Scala Übung 1

Im ersten Teil der Vorlesung haben sie mit Haskell einen Compiler und Interpreter für eine einfache imperative Programmiersprache entwickelt. In dieser Aufgabe soll der Interpreter in Scala realisiert werden. Es handelt sich dabei fast um eine 1:1 Übersetzung der Haskell-Lösung. Anstelle der Haskell-Datenklassen verwenden Sie in Scala Case-Klassen.

Für die Bool'schen Ausdrücke sehen die Definitionen z.B. wie folgt aus:

```
abstract class BoolOperator
object CondAnd extends BoolOperator
object CondOr extends BoolOperator

abstract class BoolExpr
case class LitBExpr(value: Boolean) extends BoolExpr
case class NegBExpr(expr: BoolExpr) extends BoolExpr
case class DyaBExpr(operator: BoolOperator,
            op1: BoolExpr, op2: BoolExpr) extends BoolExpr
```

und Bool'sche Ausdrücke können dann mit folgender Funktion evaluiert werden, wobei dabei der State für Variablen in einer `Map` aus dem Paket `scala.collection` gehalten wird. Eine Map kann mit der Methode `m.updated(key, value)` aktualisiert werden (das Resultat ist eine neue Map)

```
type Value = Int
type State = Map[String, Value]
def eval(expr: BoolExpr, state: State) : Boolean =
   expr match {
   case LitBExpr(value) => value
   case NegBExpr(expr) => ! eval(expr, state)
   case DyaBExpr(CondAnd, op1, op2) =>
                          eval(op1, state) && eval(op2, state)
   case DyaBExpr(CondOr, op1, op2) =>
                          eval(op1, state) || eval(op2, state)
   }
```

Mit obigen Definitionen kann z.B. bereits folgender Ausdruck evaluiert werden:

```
val bExpr = DyaBExpr(CondAnd, LitBExpr(true), LitBExpr(false));
println(eval(bExpr, new HashMap[String, Int]()));
```

Ergänzen Sie diese Definitionen so, dass sie am Ende ein Programm (das jedoch vorerst noch "von Hand" im Sourcecode erfasst werden muss) interpretieren können, wie z.B. folgendes Programm:

```
def main(args: Array[String]) {
  val intDiv = CpsCmd(List(
    AssiCmd("q", LitAExpr(0)),
    AssiCmd("r", IdAExpr("m")),
    WhileCmd(
      RelBExpr(GreaterEq, IdAExpr("r"), IdAExpr("n")),
      CpsCmd(List(
        AssiCmd("q", DyaAExpr(Plus, IdAExpr("q"), LitAExpr(1))),
        AssiCmd("r", DyaAExpr(Minus, IdAExpr("r"), IdAExpr("n")))))),
    SkipCmd));

  val map = Map[String, Int]("m" -> 5, "n" -> 2)
  println("map = " + map);
  println("res = " + Interpreter.interpret(intDiv, map))
}
```

Ausgabe: 02. 05. 2018
Abgabe: 16. 05. 2018