Laboration 4

1 Syntaxanalys

För att testa programmet i laboration 2 var du tvungen att konstruera den abstrakta representationen av uttrycken med hjälp av de olika konstruerarna.

I denna laboration skall du konstruera en parser som analyserar en sträng som beskriver ett satslogiskt uttryck och bygger den abstrakta representationen.

Den konkreta grammatiken ges av

```
expr      ::= primary ('->' primary)?
primary   ::= term ('|' term)*
term      ::= factor ('&' factor)*
factor    ::= ID | '!' factor | '(' expr ')'
```

`ID` är ett variabelnamn som beskrivs av det reguljära uttrycket `[a-z][a-z0-9]*`.

Laboration 2 innehöll också ekvivalens. Eftersom detta inte tillför något principiellt annorlunda har det utelämnats.

Parserklassen skall tillhandahålla metoden

```
public Expr build(String string)
```

Metoden skall kasta ett undantag om `string` inte genereras av grammatiken.

Var noga med att följa konventionen att varje parser-metod ansvarar för att nästa grundsymbol har hämtats när metoden tar slut.

Det finns två testklasser som utför samma uppsättning av tester. Den ena använder JUint, den andra inte. Välj själv vilken du vill använda.

2 Förberedelser

Det kan vara svårt att hinna med att utföra hela uppgiften under själva laborationspasset. Du bör alltså förbereda dig.

1. Studera det exempel på syntaxanlys av aritmetiska uttryck som finns i kapitel 6-7 i Diskreta strukturer. På föreläsningsbilderna från F12 finns motsvarande exempel med samma scanner som i laborationen.
2. På hemsidan finns ett paket som innehåller en färdig skanner och ett skelett till Parserklassen samt testklasser. Studera det! Paketet skall användas tillsammans med resultatet från Laboration 1.
3. Börja med att implementera metoden `Parser.factor`.