FernUniversität in Hagen

FAKULTÄT FÜR
MATHEMATIK UND INFORMATIK

Lehrgebiet Programmiersysteme
Prof. Dr. Friedrich Steimann

_____
(Name, Vorname)
_____

_____
(Straße, Nr.)

_____ _____
(PLZ) (Wohnort)

_____
(Land, falls außerhalb Deutschlands)

Kurs 1618 SS 2012

„Einführung in die objektorientierte Programmierung“

Klausur am 08.09.2012

Dauer: 3 Std., 10 – 13 Uhr

Lesen Sie zuerst die Hinweise auf der folgenden Seite!

Matrikelnummer: | | | | | | | | **Geburtsdatum:** | | | . | | . | | |

Klausurort: _____

Aufgabe	1	2	3	4	5	6	7	8	9	10	Summe
habe bear-beitet											
maximal	10	10	10	10	10	10	10	10	10	10	100
erreicht											
Korrektur											

○ Herzlichen Glückwunsch, Sie haben die Klausur bestanden. Note: .....

○ Sie haben die Klausur leider nicht bestanden. Für den nächsten Versuch wünschen wir Ihnen viel Erfolg. Die nächste Klausur findet im Wintersemester 2012/13 statt.

Hagen, den 19.9.2012

Im Auftrag

Musterlösung

Hinweise zur Bearbeitung

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst auf insgesamt 18 Seiten:
   - 1 Deckblatt
   - Diese Hinweise zur Bearbeitung
   - 10 Aufgaben auf Seite 3-16
   - Zwei zusätzliche Seiten 17 und 18 für weitere Lösungen

2. Füllen Sie jetzt bitte zuerst das Deckblatt aus:
   - Name, Vorname und Adresse
   - Matrikelnummer, Geburtsdatum und Klausurort

3. Schreiben Sie Ihre Lösungen mit Kugelschreiber oder Füllfederhalter (*kein Bleistift*) direkt in den bei den jeweiligen Aufgaben gegebenen, umrahmten Leerraum. Benutzen Sie auf keinen Fall die Rückseiten der Aufgabenblätter. Versuchen Sie, mit dem vorhandenen Platz auszukommen; Sie dürfen auch stichwortartig antworten. Sollten Sie wider Erwarten nicht mit dem vorgegebenen Platz auskommen, benutzen Sie bitte die beiden an dieser Klausur anhängenden Leerseiten. **Es werden nur Aufgaben gewertet, die sich auf dem offiziellen Klausurpapier befinden.** Eigenes Papier ist nur für Ihre persönlichen Notizen erlaubt.

4. Kreuzen Sie die bearbeiteten Aufgaben auf dem Deckblatt an. Schreiben Sie unbedingt *auf jedes Blatt* Ihrer Klausur Ihren Namen und Ihre Matrikelnummer, auf die Zusatzblätter auch die Nummer der Aufgabe.

5. Geben Sie die gesamte Klausur ab. Lösen Sie die Blätter nicht voneinander.

6. Es sind *keine Hilfsmittel* zugelassen.

7. Lesen Sie vor der Bearbeitung einer Aufgabe den *gesamten* Aufgabentext sorgfältig durch.

8. Es sind maximal 100 Punkte erreichbar. Wenn Sie mindestens 50 Punkte erreichen, haben Sie die Klausur bestanden.

9. Sie erhalten die korrigierte Klausur zurück, zusammen mit einer Bescheinigung für das Finanzamt und ggf. dem Übungsschein.

10. Legen Sie jetzt noch Ihren Studierendenausweis und einen amtlichen Lichtbildausweis bereit, dann kann die Arbeit beginnen. Viel Erfolg!

Aufgabe 1: Aliase (10 Punkte)

Gegeben sei folgendes Java-Programm:

```
class Guthabenkarte {
    private int guthaben = 0;

    public void aufladen(int betrag) {
        guthaben = guthaben + betrag;
    }

    public void abheben(int betrag) {
        if (betrag > guthaben)
            System.out.println("Nicht genuegend Guthaben!");
        else {
            System.out.println(betrag + " Euro abgehoben.");
            guthaben = guthaben - betrag;
        }
    }
}

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

        Guthabenkarte leereKarte = new Guthabenkarte();

        Guthabenkarte tanjasKarte = leereKarte;
        Guthabenkarte danielasKarte = leereKarte;

        danielasKarte.aufladen(10);
        danielasKarte.abheben(5);

        tanjasKarte.aufladen(10);
        tanjasKarte.abheben(4);
        tanjasKarte.abheben(7);
    }
}
```

Welche Ausgabe erzeugt die `main`-Methode? Erklären Sie diese!

Die Ausgabe lautet

```
5 Euro abgehoben.
4 Euro abgehoben.
7 Euro abgehoben.
```

Es kommt insbesondere nicht zu einer Meldung, dass das Guthaben nicht ausreicht, da sowohl die Variable `tanjasKarte` als auch `danielasKarte` beide dasselbe `Guthabenkarte`-Objekt referenzieren, welches mit zweimal 10 Euro ausreichend aufgeladen wurde, um die insgesamt 16 Euro abzuheben.

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 1)

Geben Sie eine Methode `duplizieren` an, welche die Klasse `Guthabenkarte` derart erweitert, dass eine Kopie einer Guthabenkarte mit gleichem Zustand, aber geänderter Identität angelegt wird. Die Benutzung der Methode soll z.B. wie folgt möglich sein:

```
Guthabenkarte tanjasKarte = leereKarte.duplizieren();
```

```
public Guthabenkarte duplizieren() {
    Guthabenkarte klon = new Guthabenkarte();
    klon.guthaben = this.guthaben;
    return klon;
}
```

Erklären Sie anhand eines kurzen Programmbeispiels, wie Sie in Java zwei Objekte auf gleichen Zustand bzw. Identität hin untersuchen können. Was ist der Unterschied zwischen gleichem Zustand und Identität?

```
Object o1 = new Object();
Object o2 = new Object();
boolean identisch = o1 == o2;
boolean zustandsgleich = o1.equals(o2);
```

Identität kann in Java mit dem ==-Operator geprüft werden. Sie ist genau dann erfüllt, wenn zwei Referenzen exakt dasselbe Objekt referenzieren. Ob zwei Objekte zustandsgleich sind, entscheidet die Klasse selbst (bzw. die Programmiererin/der Programmierer), typischerweise mit einer Implementierung der von `Object` geerbten `equals`-Methode. In der Regel sollte diese Methode identische Objekte auch als zustandsgleich klassifizieren.

Aufgabe 2: Klassenentwurf (10 Punkte)

Erstellen Sie eine Klasse „Bruch“, die die folgenden Vorgaben umsetzt:

- Ein Bruch hat die ganzzahligen Attribute `zaehler` und `nenner`.
- Die Werte der beiden Attribute werden bei der Erzeugung eines `Bruch`-Exemplars übergeben und lassen sich danach nicht mehr ändern.
- Die Attribute sind öffentlich zugreifbar.
- In der Klasse Bruch ist die von `Object` geerbte Methode `toString()` so zu überschreiben, dass sie eine sinnvolle Information über den Bruch liefert.
- `Bruch`-Exemplare haben eine Methode `multipliziere`, deren einziger Parameter ein weiteres `Bruch`-Exemplar ist. Die Methode gibt ein neues `Bruch`-Exemplar zurück, welches das Produkt der beiden Brüche (Nachrichtenempfänger und Parameter) repräsentiert.

```
public class Bruch {

    public final int zaehler;
    public final int nenner;

    public Bruch(int zaehler, int nenner) {
        this.zaehler = zaehler;
        this.nenner = nenner;
    }

    public Bruch muliplizieren(Bruch faktor) {
        int neuerZaehler = zaehler * faktor.zaehler;
        int neuerNenner = nenner * faktor.nenner;
        return new Bruch(neuerZaehler, neuerNenner);
    }

    public String toString() {
        return zaehler + "/" + nenner;
    }
}
```

Aufgabe 3: Verschiedenes (10 Punkte)

Geben Sie an, ob die folgenden Aussagen korrekt sind. Alle Aussagen beziehen sich auf fehlerfrei kompilierende Java-Programme.

Achtung! Falsche Antworten geben Minuspunkte, eine gesamt negative Punktzahl bei dieser Aufgabe wird Ihnen aber auf null Punkte aufgerundet.

Das Schlüsselwort `this` darf unmittelbar innerhalb einer statischen Methode nicht verwendet werden.	☑ Wahr	☐ Falsch
Das Schlüsselwort `this` darf unmittelbar innerhalb eines Konstruktors nicht verwendet werden.	☐ Wahr	☑ Falsch
Eine abstrakte Klasse braucht nicht alle Methode zu implementieren, die durch die von ihr implementierten Interfaces vorgegeben sind.	☑ Wahr	☐ Falsch
Alle Methoden eines Interface sind (gegebenenfalls implizit) `public`.	☑ Wahr	☐ Falsch
Auf als `protected` deklarierte Attribute und Methoden darf ausschließlich aus demselben Typen und den Subtypen des jeweiligen Typs zugegriffen werden.	☐ Wahr	☑ Falsch
Der Bytecode eines kompilierten Java-Programmes ist unabhängig von der jeweiligen Rechnerarchitektur und des Betriebssystems, auf welchem das Programm laufen kann.	☑ Wahr	☐ Falsch
Die virtuelle Maschine ist unabhängig von der jeweiligen Rechnerarchitektur und des Betriebssystems, auf welchem sie läuft.	☐ Wahr	☑ Falsch
Ein als final deklariertes Attribut referenziert nach der erstmaligen Initialisierung stets über die gesamte Programmdauer hinweg dasselbe Objekt bzw. denselben Wert.	☑ Wahr	☐ Falsch
Eine als final deklarierte Methode liefert nach der ersten Ausführung stets über die gesamte Programmdauer hinweg denselben Wert bzw. dasselbe Objekt als Rückgabewert.	☐ Wahr	☑ Falsch
Weder eine abstrakte Klasse noch ein Interface lässt sich instanzieren.	☑ Wahr	☐ Falsch

Aufgabe 4: Überschreiben und Überladen (10 Punkte)

Gegeben sei folgendes Java-Programm:

```
class Person {}
class Erwachsener extends Person {}
class Kind extends Person {}

class Fahrzeug {
    public void fahre(Person person, Person person2) {
        System.out.println("Person und Person in Fahrzeug");
    }
    public void fahre(Person person, Erwachsener erwachsener) {
        System.out.println("Person und Erwachsener in Fahrzeug");
    }
}

class Auto extends Fahrzeug {
    public void fahre(Person person, Erwachsener erwachsener) {
        System.out.println("Person und Erwachsener in Auto");
    }
    public void fahre(Kind Kind, Erwachsener erwachsener) {
        System.out.println("Kind und Erwachsener in Auto");
    }
}

class Boot extends Fahrzeug {
    public void fahre(Person person, Erwachsener erwachsener) {
        System.out.println("Person und Erwachsener in Boot");
    }
    public void fahre(Kind Kind, Erwachsener erwachsener) {
        System.out.println("Kind und Erwachsener in Boot");
    }
}

class Gummiboot extends Boot {
    public void fahre(Person person, Erwachsener erwachsener) {
        System.out.println("Person und Erwachsener in Gummiboot");
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        Erwachsener erwachsener = new Erwachsener();
        Kind kind = new Kind();

        Fahrzeug fahrzeug1 = new Auto();
        Fahrzeug fahrzeug2 = new Gummiboot();
        Boot boot = new Gummiboot();

        fahrzeug1.fahre(kind, erwachsener);
        fahrzeug1.fahre(person, erwachsener);
        fahrzeug2.fahre(kind, kind);
        boot.fahre(erwachsener, erwachsener);
        boot.fahre(kind, erwachsener);
    }
}
```

(Fortsetzung der Aufgabe auf folgender Seite)

(Fortsetzung von Aufgabe 4)

Geben Sie für jeden der angegebenen Ausdrücke an, welche Ausgabe dieser zur Folge hat. Begründen Sie Ihre Antwort!

```
fahrzeug1.fahre(kind, erwachsener);
```

```
Person und Erwachsener in Auto
```

Der Compiler bindet diesen Aufruf an `Fahrzeug#fahre(Person, Erwachsener)`. Zur Laufzeit wird dann klar, dass es sich beim Nachrichtenempfänger um ein `Auto` handelt, welches diese Methode überschreibt.

```
fahrzeug1.fahre(person, erwachsener);
```

```
Person und Erwachsener in Auto
```

Der Compiler bindet diesen Aufruf an `Fahrzeug#fahre(Person, Erwachsener)`. Zur Laufzeit wird dann klar, dass es sich beim Nachrichtenempfänger um ein `Auto` handelt, welches diese Methode überschreibt.

```
fahrzeug2.fahre(kind, kind);
```

```
Person und Person in Fahrzeug
```

Der Compiler bindet diesen Aufruf an `Fahrzeug#fahre(Person, Person)`. Zur Laufzeit wird dann klar, dass es sich beim Nachrichtenempfänger um ein `Gummiboot` handelt, welches diese Methode allerdings nicht überschreibt.

```
boot.fahre(erwachsener, erwachsener);
```

```
Person und Erwachsener in Gummiboot
```

Der Compiler bindet diesen Aufruf an `Boot#fahre(Person, Erwachsener)`. Zur Laufzeit wird dann klar, dass es sich beim Nachrichtenempfänger um ein `Gummiboot` handelt, welches diese Methode überschreibt.

```
boot.fahre(kind, erwachsener);
```

```
Kind und Erwachsener in Boot
```

Der Compiler bindet diesen Aufruf an `Boot#fahre(Kind, Erwachsener)`. Zur Laufzeit wird dann klar, dass es sich beim Nachrichtenempfänger um ein `Gummiboot` handelt, welches diese Methode aber nicht überschreibt.

Aufgabe 5: Beschränkt parametrische Typen (10 Punkte)

Gegeben sei folgendes Interface `Messbar`

```
interface Messbar {
    public int groesse();
}
```

Implementieren Sie eine Klasse `MaxFilter`, die die folgenden Bedingungen erfüllt:

- `MaxFilter` ist typparametrisierbar, wobei für den einzigen Typparameter nur Subtypen von `Messbar` Verwendung finden dürfen.
- `MaxFilter` besitzt eine Methode `hinzufuegen`, welche genau ein Argument erwartet und welches vom angegebenen parametrisierten Typen ist.
- `MaxFilter` hat ein Attribut `maxElement`, welches das größte, bisher über `hinzufuegen` übergebene Element referenziert. Bei gleichgroßen Elementen referenziert `maxElement` das größte zuletzt hinzugefügte Element.
- Eine Methode `getMaxElement` gibt eine Referenz auf das von `maxElement` referenzierte Objekt zurück.
- Hinweis: beim ersten Aufruf von `hinzufuegen` gibt es eventuell ein zusätzliches Problem zu beachten.

```
class MaxFilter<T extends Messbar> {
    private T maxElement;

    public void hinzufuegen(T element) {
        if (maxElement == null || maxElement.groesse() <= element.groesse())
            maxElement = element;
    }

    public T getMaxElement() {
        return maxElement;
}
```

Aufgabe 6: Ausnahmebehandlung (10 Punkte)

Gegeben sei folgende Klasse `Box`, welche das Einlagern und das Entnehmen einen Strings erlaubt. `Box` kennt zwei mögliche Fehlerfälle: das Entnehmen eines Strings aus einer leeren Box sowie das Einfügen eines Strings in eine volle Box.

```
class Box {
    String inhalt;

    public void einlagern(String inhalt) {
        if(this.inhalt != null)
            System.out.println("Fehler, die Box war nicht leer!");
        this.inhalt = inhalt;
    }

    public String entnehmen(){
        if(inhalt == null)
            System.out.println("Fehler, die Box ist leer!");
        String entnommen = inhalt;
        inhalt = null;
        return entnommen;
    }

    public static void main(String[] args) {
        Box box = new Box();
        box.einlagern("abc");
        System.out.println(box.entnehmen());
    }
}
```

Nun ist es kein guter Programmierstil, die Fehlerbehandlung durch die Box vornehmen zu lassen, da nicht die Box selbst, sondern höchstens ihre Verwender die Fehler Auslösen und entsprechend auch behandeln sollten.

Implementieren Sie die Box noch einmal neu, dieses Mal unter Verwendung von Exceptions. Beim Entnehmen eines Strings aus einer leeren Box soll dabei eine `BoxIstLeerException` und beim Einfügen eines Strings in eine volle Box soll eine `BoxIstVollException` geworfen werden. Denken Sie auch daran, diese beiden Exception-Typen zu deklarieren sowie die main-Methode so anzupassen, dass etwaige Fehlermeldungen in der Main-Methode ausgegeben werden.

(Platz für Ihre Lösung finden Sie auf der kommenden Seite)

(Fortsetzung von Aufgabe 6)

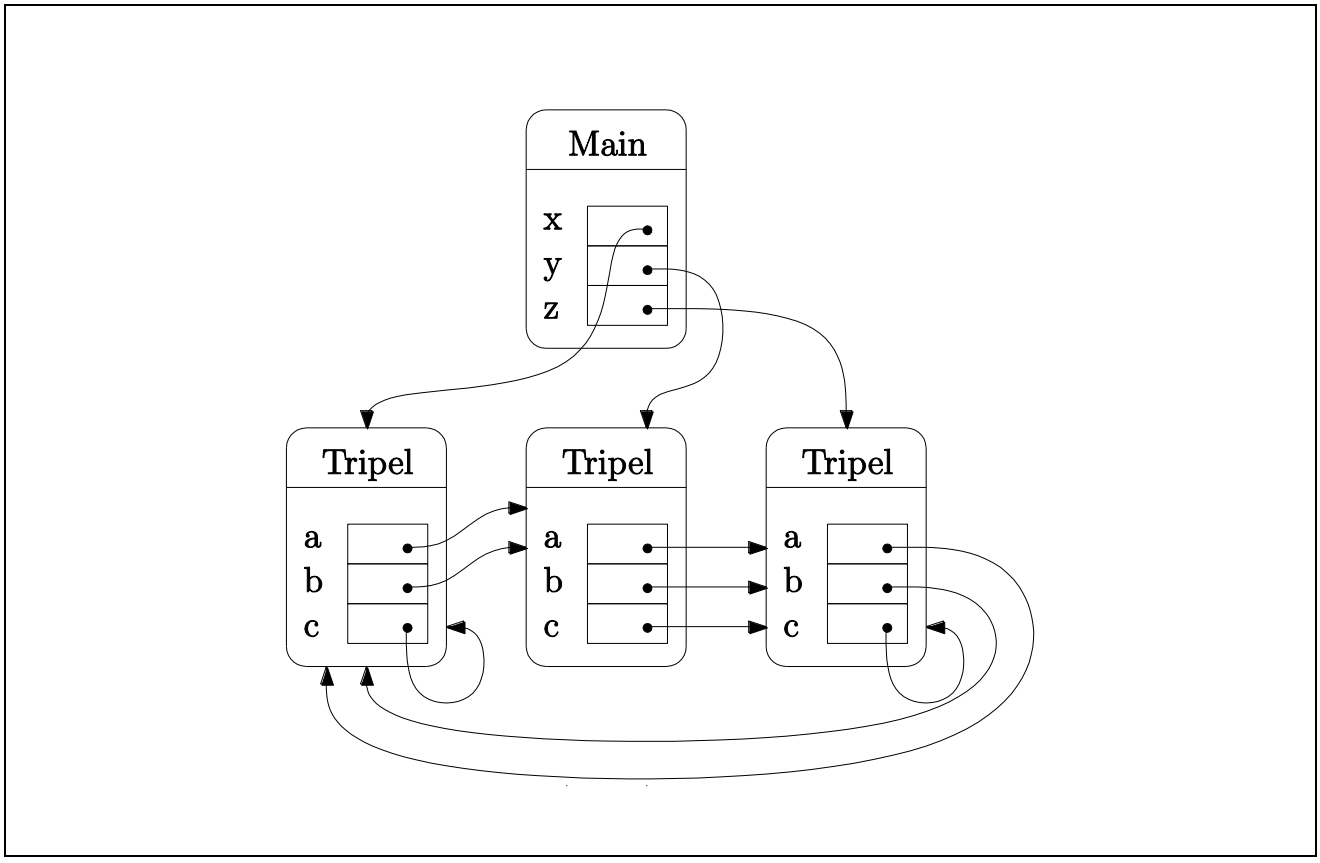
```
class BoxIstLeerException extends Exception {}
class BoxIstVollException extends Exception {}

class Box {
    String inhalt;

    public void einlagern(String inhalt) throws BoxIstVollException {
        if (this.inhalt != null)
            throw new BoxIstVollException();
        this.inhalt = inhalt;
    }

    public String entnehmen() throws BoxIstLeerException {
        if (inhalt == null)
            throw new BoxIstLeerException();
        String entnommen = inhalt;
        inhalt = null;
        return entnommen;
    }

    public static void main(String[] args) {
        Box box = new Box();
        try {
            box.einlagern("abc");
        } catch (BoxIstVollException e) {
            System.out.println("Fehler, die Box war nicht leer!");
        }
        try {
            System.out.println(box.entnehmen());
        } catch (BoxIstLeerException e) {
            System.out.println("Fehler, die Box ist leer!");
        }
    }
}
```

Aufgabe 7: Objektgeflechte & Serialisierung (10 Punkte)

Zeichnen Sie das Objektgeflecht, das bei Beendigung der Main-Methode entstanden ist.

```
class Tripel {
    Tripel a = null;
    Tripel b = null;
    Tripel c = null;
}
```

```
class Main {
    Tripel x = new Tripel();
    Tripel y = new Tripel();
    Tripel z = new Tripel();
    void verbinde() {
        x.a = x.b = z.a = y;
        y.a = y.b = y.c = z.c = z;
        z.a = z.b = x.c = x;
    }
     public static void main(String[] args) {
        new Main().verbinde();
    }
}
```



Geben Sie **einen** Grund an, warum Serialisierung eine nichttriviale Aufgabe ist. Erläutern Sie dies anhand Ihrer Zeichnung.

Bei der Serialisierung eines Objektgeflechts müssen Zyklen erkannt werden, damit jedes Objekt nur einmal serialisiert wird. In obigem Beispiel referenziert `x.a` das Objekt `y`, `y.a` referenziert `z`, `z.a` referenziert `x`. `x` soll aber kein zweites Mal serialisiert werden.

Aufgabe 8: Verteilte Systeme

Der Kurstext nennt **vier** verschiedene Kommunikationsmittel für verteilte Systeme, wovon zwei primär zur asynchronen und zwei primär zur synchronen Kommunikation eingesetzt werden. Nennen Sie diese und ordnen Sie sie der asynchronen bzw. synchronen Kommunikation zu.

Asynchrone Kommunikation:

- Kommunikation über gemeinsamen Speicher
- Kommunikation über Nachrichten

Synchrone Kommunikation:

- Entfernter Prozedur- bzw. Methodenaufruf
- Spezielle Kommunikationskonstrukte

Nennen Sie **zwei** Gründe, warum sich das objektorientierte Grundmodell besonders gut für die Programmierung verteilter Systeme eignet.

- Objekte fassen Daten und Operationen zu Einheiten mit klar definierten Schnittstellen zusammen; Objekte bzw. Klassen bilden demnach weitgehend unabhängige Programmteile, die sich auf verschiedene Prozesse bzw. Rechner verteilen lassen.

- Kommunikation zwischen Objekten über Nachrichten ist integraler Bestandteil des objektorientierten Grundmodells, sodass bereits alle wesentlichen Sprachmittel für die Kommunikation im Rahmen verteilter Systeme zur Verfügung stehen.

Aufgabe 9: Objektorientierte Modellierung (10 Punkte)

In Java kann eine Klasse stets nur maximal eine weitere, direkte Superklasse haben. Wie heißt die Funktionalität, die Java somit fehlt?

Mehrfachvererbung.

Welches Konstrukt bietet Java an, dennoch eine Klasse mit mehr als einem direkten Supertypen auszustatten?

Interfaces.

Nutzen Sie dieses Konstrukt mindestens einmal und implementieren Sie fünf Typen `Touchscreen`, `Maus`, `Klavier`, `MitTasten` und `Eingabegeraet`, so dass die folgenden Bedingungen erfüllt sind:

- `Touchscreen` ist Subtyp von `Eingabegeraet`
- `Maus` ist Subtyp von `Eingabegeraet`
- `Maus` ist Subtyp von `MitTasten`
- `Klavier` ist Subtyp von `MitTasten`
- `Touchscreen` ist **kein** Subtyp von `MitTasten`
- `Klavier` ist **kein** Subtyp von `Eingabegeraet`
- Es kann **keine** Instanzen von `Eingabegeraet` und von `MitTasten` geben, aber von jedem der übrigen Typen.

Weiterhin sollen der Typ `MitTasten` eine Methode `public int anzahlTasten()` und der Typ `Eingabegeraet` eine Methode `public String geraeteArt()` bekommen, die geeignet zu implementieren sind. (Sie dürfen bei der Anzahl der Tasten schätzen.)

(Platz für Ihre Lösung finden Sie auf der kommenden Seite)

(Fortsetzung von Aufgabe 9)

```
abstract class Eingabegeraet {
   public abstract String geraeteArt();
}

class Touchscreen extends Eingabegeraet {
   public String geraeteArt() {
      return "Touchscreen";
   }
}

interface MitTasten {
   public int anzahlTasten();
}

class Maus extends Eingabegeraet implements MitTasten {
   public String geraeteArt() {
      return "Maus";
   }
   public int anzahlTasten() {
      return 3;
   }
}

class Klavier implements MitTasten {
   public int anzahlTasten() {
      return 88;
   }
}
```

Aufgabe 10: Kleine Fehlersuche

Zu folgendem Programm meldet der Java-Compiler fünf Compilerfehler.

```
public abstract class C {
    int i;
    private static int j;

    public static void main(String[] args) {
        C c = new C();
        D d = c;
        c = d;
        i = 3;
        j = 3;
    }
    abstract void m();
}

class D extends C {
    int k = j;
}
```

Finden **und begründen** Sie die fünf Fehler anhand der Hinweise:

Compilerfehler zu statischen Deklarationen	Das nicht-statische Feld `i` darf nicht innerhalb der statischen Methode `main` referenziert werden.
Compilerfehler zu Subtyping	In der Zuweisung `d = c` darf nicht ein Supertyp einem Subtypen zugewiesen werden.
Compilerfehler zu abstrakten Klassen	Der abstrakte Typ `C` kann nicht durch `new C()` instanziert werden.
Noch ein Compilerfehler zu abstrakten Klassen	Die nicht-abstrakte Klasse `D` muss die abstrakte Methode `m` aus `C` implementieren.
Compilerfehler zu Sichtbarkeiten	In `D` darf nicht auf das private Feld `j` zugegriffen werden.

Zusätzlicher Platz für Ihre Lösungen

Ergänzung zu Aufgabe Nr.

Ergänzung zu Aufgabe Nr.

Zusätzlicher Platz für Ihre Lösungen

Ergänzung zu Aufgabe Nr.

Ergänzung zu Aufgabe Nr.