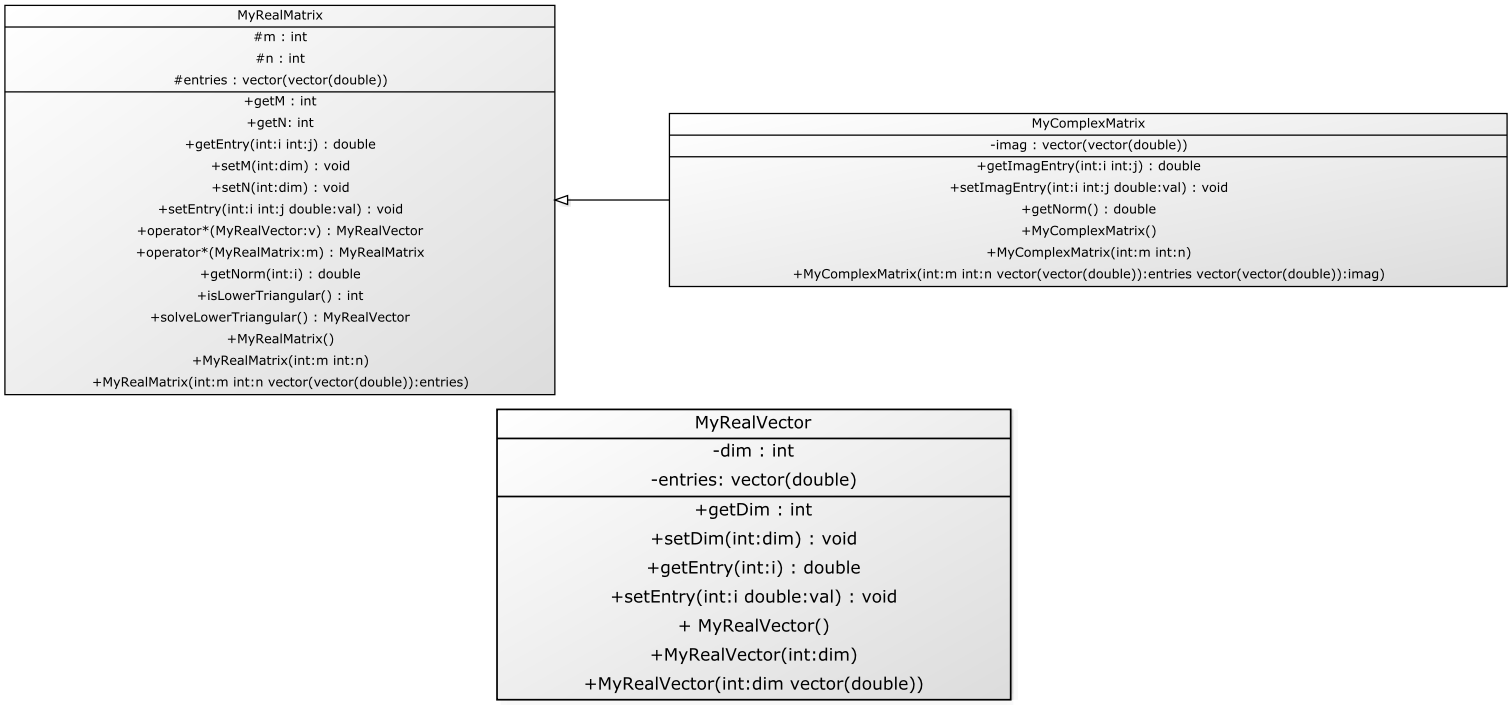
Dirk Praetorius,
Marcus Page

Wintersemester 2011
über die Ferien 2011/2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 10

In den folgenden Aufgaben implementieren Sie die Klassen `MyRealMatrix` und `MyComplexMatrix` entsprechend dem nachfolgenden UML Diagramm. Die einzelnen Probleme werden hierbei in verschiedenen Aufgaben gelöst.



Aufgabe 10.1*. Erweitern Sie Ihre Klasse `MyRealVector` aus Aufgabe 8.3 so, dass sie obigem UML Diagramm entspricht. Implementieren Sie insbesondere einen Konstruktor dem Sie eine Länge $n \in \mathbb{N}$, sowie einen Vector der Länge n übergeben um damit ein entsprechendes Objekt zu initialisieren.

Aufgabe 10.2*. Schreiben Sie die Klasse `MyRealMatrix`. Implementieren Sie `get`- und `set`-Methoden um Dimensionen und Einträge Ihrer Matrizen manipulieren zu können. Schreiben Sie außerdem Konstruktoren um eine Nullmatrix der Größe $m \times n$, bzw. eine voll besetzte Matrix der Größe $m \times n$ zu initialisieren.

Aufgabe 10.3*. Implementieren Sie die Funktion `getNorm` die, abhängig vom Eingabewert (1,2 oder 3), die Zeilensummen-, Spaltensummen oder die Frobeniusnorm zurückgibt. Die Frobeniusnorm einer

$(m \times n)$-Matrix ist hierbei durch

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

gegeben.

Aufgabe 10.4*. Schreiben Sie eine Matrix-Vektor-Multiplikation die mittels $*$-Operator aufgerufen werden kann und den Lösungsvektor im Format `MyRealVector` zurückgibt.

Aufgabe 10.5. Überladen Sie den $*$-Operator von `MyRealMatrix` so, dass je nach Signatur auch eine Matrix-Matrix-Multiplikation möglich ist.

Aufgabe 10.6. Schreiben Sie die Klasse `MyComplexMatrix` die Sie von `MyRealMatrix` ableiten. Wie Sie dem oben stehenden UML Diagramm entnehmen können, soll diese Klasse zusätzlich imaginäre Einträge in `imag` speichern. Schreiben Sie sich `get-` und `set`-Methoden um auf die imaginären Werte zugreifen zu können. Schreiben Sie außerdem einen Konstruktor der Vektoren für Real- und Imaginärteil übernimmt und die Matrix entsprechend initialisiert. Der Realteil soll hierbei über den Orginalkonstruktor von `MyRealMatrix` initialisiert werden.

Aufgabe 10.7. Redefinieren Sie die Methode `getNorm` in `MyComplexMatrix` so, dass sie die Frobeniusnorm der komplexen Matrix zurückgibt.

Aufgabe 10.8. Implementieren Sie die Methode `isLowerTriangular` in `MyRealMatrix` so, dass Sie 1 zurückliefert, falls die Matrix untere Dreiecksmatrix ist, d.h. falls $A_{ij} = 0$ für $i < j$ gilt. Ansonsten soll 0 zurückgegeben werden.

Aufgabe 10.9. Implementieren Sie die Methode `solveLowerTriangular`, die für eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Vektor $b \in \mathbb{R}^n$ die Lösung $x \in \mathbb{R}^n$ von $Ax = b$ berechnet und als Vektor zurückgibt. Stellen Sie sicher, dass diese Methode nur aufgerufen werden kann, wenn es sich tatsächlich um eine untere Dreiecksmatrix handelt. Wie könnte man dieses Problem im Sinne der Objektorientierung 'besser' lösen?

Aufgabe 10.10. Schreiben Sie eine Klasse `Hangman` mit den Methoden `guessChar`, `solve` und `newString`. Die Klasse soll nun einen `string` der Länge n speichern, den es zu erraten gilt. Nach und nach dürfen mittels `guessChar` Buchstaben geraten werden, wobei die Methode immer den Index, bzw. die Indizes der eingegebenen Buchstaben auf dem Bildschirm ausgibt. Falls der eingegebene Buchstabe im gesuchten Wort nicht vorkommt, soll eine entsprechende Meldung ausgegeben werden. Der Spieler verliert, wenn er 8 mal falsch geraten hat. Wie lässt sich das geschickt umsetzen? Schreiben Sie alle nötigen Zugriffsfunktionen und Konstruktoren und außerdem die Methoden `solve` und `newString` um das Rätsel zu lösen bzw. das Spiel mit einem neuen Wort neu zu beginnen. Testen Sie Ihr Spiel indem Sie mittels einer geeigneten Schleife solange neue Buchstaben raten, bis Sie entweder gewonnen oder verloren haben.