Angular – Controllers, Filters, Directives and Factories

General part

A common way to make a system easier to maintain, is to break it up into smaller, sometimes reusable, parts.

- Explain how java programs can be divided up into smaller parts
- Explain different ways how java "parts" can be reused and included in many projects
- Explain how Angular applications can be divided up into smaller parts

Practical part

In this you will have to demonstrate several of AngularJS's ways to break you code up into smaller parts, like filters, directives, Factories etc.

Unpack the file *FilterDirectivesFactories.zip* and open the project contained in the file. This contains a plain NetBeans Web Project where the only files of interest for this exercise, are the two files in the "Web Pages" folder.

The file *app.js* contains a controller with a hardcoded data-model which you must use to create a table as sketched below:

	Basic Programming	Advanced Programming	DataBase Intro	Average
Peter Hansen	10	12		11.00
Jan Olsen	7	10		8.50
Gitte Poulsen	7	7		7.00
John McDonald	10		7	8.50

The data model contains an array `allCourses`, which hold all courses a student can take (the top row)

It also contains a list of all `students` and their `grades`. You can assume that when the model was built it was done so that each student will have a number of grades matching the number of courses found in the `allCourses` array, and in that order. If a student has not yet taken a course, it is added as an empty grade-item ({}).

1. Rewrite the Controller to use the Controller as Strategy
2. Write the necessary angular code (using ng-repeat) to create the table above, initially without the "average" column
3. Write an angular filter, used to display the average of all grades for a student (last column in the figure above)
4. It is assumed that the table will be used in more than one view, and perhaps even in more than one application. Refactor the code into an angular directive so the full table can be included on an html-page using only a directive like: <student-grades></student-grades>
5. The data model is hardcoded into the controller. Move it into a factory or service and include this in your controller.
6. Show how you would fetch the data, if a REST service existed (don't implement the actual REST service) that could provide a JSON version of data model used up until now.

Note: this part is NOT a part of the exercise, it's meant as FYI.

This is not required for this question, but if you want to do part 6 the cool way, here is a hint for how you can actually use the REST call implemented in step 6, without the need for a real backend:

The angular module provides a function `run(..)` function which you can use to register work used for the rest of the application as sketched below. You could inject $httpBackend and provide a mock response so you could actually use the code from step 5

```
app.run(['$httpBackend',function($httpBackend) {
//Use the httpBackend to provide a Mocked result for the REST call in part 5
}]);
```

Using this strategy you can create the service or factory as if the backend really existed, get a real response and you only have to change a few lines (the run part) when you want to connect to a real backend.

You must include angular-mocks.js (included with the start code), and inject it as sketched below, if you want to try this

```
var app = angular.module('examEx19', ["ngMockE2E"]);
```