throws
Throwable{
    boolean b;
    int i;
    int anzahl=0;
    String mySQL;
    ResultSet rsA;

    String artNr;

```

```

        String bezeichnung;
        int lagerbestand;
        double kleinBestPreis;
        double grossBestPreis;
        int lieferzeit;

        mySQL = "SELECT ArtNr, Bezeichnung, Lagerbestand, KleinBestPreis,
GrossBestPreis, Lieferzeit FROM tblArtikel ";
        rsA = pDBVerbindung.executeQuery(mySQL);
        // Länge des ResultSet bestimmen
        do{
            b = rsA.next();
            anzahl++;
        }while (b==true);

        rsA.close();
        return(anzahl);
    }

    public Artikel getArtikelAt(int index){
        Artikel artikel = null;
        if(index>=0 && index<anzahlArtikel){
            artikel=mArtikel[index];
        }
        return artikel;
    }

    public int getAnzahlArtikel(){
        return anzahlArtikel;
    }
}

/*
Enthält neben den elementaren Attributen auch die Methode
sucheArtikel() , die auf die Datenbank zugreift.
Dieses Design ist suboptimal. Diese Methode sollte in eine
andere Klasse aufgenommen werden.
*/
class Artikel {
    private String artNr;
    private String bezeichnung;
    private int lagerbestand;
    private double kleinBestPreis;
    private double grossBestPreis;
    private int lieferzeit;

    Artikel() {

    }

    public Artikel(String artNr, String bezeichnung, int lagerbestand,
double kleinBestPreis, double grossBestPreis, int lieferzeit) {
        this.artNr = artNr;
        this.bezeichnung = bezeichnung;
        this.lagerbestand = lagerbestand;
        this.kleinBestPreis = kleinBestPreis;
        this.grossBestPreis = grossBestPreis;
        this.lieferzeit = lieferzeit;
    }

    public void setArtNr(String artNr) {
        this.artNr = artNr;
    }

    public void setBezeichnung(String bezeichnung) {
        this.bezeichnung = bezeichnung;
    }

    public void setKleinBestPreis(double grossBestPreis) {
        this.kleinBestPreis = kleinBestPreis;
    }
}

```

```

public void setGrossBestPreis(double grossBestPreis) {
    this.grossBestPreis = grossBestPreis;
}

public void setLagerbestand(int lagerbestand) {
    this.lagerbestand = lagerbestand;
}

public void setLieferzeit(int lieferzeit) {
    this.lieferzeit = lieferzeit;
}

public String getArtNr() {
    return artNr;
}

public String getBezeichnung() {
    return bezeichnung;
}

public double getKleinBestPreis() {
    return kleinBestPreis;
}

public double getGrossBestPreis() {
    return grossBestPreis;
}

public int getLagerbestand() {
    return lagerbestand;
}

public int getLieferzeit() {
    return lieferzeit;
}

/*
Zugriff auf DB (eigentlich schlechtes Design)
Setzt eine Verbindung zur Datenbank voraus (als Parameter in der Methode).
*/
public void sucheArtikel(DBConnection pDBConnection) throws Throwable
{
    // der gefundene Artikel
    String mSQL;
    ResultSet rsA;
    mSQL = "SELECT ArtNr, Bezeichnung, Lagerbestand, KleinBestPreis,
GrossBestPreis, Lieferzeit FROM tblArtikel ";
    mSQL = mSQL + "WHERE ArtNr = '" + artNr + "'";

    rsA = pDBConnection.executeQuery(mSQL);

    try {
        //System.out.println("mSQL="+mSQL);
        rsA.next();
        //artikelnummer = rsA.getString("Artikelnummer");
        bezeichnung = rsA.getString("Bezeichnung");
        lagerbestand = rsA.getInt("Lagerbestand");
        kleinBestPreis = rsA.getDouble("KleinBestPreis");
        grossBestPreis = rsA.getDouble("GrossBestPreis");
        lieferzeit = rsA.getInt("Lieferzeit");
    } catch (Throwable t) {
        t = new Throwable("Datensatz in sucheArtikel konnte nicht
gelesen werden in Artikel.sucheArtikel()");
        throw t;
    }
    rsA.close();
}
}

```

```

/*
Hier wird die Verbindung zur Datenbank aufgebaut.
Die Verbindung wird charakterisiert durch 3 Zustände:
- Datenquelle: Datei
- Connection: Die eigentliche Anbindung
- Statement: Möglichkeit Datenbankabfragen zu stellen
Diese werden in der Klasse DBConnection verwaltet.
*/
class DBZugriff {
    private DBConnection DBVerbindung;

    public DBZugriff() throws Throwable{
        DBVerbindung = new DBConnection();
        // Die o.g. 3 Zustände werden geeignet initialisiert und im
Attribut
        // DBVerbindung gespeichert.
        aufbaueVerbindung();
    }

    public void aufbaueVerbindung() throws SQLException {
        String datenQuelle;
        Connection connectionDB;
        Statement statementsSQL;
        try {
            // Lädt die Klasse mit dem Namen "sun.jdbc.odbc.JdbcOdbcDriver"
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            // Variable für den Treiber und den Pfad zur DB (Deklaration
und Wertzuweisung)
            // getConnection(...) benötigt 3 Parameter:
            // eine Datenquelle, einen User, ein Passwort
            // eine Datenquelle ist wie folgt aufgebaut:
            // jdbc:Subprotokoll:Datenquellennamen
            // Für ODBC-Datenquellen ist das Subprotokoll obcd
            //
            // datenQuelle = "jdbc:odbc:Driver={Microsoft Access Driver
(*.mdb)};DBQ=H:/Daten_Austausch/MEINE_SKRIPTTE/ProgJava/100_Eigenes/PROG/Abs
chlusspruefungenFI_NetBeans/W2006AENr2Ver1/dBMSAccess2.mdb";
            // Dieser Pfad muss am jeweiligen Computer angepasst werden!
            datenQuelle = "jdbc:odbc:Driver={Microsoft Access Driver
(*.mdb)};DBQ=C:/Daten_Austausch/MEINE_SKRIPTTE/ProgJava/100_Eigenes/PROG/Abs
chlusspruefungenFI_NetBeans/AE/W2006AENr2Ver1/dBMSAccess2.mdb";

            connectionDB = DriverManager.getConnection(datenQuelle, "",
""");
            statementsSQL = connectionDB.createStatement();
            DBVerbindung.setConnectionDB(connectionDB);
            DBVerbindung.setDatenQuelle(datenQuelle);
            DBVerbindung.setStatementsSQL(statementsSQL);
        } catch (SQLException fehler) {
            fehler = new SQLException("Treiber existiert nicht in in
DBZugriff.abbauenVerbindung()");
            throw fehler;
        } catch (ClassNotFoundException fehler) {
            System.out.println("Treiber existiert nicht in
DBZugriff.abbauenVerbindung()");
            fehler.printStackTrace();
        }
    }

    public void abbauenVerbindung() throws SQLException {
        try {
            DBVerbindung.getConnectionDB().close();
            DBVerbindung.getStatementsSQL().close();
        } catch (SQLException fehler) {
            fehler = new SQLException("DB konnte nicht geschlossen werden
in DBZugriff.abbauenVerbindung()");
            throw fehler;
        }
    }
}

```

```

        public DBConnection getVerbindung() throws SQLException {
            aufbaueVerbindung();
            return DBVerbindung;
        }
    }

    /*
    Die Verbindung wird charakterisiert durch 3 Zustände:
    - Datenquelle: Datei
    - Connection: Die eigentliche Anbindung
    - Statement: Möglichkeit Datenbankabfragen zu stellen
    */

    class DBConnection {
        private String datenQuelle;
        private Connection connectionDB;
        private Statement statementsSQL;

        public DBConnection() {
        }

        public Connection getConnectionDB() {
            return connectionDB;
        }

        public String getDatenQuelle() {
            return datenQuelle;
        }

        public Statement getStatementsSQL() {
            return statementsSQL;
        }

        public void setConnectionDB(Connection connectionDB) {
            this.connectionDB = connectionDB;
        }

        public void setDatenQuelle(String datenQuelle) {
            this.datenQuelle = datenQuelle;
        }

        public void setStatementsSQL(Statement statementsSQL) {
            this.statementsSQL = statementsSQL;
        }

        public ResultSet executeQuery(String sqlString) throws SQLException {
            ResultSet rs;
            try {
                rs = statementsSQL.executeQuery(sqlString);
            } catch (SQLException fehler) {
                fehler = new SQLException("Fehler in
                DBConnection.executeQuery()");
                throw fehler;
            }
            return rs;
        }
    }

```


KLAUSUR 3 SAE E3FI 21.3.2011 Zeit: 45 Minuten
SYSTEMINTEGRATOREN

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

Für Händler wird auf der Homepage der Greif-Bike-GmbH ein Zugang eingerichtet. Über eine eindeutige Zugangsnummer und Passwort wird dem Händler Zugang zum geschlossenen Bereich ermöglicht, um dort auf einen Onlineshop für Händler zuzugreifen,. Die 10-stellige Zugangsnummer wird über ein Skript auf Gültigkeit überprüft, bevor die Abfrage an die Datenbank weitergereicht wird. Hierfür wird eine Prüfziffer aus den neun ersten Ziffern errechnet.

Der Algorithmus für die Prüfung sieht folgendermaßen aus:

Nach Prüfung der Zugangsnummer auf 10 Stellen, werden die ersten neun Ziffern der Zugangsnummer von links nach rechts mit den Ziffern 2, 1, 2, 1, 2, 1, 2, 1, 2 multipliziert und aufsummiert.

Von der Summe wird für die Weiterverarbeitung nur die Einerstelle berücksichtigt.

Die Einerstelle wird von dem Wert 10 subtrahiert. Als Ergebnis erhält man die Prüfziffer (10. Stelle der Zugangsnummer).

Ergibt sich nach der Subtraktion der Rest 10, so ist die Prüfziffer 0.

Beispiele für gültige Zugangsnummern (Prüfnummer fett formatiert):

0009290701, 0539290859, 0001501825, 0001501833

Erstellen Sie ein Struktogramm für eine Funktion / Methode test, die als Parameter die Zugangsnummer (Zeichenkette) erhält und als Rückgabe einen Wert vom Typ boolean liefert und nach oben beschriebenem Verfahren als Zugangsnummer auf Gültigkeit prüft.

Stelle
Wert
Multiplikator

1	2	3	4	5	6	7	8	9	10
2	1	2	1	2	1	2	1	2	

Lösung:

Methode: test Parameter: String[] zugangsnummer		
zugangsnummer in array ziffern abspeichern summe = 0		
Zugansnummer hat 10 Stellen		back=false
w	f	
for (i=0, i<9; i++)		
i ist gerade		
w	f	
summe = summe + (int) ziffern[i]*2	summe = summe + (int) ziffern[i]	
einerstelle = summe % 10		
pruefziffer = 10 - einerstelle		
pruefziffer == 10		
w	f	
pruefziffer = 0		
ziffern[9] == pruefziffer		
back=true	back=false	
return back		