Patterns and their implementation

General part

- Briefly explain about Design Patterns, what they are, and what they can do for us
- Provide practical examples of design patterns, using code you have implemented in any of the last three semesters.
- This is a number of suggestions (feel free to choose others) to pick from when you select your examples: Singleton, Observer, Façade, Factory, Bridge, Composite, Command etc.
- You must include both Java and at least one JavaScript example.

Practical part

1) Often, when we create our web applications, we have the need for test data. Implement a reusable nodejs module, using JavaScripts *module pattern*, which can provide random test data as sketched below[1]:

```
var data = dataGenerator.getData(100,"fname, lname, street, city, zip");
```

This should return a JavaScript array (not JSON) with 100 test data on the form:

```
[{fname: "Bo", lname:"Hansen", street: "Lyngbyvej 26", city: "Lyngby", zip: "2800"},..]
```

If you call it like this: `var data = dataGenerator.getData(25,fname, lname);`

it should return 25 test data as sketched below:

```
[{fname: "Bo", lname:"Hansen"},..]
```

2) Create an Express application, which should implement a REST service, that when called like this:

```
GET: http:/……./api/addresses/100/fname,lastname,zip
```

Should return a JSON array as sketched below:

```
[{"fname": "Bo", "lname" : "Hansen","zip": "2800"}, {..}, …]
```

*Pick only one of the two suggestions given below (3b is **advanced**)*

3a) Create a simple web page that, using whatever technology you prefer, renders a table that should show the test data fetched via the REST method implemented in step 2.

3b) The module, as implemented in step 1, has a serious problem. If it's called to generate a large number of sample data, it could seriously block the caller, until all data are generated. Rewrite the getData() method using JavaScripts Callback Pattern to be called like:

```
dataGenerator.getData(100,"fname, lname, zip",function(err,callback(data));
```

Hints for 3b:
Use the `process.nextTick(..)` method to voluntarily give up control (for example for each 1000 names you generate) See "Learning Node.js p58-59 or http://howtonode.org/understanding-process-next-tick

---

[1] *Hint: Include some hard coded arrays in your module like ["Ib", "Bo", "Lars", "Henrik"] and pick a random value for each name you generate*

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

Patterns:

You have been introduced to design patterns many times during your three semesters with us.

As pre-preparation for pattern questions, you could find some of your old pattern examples for implementation examples. You will not have time to do this during your 80 minutes preparation, so do this during your general preparation up till the exam.

You should have a great deal of examples from second semester, including your second semester project.

This semester, you probably used the observer pattern in your chat client

The JavaScript callbacks are "a kind of observers" (your callback is called, when "data" is ready)

For JavaScript patterns, we have (as a minimum) introduced the module pattern, the IIFE pattern and have used factories, facades and MVC for our JavaScript CA's.

General advices:

Spend you 80 minutes well. Don't (initially) waste your time in trying to come up with "sensible" data.

As a start ["aa","bb","cc","dd"] is just as good as ["Lyngbyvej 25", "Vesterbrogade 23","Sofie vej 23"]

Spend you 80 minutes well. It is better to have a solution that can only generate *firstname* and *lastname*, but also includes the REST part, and perhaps the client side part, compared to a solution that only covers step-1. The final fields can be added if you have time.

If your first impression was, "this is a hard question", don't spend any time on reading parts marked with "this is advanced".

Links and references:

- http://en.wikipedia.org/wiki/Software_design_pattern
- http://toddmotto.com/mastering-the-module-pattern/
- http://en.wikipedia.org/wiki/Immediately-invoked_function_expression

From the second semester exam paper:

- Martin Fowler: Patterns of Enterprise Application Architecture,
- Addison Wesley 2003, ISBN 0-321127420:
- kap 1 (Layering) + s.30-32 (Service Layer) + s.116-119 (Domain Model) + s.133-136 (Service Layer).
- s.33 (~ 1 side af Mapping to Relational Databases), s.50-52 (Database Connections), 165-170 (Data Mapper), 216-218 (Identity Field) , 236-244(Foreign Key Mapping)
- http://userpages.umbc.edu/~tarr/dp/lectures/Facade.pdf
- http://userpages.umbc.edu/~tarr/dp/lectures/Singleton.pdf