Familienname:

Vorname:

Matrikelnummer:

Aufgabe 1 (4 Punkte):
Aufgabe 2 (2 Punkte):
Aufgabe 3 (2 Punkte):
Aufgabe 4 (1 Punkte):
Aufgabe 5 (1 Punkte):
Aufgabe 6 (7 Punkte):
Aufgabe 7 (4 Punkte):
Aufgabe 8 (3 Punkte):
Aufgabe 9 (6 Punkte):

Gesamtpunktzahl:

Schriftlicher Test zu C++ (60 Minuten)
VU Einführung ins Programmieren für TM

19. Januar 2012

Im Verlauf dieses Tests sollen die durch die folgenden UML-Diagramme dargestellten Klassen verwendet werden.
Hinweis: Nicht alle Methoden in den unten stehenden Diagrammen sind im Laufe des Tests tatsächlich zu implementieren.

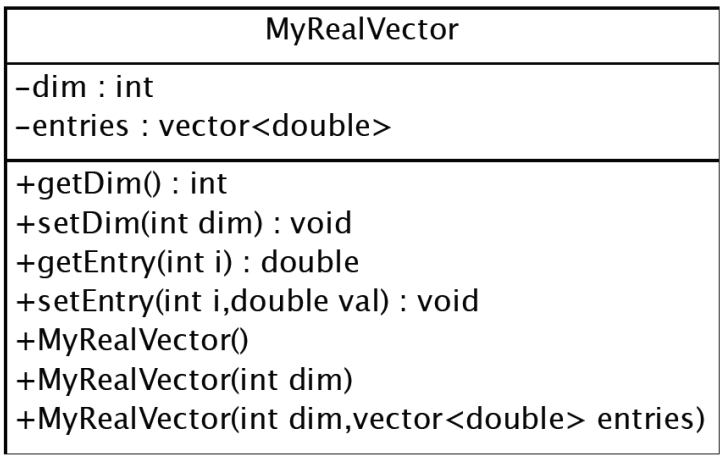

Abbildung 1: UML-Diagramm für die Klasse `MyRealVector`. Diese Klasse ist **nicht** zu implementieren.

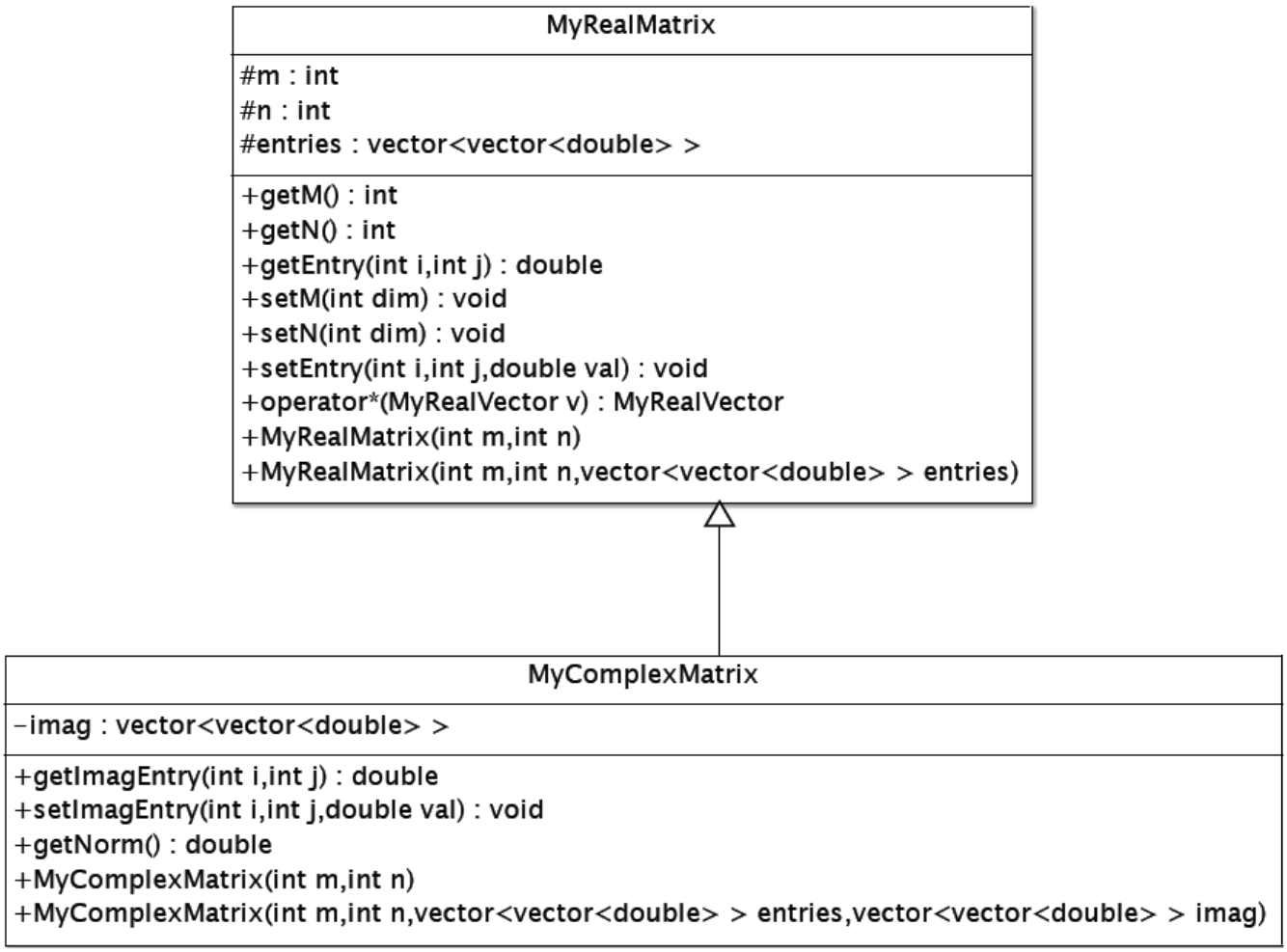

Abbildung 2: UML-Diagramm für die Klassen `MyRealMatrix` und `MyComplexMatrix`. Beachten Sie, dass nicht alle Methoden vollständig zu implementieren sind.

Aufgabe 1 (4 Punkte). Schreiben Sie die Klassendefinition für eine Klasse `MyRealMatrix` mit allen Methoden und Feldern die Sie obigem UML-Diagramm entnehmen können. Die einzelnen Methoden sollen an dieser Stelle noch nicht implementiert werden.

Aufgabe 2 (2 Punkte). Implementieren Sie einen Konstruktor für die Klasse `MyRealMatrix`, der bei Übergabe zweier Integer m und n eine $m \times n$ Nullmatrix vom Typ `MyRealMatrix` erzeugt.

Aufgabe 3 (2 Punkte). Überladen Sie den Konstruktor aus Aufgabe 2 so, dass Sie (zusätzlich zu den Dimensionen m und n) auch die Einträge der Matrix im Format `vector<vector<double> >` mit übergeben können.

Aufgabe 4 (1 Punkt). Implementieren Sie die Methode `getEntry`, die für gegebene Indizes i, j den Eintrag A_{ij} einer Matrix A vom Typ `MyRealMatrix` zurückgibt.

Aufgabe 5 (1 Punkt). Implementieren Sie die Methode `setEntry`, die zu gegebenen Indizes i, j den Wert val als Eintrag A_{ij} in einer Matrix A vom Typ `MyRealMatrix` abspeichert.

Aufgabe 6 (7 Punkte). Implementieren Sie eine Matrix-Vektor-Multiplikation, der Sie einen Vektor vom Typ `MyRealVector` übergeben. Der Rückgabewert der Methode soll ebenfalls vom Typ `MyRealvector` sein. Zu einer Matrix A vom Typ `MyRealMatrix` und Vektoren x, b vom Typ `MyRealVector` soll der Aufruf $x = A * b$ das Ergebnis der Matrix-Vektor-Multiplikation in x speichern.
Hinweis: Die Klasse `MyRealVector` müssen Sie hierzu nicht extra implementieren. Sie können stattdessen alle im obigen UML-Diagramm dargestellten Methoden einfach verwenden.

Aufgabe 7 (4 Punkte). Schreiben Sie eine Klasse `MyComplexMatrix`, die Sie öffentlich von `MyRealMatrix` ableiten. Diese soll über einen zusätzlichen Vektor `imag` zur Speicherung der imaginären Einträge sowie eine Funktion `getNorm` verfügen. Zudem soll die Klasse über `get-` und `set-`Methoden verfügen, um auf den imaginären Anteil zugreifen zu können. Außerdem redefiniert die Klasse `MyComplexMatrix` die beiden Konstruktoren aus `MyRealMatrix`, wie es sich an obigem UML-Diagramm ablesen lässt. Analog zu Aufgabe 1 ist hier nur die Klassendefinition gefragt, es sollen also keine Methoden implementiert werden.

Aufgabe 8 (3 Punkte). Was bedeuten die Zugriffsspezifizierer `private`, `public` und `protected`. Warum werden Sie verwendet?

Aufgabe 9 (6 Punkte). Die Frobeniusnorm einer (reellen oder komplexen) $m \times n$ Matrix $A \in \mathbb{C}^{m \times n}$ ist durch

$$\|A\|_F := \sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |A_{ij}|^2}$$

gegeben, wobei $|\cdot|$ den entsprechenden Betrag bezeichnet und der Betrag einer komplexen Zahl $a \in \mathbb{C}$ mit $a = a_1 + i \cdot a_2$ als

$$|a| := \sqrt{a_1^2 + a_2^2}$$

definiert ist. Implementieren Sie die Methode `getNorm` aus `MyComplexMatrix` so, dass Sie $\|A\|_F^2$ zurückliefert. Ohne Implementierung können Sie hierzu auch die restlichen Methoden verwenden, die im UML-Diagramm zu `MyComplexMatrix` angegeben sind.