Prof. Dr. Michael Munz

Klausur zu ”Objektorientierte Softwareentwicklung” für MC/MT

Wintersemester 2016/17

Name, Vorname: _____

Matrikelnummer: _____

Semester: ____

Studiengang: **MC** ☐ **MT** ☐

- **Zugelassene Hilfsmittel:** 4 handgeschriebene Seiten (= 2 Blätter)
- Es sind keine weiteren Hilfsmittel zugelassen. Die Benutzung anderer Hilfsmittel (insbesondere von Mobiltelefonen, etc.) zählt als Täuschungsversuch
- Die Lösungen bitte in die jeweils angegebenen Lösungsbereiche schreiben.
- Falls der Platz nicht ausreicht, bekommen Sie weiteres Papier
- Auf jedes zusätzliche Blatt Papier **unbedingt Name, Matrikel-Nummer** notieren!
- Bei jeder Lösung unbedingt die Nummer der Aufgabe und Teilaufgabe angeben (außer in den Lösungsbereichen)

Dauer: 90 Minuten

Aufgabe	**1**	**2**	**3**	**4**	**Summe**
Punke max.	**20**	**10**	**10**	**10**	**(max. 50)**
Erreichte Punkte					

Aufgabe 1: OO-Techniken (20 Punkte)

Eine Anwendung zur Speicherung von Kontaktdaten für ein Smartphone soll entwickelt werden. Darin soll eine Klasse Kontakt existieren, die den Namen und verschiedene Einträge speichert: Telefonnummern (privat, geschäftlich, mobil) und E-Mail-Adressen (geschäftlich, privat).

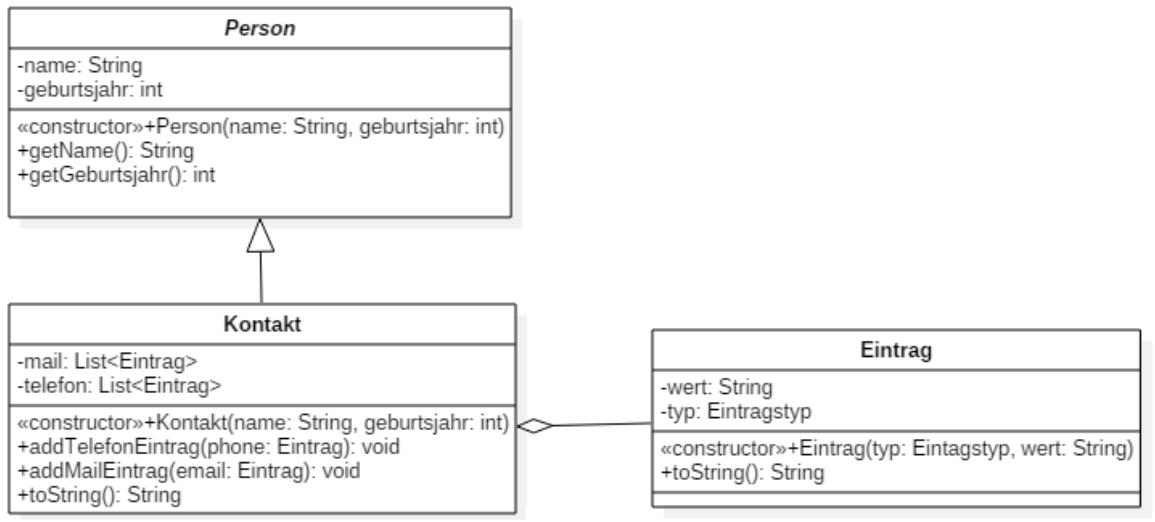
Es existiert bereits unten stehendes UML-Diagramm. Der Code der Klasse Person ist auch bereits vorhanden und unten angegeben.

```
public abstract class Person {
    private String name;
    private int geburtsjahr;

    public Person(String name, int geburtsjahr){
        this.name = name;
        this.geburtsjahr = geburtsjahr;
    }

    public String getName(){
        return this.name;
    }

    public int getGeburtsjahr(){
        return this.geburtsjahr;
    }
}
```



a) Implementieren Sie die verbleibenden Klassen Kontakt, Eintrag und den Typ Eintragstyp. Die Methode toString() der Klasse Kontakt soll einen String erzeugen, der den Namen und alle gespeicherten Einträge enthält.
Achten Sie dabei auf die **korrekte Verwendung** der **Konstruktoren** und die Deklaration der **Datentypen** inkl. entsprechender **Zugriffs-Modifizierer**.
(7 Punkte)

Zum einfacheren Versenden von E-Mails wird die Klasse EMailVerteiler erstellt. Dieser E-Mail-Verteiler soll mehrere Teilnehmer speichern, an die eine Mail geschickt werden soll. Der Mail-Verteiler soll nicht nur mit den Kontakten aus der vorliegenden Anwendung arbeiten, sondern mit Teilnehmern aller Art. Daher wird das Interface ITeilnehmer erstellt und in der Klasse EMailVerteiler verwendet.

Das Interface soll zwei Methoden vorgeben:

- **hasMail(…): Prüfung**, ob der Kontakt mindestens eine Mail-Adresse besitzt
- **sendMail(…) Senden** einer Mail, Parameter sind: Betreff (String) und Nachricht (String)

b) **Ergänzen** Sie das **UML**-Diagramm um das **Interface** ITeilnehmer mit den **sinnvollen Methodensignaturen** für oben genannte Methoden. **Zeichnen** Sie auch ein, welche Klasse(n) **dieses Interface implementieren** müssen.
(3 Punkte)

c) Geben Sie den **Quellcode** des **Interfaces** an und **implementieren** Sie das **Interface** entsprechend **sinnvoll** in der/den entsprechenden Klasse(n). In dieser Implementierung soll die Senden-Methode den Nachrichten-Text und Betreff einfach nur auf der Konsole ausgeben.
(5 Punkte)

Im unten stehenden Code-Beispiel werden diese Klasse alle verwendet. Daraus wird ersichtlich, dass die Klasse **EMailVerteiler** die folgenden **zwei Methoden** besitzen muss:

- **addTeilnehmer**: Fügt einen Teilnehmer hinzu.
- **sendNewsMail**: Versendet eine Mail mit dem Betreff „News“ und dem übergegebenen Text an die eingetragenen Empfänger. Die Mail soll jedoch nur versendet werden, falls der Empfänger eine Mail-Adresse besitzt!

Beispiel-Code:

```
Kontakt pl = new Kontakt("Peter Lustig", 1978);

pl.addMailEintrag(new Eintrag(Eintagstyp.Mail_privat,"lustig@yahoo.com"));

pl.addTelefonEintrag(new Eintrag(Eintagstyp.Telefon_mobil,"0172 662385"));
pl.addTelefonEintrag(new Eintrag(Eintagstyp.Telefon_privat,"0731 226688"));

Kontakt mm = new Kontakt("Mickey Mouse", 1928);

mm.addMailEintrag(new Eintrag(Eintagstyp.Mail_geschaeftlich,
     "mickey.mouse@disney.com"));

EMailVerteiler verteiler = new EMailVerteiler();
verteiler.addTeilnehmer(pl);
verteiler.addTeilnehmer(mm);

verteiler.sendNewsMail("Hallo an Alle!");
```

d) **Implementieren** Sie die Klasse **EMailVerteiler**. Achten Sie auf **korrekte Methodensignaturen** entsprechend obigem Code-Beispiel und **berücksichtigen** Sie die **Versandlogik** in der Methode sendNewsMail!
(5 Punkte)

Hilfreiche Methoden des Interfaces List<E> bzw. der Klasse LinkedList<E>:

- void add(E e): fügt das Element e vom Typ E hinzu
- int size(): liefert die Anzahl der Elemente in der Liste
- E get(int index): liefert das Element an Position index

Imports müssen nicht angegeben werden!

Aufgabe 2: Datenbanken (10 Punkte)

Für ein Buchungssystem eines Kinos sind folgende Datenbank-Tabellen gegeben:

Tabelle **Filme:**

ID	Titel	Filmart	FreiAb
0	Feuerwehrman Sam	3	6
1	Rogue One: A Star Wars Story	0	12
2	Resident Evil: The Final Chapter	1	16

Tabelle **Filmarten:**

ID	Filmart
0	Action
1	Horror
2	Thriller
3	Kinder
4	Drama

Tabelle **Veranstaltungen**

ID	Film	MaxPlaetze	GebuchtePlaetze	Tag	Stunde	Minute
0	0	100	80	Montag	16	0
1	1	100	100	Dientag	20	15
2	1	100	50	Dienstag	22	0
3	2	50	30	Mittwoch	20	15

Erläuterungen zu den Tabellenspalten:

Filme:

- FreiAb ist die Altersfreigabe (Film ist frei ab dem angegeben Alter)

Veranstaltungen:

- MaxPlaetze ist die Anzahl der maximal verfügbaren Plätze
- GebuchtePlaetze ist die Anzahl der aktuell bereits gebuchten Plätze
- Tag, Stunde, Minute: Termin für die Veranstaltung

a) Welche SQL-**Datentypen** und **Constraints** (NOT NULL, etc.) müssen hier für die Attribute der Tabelle „**Veranstaltungen**“ **sinnvollerweise** verwendet werden? (2 Punkte)

b) Erläutern Sie, welche **Relationstypen** (1:1, 1:n, etc.) zwischen **welchen Tabellen** aufgrund welcher **Fremdschlüssel** existieren.
(Hinweis: Fremdschlüssel ergeben sich aus der Logik und tragen nicht zwingend das Prefix „ID_“ wie in der Vorlesung!)

c) Formulieren Sie **SQL-Ausdrücke** wie folgt:
(je 2 Punkte)

- Auflisten der Filmtitel und Altersfreigabe aller Filme, die eine Person anschauen darf, die 12 Jahre alt ist, alphabetisch sortiert nach Titel.

- Auflisten der Filmtitel und der Veranstaltungstermine (Tag und Zeit) aller Filme, für die es noch freie Plätze gibt.

- Aktualisierung der Tabelle, da die Aufführung des Films „Feuerwehrman Sam“ am Montag ausverkauft ist, unter Verwendung der Film-ID und des Tags.

Aufgabe 3: Objektorientierung allgemein (10 Punkte)

Gegeben ist der folgende Quellcode:

```
public interface IC {
    String nameDerKlasse();
}
```

```
public abstract class A implements IC {
    protected int wert;
    private List<String> texte;

    public A(int wert){
        this.wert = wert;
    }

    public abstract int berechneEtwasAusWert();

    public void addText(String text){
        texte.add(text);
    }
}
```

```
public class B extends A{
    protected int wert;

    public B(){
        super(42);
    }

    @Override
    public int berechneEtwasAusWert(){
        return wert * 10;
    }

}
```

Hauptprogramm:

```
public static void main(String[] args) {
  A obj = new B();
  System.out.println(obj.berechneEtwasAusWert());
  obj.addText("Hallo");
  String name = obj.nameDerKlasse();
  obj.addText("test");
}
```

a) Beim **Compilieren** erhalten Sie eine **Fehlermeldung in der Klasse** B, dass diese nicht abstrakt sei und eine Methode nicht überschreibt. Auf **welche Methode** bezieht sich diese Fehlermeldung? Geben Sie den **Code** an, der diesen **Fehler behebt**.
(2 Punkte)

b) Beim Starten des Programms erhalten Sie eine **Nullpointer-Exception**. **Warum**? **Wie** kann dies **behoben** werden?
(2 Punkte)

c) Der **Aufruf** der Methode **berechnetEtwasAusWert(…)** auf dem Objekt obj der Klasse B liefert immer 0 (was offensichtlich falsch ist). **Korrigieren** Sie den Fehler (Quellcode).
(2 Punkte)

d) Würde das Hauptprogramm so auch funktionieren, wenn die Methode addText(…) in die Klasse B verschoben werden würde? (mit Begründung)
(2 Punkte)

e) Was müsste geändert werden, um folgenden Aufruf im Hauptprogramm realisieren zu können: `A.addText("test")`
Hinweis: Denken Sie an **alle** Folgen!
(2 Punkte)

Aufgabe 4: Multiple Choice (10 Punkte)

Bei den folgenden Multiple-Choice-Aufgaben ist jeweils **genau eine Antwort richtig.** Kreuzen Sie diese an.
(je 1 Punkt)

a) Eine Datei soll in Java mittels TextReader geöffnet werden. Diese ist aber nicht vorhanden. Was passiert?

- ☐ Es wird null zurück geliefert.
- ☐ Eine FileNotFound-Exception wird geworfen.
- ☐ Es wird ein leerer String zurück geliefert.
- ☐ Gar nichts. Beim Lesen kommt ein Fehler-Code.

b) Was wird durch die Verwendung eines BufferedReaders im Gegensatz zu einem FileReader beim Lesen von Textdateien vereinfacht?

- ☐ Es muss kein Fehlerhandling beim Dateizugriff berücksichtigt werden.
- ☐ Der Code läuft viel schneller ab.
- ☐ Die Programmierung ist einfacher, da weniger Funktionen aufgerufen werden müssen.
- ☐ Die Dateien können direkt zeilenweise gelesen werden.

c) Was haben alle Vorgehensmodelle gemeinsam?

- ☐ Alle Vorgehensmodelle wurden in der 60er Jahren entwickelt.
- ☐ Jedes Vorgehensmodell ist für jedes Projekt immer gleich sinnvoll anwendbar.
- ☐ Es existieren definierte Phasen mit Phasenergebnis und Aktivitäten.
- ☐ Jede Phase wird nur einmal durchlaufen.

d) Was passiert, wenn eine Verweisvariable an eine andere Verweisvariable gleichen Typs zugewiesen wird?

- ☐ Das Objekt wird kopiert.
- ☐ Gar nichts, da dies nicht möglich ist.
- ☐ Dies ist nur möglich, wenn die Verweisvariable zuvor auf null gesetzt wurde.
- ☐ Beide Verweise zeigen anschließend auf dasselbe Objekt.

e) Was passiert bei der Zuweisung von null an eine Verweisvariablen?

- ☐ Dies ist in Java nicht möglich.
- ☐ Das Objekt, auf das die Verweisvariable aktuell zeigt, wird sofort gelöscht.
- ☐ Das Objekt, auf das die Verweisvariable aktuell zeigt, wird vom Garbage Collector später gelöscht.
- ☐ Die Variable wird gelöscht.

f) Was passiert, wenn das Prinzip „Information Hiding“ nicht eingehalten wird?

- ☐ Es können sehr leicht semantische Programmierfehler entstehen.
- ☐ Das Programm stürzt ab.
- ☐ Es gibt einen Compilerfehler.
- ☐ Gar nichts.

g) Was wird mit Hilfe der Aggregation realisiert?

- ☐ Vererbung
- ☐ Interface-Implementierung
- ☐ Abstraktion in Form einer Teile-/Ganzes-Beziehung
- ☐ Statische Methodenaufrufe

h) Was ist die Folge von Polymorphie?

- ☐ Methoden werden mehrfach aufgerufen.
- ☐ Es wird zur Laufzeit entschieden, von welcher Klasse eine Methode aufgerufen wird.
- ☐ Es muss immer gecastet werden.
- ☐ Methoden können direkt auf einem Klassenname aufgerufen werden.

i) Das Observer-Pattern ist ein…

- ☐ Erzeugungsmuster
- ☐ Verhaltensmuster
- ☐ Strukturmuster
- ☐ Persistenzmuster

j) Für welche Art von Problemen ist die Anwendung des MVC-Patterns gedacht?

- ☐ Für das Speichern von Daten in Datenbanken
- ☐ Für die Benachrichtigung von Zustandsänderungen.
- ☐ Für die bessere Wiederverwendbarkeit von Programmteilen (Objektverbünden) auf verschiedenen Plattformen.
- ☐ Für grafische Benutzeroberflächen.