Exercise Sheet 8

December 7: Abstract Factory and Singleton

Exercise 1

Read the Java tutorial on JDBC Basics for connecting to a database (`http://download.oracle.com/javase/tutorial/jdbc/basics/index.html`). In particular, try to understand where the Abstract Factory Pattern is used!

Exercise 2

Have a look at the `XMLReaderFactory` class (`http://docs.oracle.com/javase/7/docs/api/org/xml/sax/helpers/XMLReaderFactory.html`). Is this an example of the Abstract Factory Pattern?

Exercise 3

(Optional) Create a simple Parser for Arithmetic Expressions (compare exercise 3) based on the shunting-yard algorithm by Edsger Dijkstra (`https://en.wikipedia.org/wiki/Shunting-yard_algorithm`). The Parser should have the following public method

```
public ArithmeticExpr parse(String input)
```

returning an ArithmeticExpr corresponding to an input. Example inputs: $3 + 4 * 5$, $6 + 4 * 8 + 3$, etc. If you do not feel comfortable about the shunting-yard algorithm you could develop a simple recursive descent parser instead.

Hints:

- Maintain two stacks
  - `Stack<ArithmeticExpr> exprStack` and
  - `Stack<ArithmeticExpr> operatorStack`

  for the operators and the composite Expressions.
  Read the explanation in `https://en.wikipedia.org/wiki/Shunting-yard_algorithm`) carefully and implement the parsing logic accordingly
- Think of operator precedence later (ignore for now)
- ignore error handling
- Is the class `ArithmeticExpressionParser` an example of the Factory Pattern?

Exercise 4

How would you test the uniqueness of a singleton in a JUnit test case? Any problems associated with Unit Testing singletons?

Exercise 5

Implement a singleton in Java and try to make it subclassable!

Hints

- Consult the literature!
- You can work in pairs, if you want!
- If you want to learn a Java API, look into the java docs!
- Always use the same familiar IDE (suggestion Eclipse)!