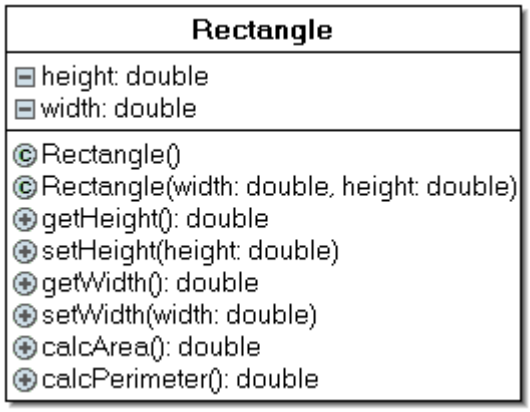
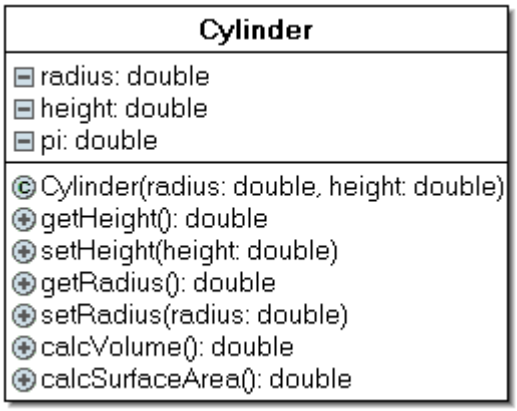
Aufgabenblatt: OOP

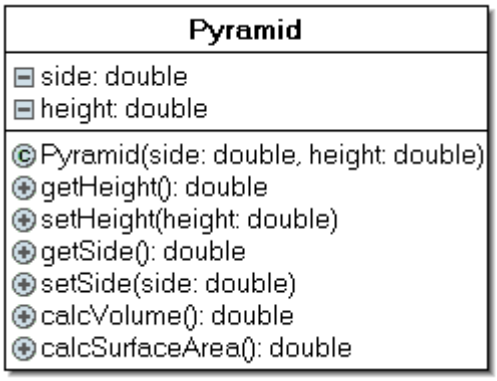
(1.) Gegeben ist nebenstehende die Klasse Rectangle!
Laden Sie die zugehörige Datei OOP und entpacken diese. In jedem Ordner Vorlagen befindet sich zwei Dateien die jeweils zu ergänzen sind!

- *(a.) Laden Sie die Datei TestRectangle.java! Deklarieren sie eine Instanz von Rectangle, initialisieren Sie die Variable mit Konstruktor ohne Parameter, weisen Sie die Werte 5 und 7 den Seiten zu und geben Sie die Seitenlängen, den Flächeninhalt und den Umfang aus!*
- *(b.) Verfahren Sie wie in (a.), verwenden Sie aber den Konstruktor mit den beiden Parametern um die Werte 5 und 7 zuzuweisen!*
- *(c.) Laden Sie die Datei Rectangle.java und kopieren Sie eine Ihrer beiden Lösungen TestCRectangle.java in den Ordner! Füllen Sie die fehlenden Informationen zu der Klasse Rectangle hinzu, damit die Klasse wie in (a.) vollständig ist! Erstellen Sie auch einen Konstruktor, so dass nur die Breite eingelesen wird (Die Höhe soll dann automatisch 7 sein!)!*
- *(d.) Erstellen Sie eine Klasse Cylinder, die sinnvolle Methoden und Eigenschaften zum Berechnen von Oberfläche und Volumen beinhaltet und schreiben Sie ein Rahmenprogramm zum Testen! Achten Sie auf sinnvolle Namensgebung!*

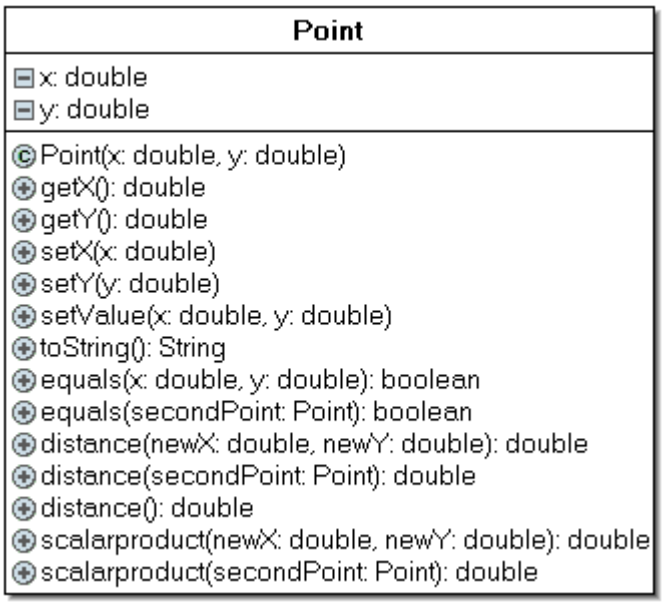




- *(e.) Erstellen Sie eine Klasse Pyramid, die sinnvolle Methoden und Eigenschaften zum Berechnen von Oberfläche und Volumen einer Pyramide mit quadratischer Grundfläche beinhaltet und schreiben Sie ein Rahmenprogramm zum Testen! Achten Sie auf sinnvolle Namensgebung!*



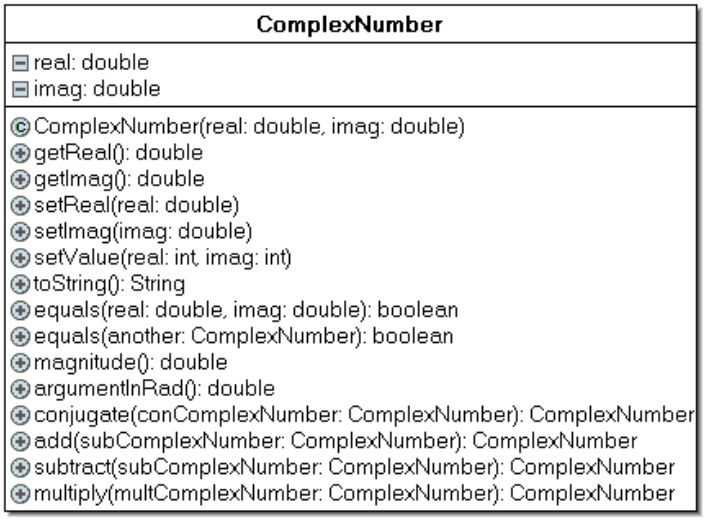
(2.) Schreiben Sie eine Klasse Point die zwei Konstruktoren, zwei Setter und Getter für die internen Variablen x und y, ein Setter für x und y, eine Methode zur Ausgabe von x und y als String und eine Methode distance zum Berechnen des Abstandes des Punktes von einem beliebigen Punkt enthält!



(*3.)Schreiben Sie eine Klasse Square, die drei Eigenschaften length, perimeter und area enthält. Wenn eine Eigenschaft dieser Klasse gesetzt wird, werden automatisch alle anderen Eigenschaften neu berechnet!*

(4.) Schreiben Sie eine Klasse ComplexNumber, nebenstehende Methoden und Eigenschaften enthält!
*Eine komplexe Zahl z hat die Form x+y*i, wobei i= Wurzel aus -1 ist!*

- *x ist der Realteil*
- *y ist der Imaginärteil*
- *magnitude ist der Betrag also* $\sqrt{x^2 + y^2}$
- *argument ist Math.atan2(y, x)*
- *conjugate berechnet die konjugiert komplexe Zahl d.h. das Vorzeichen von y wird umgekehrt*
- *add berechnert die Summe zweier komplexer Zahlen: a+bi + c+di =(a+c) + (b+d)i*
- *multiply berechnet das Produkt: (a + bi) * (c + di) = (ac - bd) + (ad + bc)i*



(5.) Schreiben Sie eine Klasse GradesAverage, welches den Durchschnitt und Standardabweichung von Noten 0-15 mit gegebener Anzahl der Noten ausgibt. Erstellen Sie eine Klasse ArrayIO, welche die Eingaben und Ausgaben übernimmt.

(6.) Erstellen Sie eine Klasse Circle, die den Flächeninhalt und den Umfang berechnet. Diese enthält auch eine unveränderliche Konstante PI, aber keinen Konstruktor oder private Variablen!

(7.) Erstellen Sie eine Klasse Fraction, die Brüche addiert, subtrahiert, multipliziert, dividiert, kürzt und erweitert. Enthalten sein soll auch eine Methode frac2Double und double2Frac!
(a.) Erstellen Sie die Klasse mit den privaten Attributen Zähler und Nenner.
(b.) Erstellen Sie die Klasse mit statischen Methoden analog der Math-Klasse!

(8.) (a.) Die Windgeschwindigkeit wird in km/h, Knoten oder in der Beaufort Skala(Werte 1 bis 12). Erstellen Sie eine Klasse WindSpeed, die im Konstruktor die Windgeschwindigkeit in km/h einliest! Die Klasse hat Methoden zum Auslesen der Windgeschwindigkeit in km/h, Knoten oder auf der Beaufort Skala!
Des Weiteren muss es mit der Klasse möglich sein zu bestimmen, ob es windstill ist(<2 km/h) oder gerade ein Sturm=Orkan(>120 km/h) vorliegt!
Eine nautische Meile(Knoten) ist 1,852 Kilometer. Die Beaufort Skala ist definiert als $v=3{,}01*B^{1,5}$*, wobei v die Windgeschwindigkeit in km/h ist. Der Beaufort Wert wird zur nächsten ganzen Zahl gerundet und es gibt keine größeren Werte als 12.*
Hinweis: a^b *kann mit Math.pow(a,b) berechnet werden.*
(b.) Zeichnen Sie das UML-Diagramm von der Klasse auf ein Blatt und überprüfen Sie das Ergebnis mit dem Java-Editor!
(c.) Schreiben Sie ein Hauptprogramm(e), um die Klasse zu testen! Informieren Sie sich über White-Box-Test und Black-Box-Test und geben Sie ihre Testfälle an!

(9.) Erstellen sie eine Klasse Polynomial, die eine ganzrationale Funktion in Form der Koeffizienten speichert! Sie enthält folgende Methoden:
- zwei Konstruktoren, einen ohne und den anderen mit einem Argument, welches als Array die Koeffizienten der Funktion enthält!
- eine Methode valueAtX, welche z.B. den Wert von f(5) berechnet!
- zwei Methoden add und multiply, welche die Summe und das Produkt zweier Polynome als Ergebnis übergibt!
- eine Methode getDegree, welches den Grad der Funktion zurückgibt
- eine Funktion getCoeffizient, welche den n-ten Koeffizienten wiedergibt
- eine Funktion getf1, welche die 1. Ableitung wiedergibt

(10.)
Implement a simple calculator which evaluates arithmetic expressions given as a String. The expression should only consists of numbers 0 to 9, parenthesis, and the binary and unary operators + and -. White spaces are ignored. Implement a recursive descent parser for reading in the String.

The syntax is given as the following EBNF:

expression = term, ["+" | "-" , term] ;
term = "(", term, ")"
| "0" | "1" | ... | "9" ;
The follwing strings are valid: "1", "((2))", "2 + 3", "((4) - 5 +7)".

Inspect the methods of a String and the class Character

Mögliche Aufgaben hier:
http://www.ntu.edu.sg/home/ehchua/programming/java/J3f_OOPExercises.html#ExerciseClass
1.7 Exercise: The MyPolynomial Class