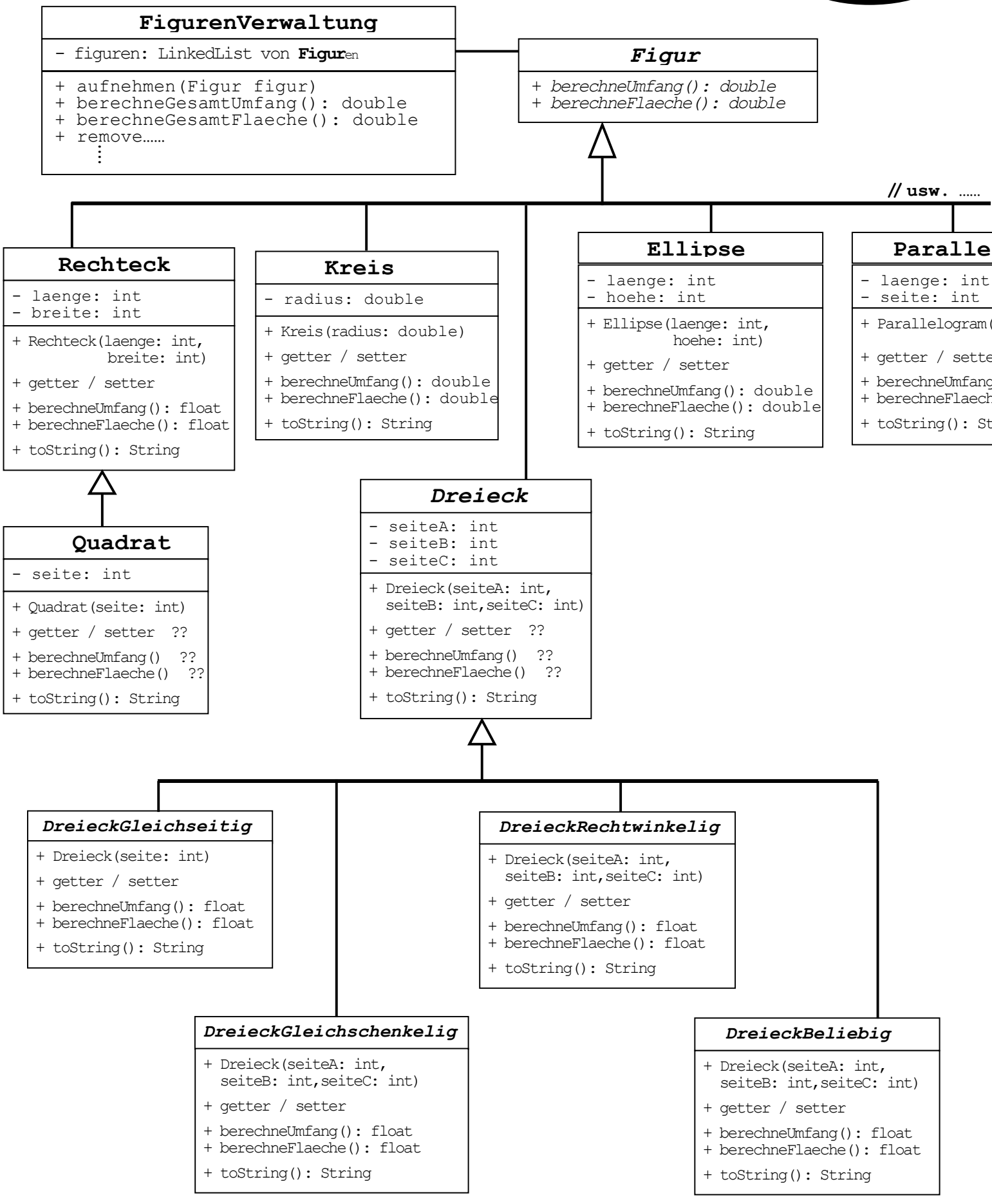
Implementieren Sie die im nachfolgenden UML-Diagramm beschriebenen Klassen und testen Sie diese in jeweils eigens dafür erstellten Test-Klassen:

FOO



Hinweis:

Implementieren Sie **Test-**Klassen, mit denen Sie die neuen Klassen **laufend** testen!!

Aufgabe 1:

Erstellen Sie eine Klasse **Figur**, die als Basisklasse für die weiter unten beschriebenen Klassen **Kreis**, **Rechteck**, **Ellipse**, **Dreieck**, **Parallelogramm** usw. dienen soll.
Diese Klasse **Figur** soll die beiden Methoden ***berechneFlaeche()*** sowie ***berechneUmfang()*** nur vorschreiben (weil ja nicht bekannt sein kann, welche geometrische Figur sich davon ableiten wird, welche Attribute diese haben könnte und wie sich deren Fläche bzw. Umfang berechnet).
Da einige der davon ableitenden Klassen zur Flächenberechnung die Zahl **PI** benötigen, kann diese als Konstante mit dem Näherungswert **3.14159** festgelegt werden.

Aufgabe 2:

Erstellen Sie die Klassen **Rechteck** und **Kreis** unter Verwendung der Klasse **Figur**. Die Konstruktoren der beiden Klassen erhalten die notwendigen Bestimungsgrößen (Radius beim **Kreis**, Länge und Breite beim **Rechteck**) als Parameter übergeben.
Implementieren Sie die von **Figur** vorgeschriebenen Methoden so, daß sie jeweils die diesen geometrischen Figuren entsprechenden Werte zurückliefern (z.B. Kreis-Umfang 2*radius*PI, für ein Rechteck aber 2*(laenge+breite)).
Implementieren Sie auch jeweils eine entsprechende ***toString()***-Methode, welche (mit kurzem Text) die wichtigsten Informationen über das Objekt enthält (Art der Figur und alle Attribut-Werte).

Aufgabe 3:

Leiten Sie von **Rechteck** die Klasse **Quadrat** ab (ein Quadrat ist ein spezielles Rechteck!, bei welchem Länge und Breite den gleichen Wert haben!).
Konzipieren Sie die Klasse **Quadrat** so, dass Sie möglichst wenig Code neu schreiben müssen und so viel wie möglich von der Basisklasse **Rechteck** verwenden können.

Aufgabe 4:

Implementieren Sie eine Klasse **Dreieck.** Diese hält Werte für alle Seiten.
Dreiecke existieren als eigenständige Objekte aber nicht, nur die sie ableitenden Klasse **DreieckGleichseitig**, **DreieckGleichschenkelig**, **DreieckBeliebig** und **DreieckRechtwinkelig** dürfen instanzierbar sein.
Jede dieser Klassen muss die im Konstruktor übergebenen Seiten auf ihre eigene Art prüfen, um festzustellen zu können, ob die Übergabe-Parameter tatsächlich dem Dreiecks-Typ entsprechen.

Aufgabe 5:

Erstellen Sie eine Klasse **FigurenVerwaltung** , welche in einer ArrayList **Figur**en (Kreise, Rechtecke usw.) verwaltet.
add(figur) wird verwendet, um eine Figur hinzuzufügen.
berechneGesamtFlaeche() retourniert die Summe der Flächen aller Figuren.
berechneGesamtUmfang() gibt die Summe der Umfänge aller Figuren zurück.
toString() erstellt eine Liste aller Figuren mit deren Informationen.

Aufgabe 6:

Erweitern Sie die Klasse **FigurenVerwaltung** um die Methoden
sortiereUmfang() sowie
sortiereFlaeche().

Dreiecks-Berechnungen:

Gleichseitiges Dreieck:

Umfang: 3a Fläche: $\frac{a^2 \sqrt{3}}{4}$

Gleichschenkeliges Dreieck:

Umfang: 2a + c Fläche: $\frac{c}{2} \sqrt{a^2 - \frac{c^2}{4}}$ oder $\frac{c \cdot h_c}{2}$ // h_c -> Höhe auf c

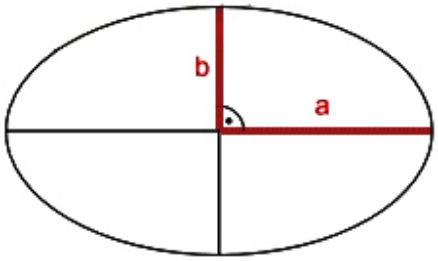
Rechtwinkeliges Dreieck:

Umfang: a+b+c Fläche: $\frac{a \cdot b}{2}$

Beliebiges Dreieck:

Umfang: a+b+c Fläche: $\sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$ wobei s = $\frac{a+b+c}{2}$

Ellipsen-Berechnungen (Ausgangs-Maße sind die Halbachsen!!, also Länge/2 und Höhe/2):



Fläche:

```
double a, b;     // sind die Halbachsen-Maße
double f = Math.PI * a*b;
// oder
double e = (Math.sqrt(a*a-b*b))/a;
double f = Math.PI* a*a * (Math.sqrt( 1- e*e) );
```

Umfang (Näherungsformel von Ramanujan):

$$u \approx (a+b) \cdot \pi \cdot (1 + \frac{3\lambda^2}{10 + \sqrt{4 - 3\lambda^2}}) \quad \text{mit} \quad \lambda = \frac{a-b}{a+b}$$

Berechnung in Java:

```
double a, b;  // Initial-Zuweisungen ergänzen!!!
double lambda= (a-b)/(a+b);
u = (a+b) * Math.PI * ( 1 + ( 3*lambda*lambda / ( 10 + Math.sqrt(4-3*lambda*lambda) ) ) );
```

für die Methode **"enthaeltPunkt"** gibt es mehrere Möglichkeiten (Vektor-Rechnung, Winkelfunktionen), am Leichtesten umzusetzen mit dem Satz von Heron….