

1. Aufgabe

Betrachten Sie die Klasse `Student` im Lehre-Ordner (/Uebung7). Die Klasse ist mit Konstanten ausgestattet, die das jeweilige Studienfach des Studenten repräsentieren. Ändern Sie die Klasse `Student` nun so, dass statt der Deklaration einzelner Konstanten innerhalb der Klasse `Student` ein eigener Aufzählungstyp `Fach` verwendet wird, der die notwendigen **enum**-Objekte enthält. Eine Klasse `StudentenTest` soll nun statt

```
public class StudentenTest {  
    public static void main(String[] args) {  
        Student Peter = new Student();  
        Peter.setName("Peter Honig");  
        Peter.setMatnr(12345);  
        Peter.setFach(Student.WIRTSCHAFTLICHESSTUDIUM);  
        System.out.println(Peter);  
    }  
}
```

so aussehen

```
public class StudentenTest {  
    public static void main(String[] args) {  
        Student Peter = new Student();  
        Peter.setName("Peter Honig");  
        Peter.setMatnr(12345);  
        Peter.setFach(WIRTSCHAFTLICHESSTUDIUM);  
        System.out.println(Peter);  
    }  
}
```

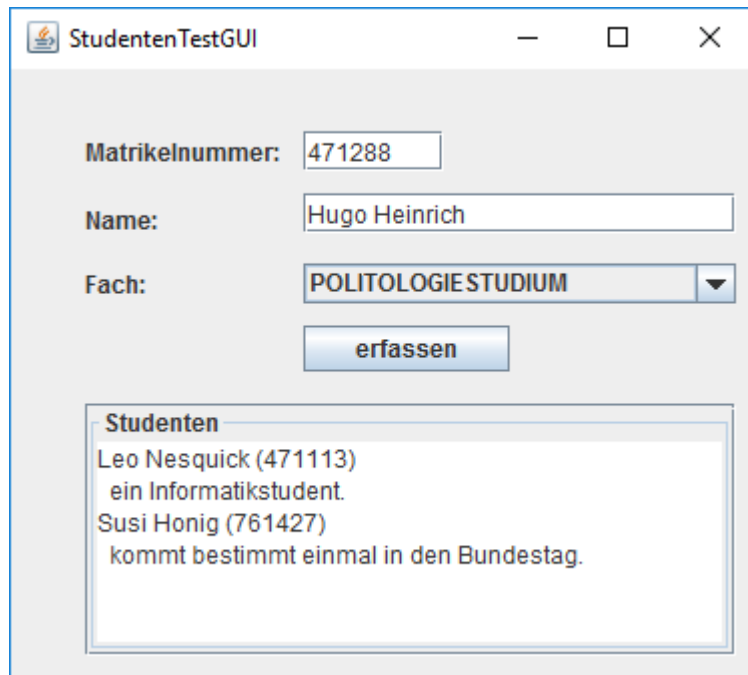
1.1 Fortsetzung 1

Erweitern Sie den Aufzählungstyp `Fach` um eine Instanz-Methode `regelstudienzeit()`, die zu jedem Studienfach die Regelstudienzeit liefert (für Politologie und Architektur 8 Semester, für alle übrigen 7 Semester). Danach sollte folgendes Code-Fragment in `StudentenTest` laufen:

```
public static void main(String[] args) {  
    Student Peter = new Student();  
    Peter.setName("Peter Honig");  
    Peter.setMatnr(12345);  
    Peter.setFach(WIRTSCHAFTLICHESSTUDIUM);  
    System.out.println(Peter);  
    System.out.println("Regelstudienzeit für sein Studium: " +  
        Peter.getFach().regelstudienzeit() + " Semester. ");  
}
```

1.2 Fortsetzung 2

Legen Sie eine `StudentenTestGUI` zum Erfassen von Studenten an - mit Combo-Box zum auswählen des Faches. Platzieren Sie diese GUI-Klasse in ein anderes Package und importieren Sie die anderen Ressourcen. Skizze:



The screenshot shows a Java Swing window titled "StudentenTestGUI". It contains the following elements:

- Matrikelnummer:** A text input field containing "471288".
- Name:** A text input field containing "Hugo Heinrich".
- Fach:** A dropdown menu (Combo-Box) with "POLITOLOGIESTUDIUM" selected.
- erfassen:** A button to submit the data.
- Studenten:** A scrollable text area displaying a list of students:
 - Leo Nesquick (471113)
ein Informatikstudent.
 - Susi Honig (761427)
kommt bestimmt einmal in den Bundestag.

2. Aufgabe

Gegeben seien die folgenden Klassen:

```
public class Tier {}  
public class Hund extends Tier {}  
public class Vogel extends Tier {}  
public class Kaefig<T> {  
    private T einTier;  
    public void setTier(T x) {  
        einTier = x;  
    }  
    public T getTier() {  
        return einTier;  
    }  
}
```

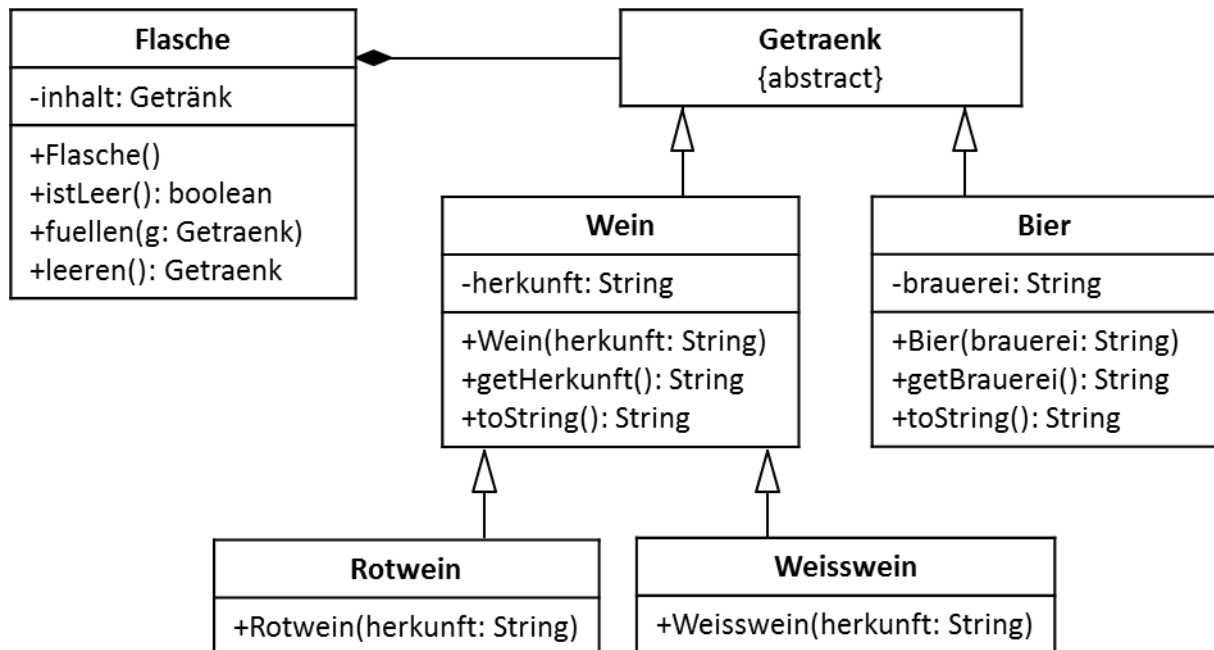
Versuchen Sie erst zu beschreiben (und testen Sie dann aus) was mit folgenden Code geschieht. Es gibt die Möglichkeiten:

- er übersetzt nicht
- er übersetzt mit Warnungen
- er erzeugt Fehler während der Ausführung
- er übersetzt und läuft ohne Probleme

- a) `Kaefig<Tier> zwinger = new Kaefig<Hund>();`
- b) `Kaefig<Vogel> voliere = new Kaefig<Tier>();`
- c) `Kaefig<?> voliere = new Kaefig<Vogel>();`
`voliere.setTier(new Vogel());`
- d) `Kaefig voliere = new Kaefig();`
`voliere.setTier(new Vogel());`

3. Aufgabe

In diesem Beispiel werden Flaschen mit verschiedenen Getränken gefüllt und entleert. Das Befüllen und Entleeren erfolgt in der `main()` Methode der Klasse `Flaschenverwaltung`. Die vorliegende Implementierung erlaubt das Befüllen der Flaschen `f1` und `f2` mit beliebigen Getränken. Die Klasse `Flaschenverwaltung` finden Sie im Lehreordner.



Ändern Sie die Implementierung der Klasse `Flasche` in eine generische Klasse, die für unterschiedliche Getränke parametrisiert werden kann.

- Passen Sie alle Methoden der Klasse `Flasche` so an, dass sie mit einem parametrisierten Getränk befüllt werden können (z.B. Bier oder Wein)
- Ändern Sie die `main()` Methode derart, dass `f1` nur mit Bier befüllt werden kann und `f2` nur mit Wein.

Klassen:

```

public class Rotwein extends Wein {
    public Rotwein(String h) {super(h);}
}
public class Weisswein extends Wein {
    public Weisswein(String h) {super(h);}
}
public abstract class Getraenk {
}
public class Wein extends Getraenk {
    private String herkunft;

    public Wein(String origin) {
        herkunft = origin;
    }
    public String getHerkunft() {
        return herkunft;
    }
    public String toString() {
        return ("Wein aus " + herkunft);
    }
}
public class Bier extends Getraenk {
    private String brauerei;

    public Bier(String b) {
        brauerei = b;
    }
    public String getBrauererei() {
        return brauerei;
    }
    public String toString() {
        return "Bier von " + brauerei;
    }
}
public class Flasche {
    Getraenk inhalt = null;
    public boolean istLeer() {
        return (inhalt == null);
    }
    public void fuellen(Getraenk g) {
        inhalt = g;
    }
    public Getraenk leeren() {
        Getraenk result = inhalt;
        inhalt = null;
        return result;
    }
}
    
```

4. Aufgabe

Welche Zeilen in der main() Methode werden vom Übersetzer nicht übersetzt?

Markieren Sie die Zeilen (bzw. kommentieren Sie diese aus) und nennen Sie den Fehler:

```
public class KoordinateTest<T extends Number> {
    public T x;
    public T y;
    public KoordinateTest(T xp, T yp) {
        x = xp;
        y = yp;
    }
    public static void main(String[] args) {
        KoordinateTest<Double> k11, k12;
        KoordinateTest<Integer> k21, k22;
        KoordinateTest<String> k31;
        KoordinateTest<Number> k41, k42;

        k11 = new KoordinateTest<Float>(2.2d, 3.3d);
        k12 = new KoordinateTest<Double>(2.2d, 3.3d);
        k21 = new KoordinateTest<Integer>(2, 3);
        k31 = new KoordinateTest<String>("11", "22");
        k41 = new KoordinateTest<Number>(21, 31);

        k41 = new KoordinateTest<Number>(4.4d, 5.5f);
        k11 = new KoordinateTest<Double>(3.3f, 9.9d);

        KoordinateTest<? super Double> k99;
        k99 = k11;
        k99 = k41;
        k99 = k31;

        k11 = k12;
        k12 = k21;
        KoordinateTest k55 = new KoordinateTest<Number>(7.7f, 8.8f);
        KoordinateTest k66 = new KoordinateTest(7.7f, 8.8f);
    }
}
```

5. Aufgabe

Betrachten Sie folgende Klasse `Liste` mit dem „Raw-Type“ Object:

```
public class Liste {
    private Object[] data;
    private int count;
    public Liste(int size){
        data = new Object[size];
    }
    public void add(Object x){
        if(count == data.length){
            Object[] neu = new Object[data.length*2];
            for(int i=0; i<data.length; i++) neu[i] = data[i];
            data = neu;
        }
        data[count] = x;
        count++;
    }
    public Object remove(){
        if(count > 0){
            count--;
            return data[count];
        }
        else return null;
    }
    public int getCount(){return count;}
}
```

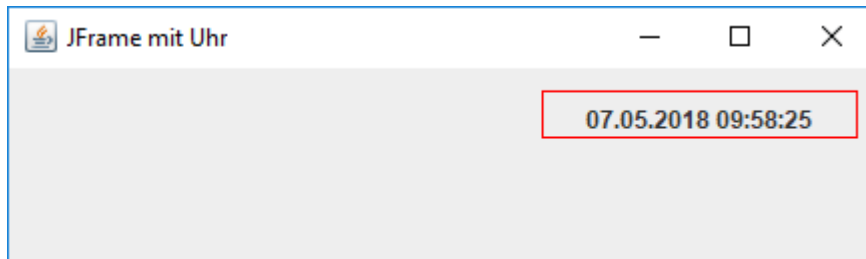
Aufgaben:

- Analysieren Sie was diese Klasse macht.
- Erstellen Sie eine GUI, die ein Objekt der Klasse `Liste` mit einer Anfangsgröße von 3 anlegt. Diese soll beliebige ganze Zahlen in die Liste speichern können (Operation `add`). Per Befehl sollen die Zahlen aus der Liste zurückgeholt (Operation `remove`) und deren Summe ausgegeben werden.
- Wandeln Sie die Klasse so um, dass Sie auch typsicher (generisch) verwendet werden kann. Erstellen Sie eine weitere GUI, welche das gleiche wie in b) macht, nun aber mit einer generischen Liste.

Loggen Sie die Eingabe (erfassen) zur Kontrolle über eine TextArea mit. Der Inhalt der TextArea wird nach jedem „erstellen“ beim ersten „erfassen“ wieder gelöscht. Skizze:

6. Aufgabe

Erstellen Sie eine Klasse DigitalUhr als Unterklasse von JPanel, die das Interface Runnable implementiert. Ein Objekt dieser Klasse soll z.B. in einem JFrame platziert werden können um aktuelles Datum und die laufende Uhrzeit sekundengenau anzuzeigen. Siehe Beispiel.



7. Aufgabe

Erstellen Sie eine Anwendung, die einen Gaststättenspielautomaten (in vereinfachter Form) simuliert. Es gibt drei Anzeigen, zunächst mit 0 vorbelegt, die durch Druck der darunter liegenden Taste gestartet werden können. Auf jeder Anzeige werden nun zufällige Ziffern zwischen 0 und 9 in dynamischer Folge erzeugt. Man kann nun sein Glück versuchen um alle Anzeigen bei gleichen Wert anzuhalten. Die Funktionalität im Detail:

- Wird ein start-Button gedrückt wechselt die Ziffer in der zugehörigen Anzeige nach jeweils einer 1/10 Sekunde. Der Button wird zum stop-Button.
- Wird ein stop-Button gedrückt stoppt der Anzeigenwechsel und die letzte erzeugte Ziffer bleibt stehen. Der Button wird wieder zum start-Button.

Zur Anschauung gibt es ein ausführbares jar-File der Anwendung im Lehre-Ordner.

