Semester assignment I
Inf102 fall 2011
Deadline: 22:00hr Thursday 29. of September

There will be two mandatory assignments and one classroom test in this course. In total these will count for 30% of the final grade. Both mandatory assignments have to be passed in order to qualify for the exam but if the classroom test has a better score than one of the mandatory assignments then this grade can be used in stead. In other words each mandatory assignment counts for 15% of the final grade.

In order to get the mandatory assignment evaluated the following conditions have to be met:

- **Overview**
  A text document containing an overview of the completed assignment. This should contain your name, e-mail address, and a short description of the added files.

- **Source code**
  During the evaluation we will check that the source code compiles and produces correct answer for a set test data not available to the students. All java files necessary to compile the project has to be included in the submittion, but do not add .class files. Other files that are of importance to the project can also be included in the submittion.

  All source code should be properly commented and indented in a way that makes it easy to read and understand how the code is supposed to work.

- **Analysis**
  Add a separate text document(any normal format is fine) where the running time for each implemented method is given in O-notation. If the method is not trivial it is also requited to justify the running time. Points will be given according to the quality of the analysis but the most important thing is to use correct arguments.

The hand inn should consist of a single zip-file with a name composed by your first and last name. If your name is Jon Doe and you want to deliver files MyBinaryTree.java, MyOblig1.java, oversikt.txt from a Linux system then the zip file with name JonDoe.zip can be crated by the command
"zip JonDoe MyBinaryTree.java MyOblig1.java oversikt.txt". Bytecode or files of type *.class shall not be included in the zip file. The assignment should be delivered through the course web page. On the main page of the course there is a rectangle at the right named assignments. In this rectangle there is a folder called

Assignment folder and in this folder there is another folder called Assignment 1, where the assignment should be delivered. Notice that the code handed in will be recompiled and tested on new data which are not available for the students.

Individual work

Each student is responsible of writing and completing his or here own code. If two delivered assignments are so similar that we suspect cheating then both will fail. It is your own responsibility to ensure that no one copy your code. The same band also holds for copying from books, internet, or other sources.

Task 1

In the file storage folder on the course web page there are some files associated to this assignment. In particular two interfaces called Stack.java and BinaryTree.java Task one is to implement these two interfaces in two classes called MyStack and MyBinaryTree. The stack implementation should be generic as it will be used in different settings later, while MyBinaryTree can be of type java.lang.Character. For the stack implementation you are free to use the java.util.ArrayList class.

Task 2

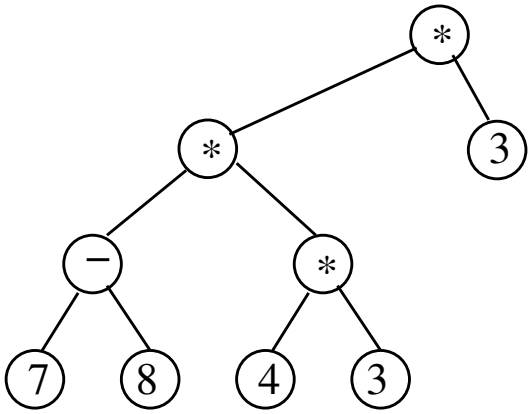
In this task we will use the data structures implemented in the previous task to make a program that evaluates mathematical expressions. Each expression will consist of operators from the set $\{+, -, *, /\}$ and to avoid tedious string manipulations each operand will be one of the integers from $0 - 9$. We will use o to represent any of the operators and d to represent any of the then possible single digit integers. Every legal mathematical expression M will be on the form defined below:

- $M = (MoM)$
- $M = (doM)$
- $M = (Mod)$
- $M = (dod)$

Let us give some examples:

- (3-4+5), 3-3, (3) are not legal expressions
- ((3-5)+5), (3-3), ((3-0)/2) are legal expressions.

The task is to rephrase the mathematical expression into a Postfix(Reverse Polish) notation and evaluate the expression. Reverse Polish notation is a mathematical notation wherein every operator follows all of its operands, i.e. $((3 - 5) + 5)$ becomes $35 - 5+$. As a hint one could mention that both the interfaces implemented in task 1 will be useful to complete this task. Below are some running examples on different inputs. It will not be necessary to add rigid test to



Figur 1: An example of the created binary tree for expression "(((7-8)*(4*3))*3)".

verify that the provided mathematical expression is legal, the most important thing is that it computes the result correctly.

Below are some example output from the program:

```
bash-4.1$ java AritExp "(((3-4)*2)+1)"
34-2*1+
(((3-4)*2)+1)=-1.0

bash-4.1$ java AritExp "((4-3)/3)"
43-3/
((4-3)/3)=0.3333333333333333
```

Task 3

People in Bergen are very interested in weather and especially how much it rains. In this assignment we are given the precipitation measured in mm for each day through the year 2010. In weather forecasts it is often mentioned the previous day exceeding today's measurements. For instance, "it has not rained as much as this since 5. of May 2010".

In the file storage folder for the course there is a data file Bergen2010.csv and a class LoadData used to read these data into objects of the PrecipitationDay class. Your task is to find the previous day it rained more than the current PrecipitationDay object and set the corresponding reference in the PrecipitationDay object. Finally print all days in correct order with this extra information. An example of the output for the first ten days can be found below.

Day	Month	Precipitation	Previous day with more	Prev. Prec.
1	jan	0.0	Out of range	
2	jan	0.1	Out of range	
3	jan	0.0	2 jan	0.1
4	jan	1.1	Out of range	
5	jan	0.8	4 jan	1.1
6	jan	0.0	5 jan	0.8
7	jan	1.0	4 jan	1.1
8	jan	0.0	7 jan	1.0
9	jan	0.0	7 jan	1.0
10	jan	0.0	7 jan	1.0