2. Übung zur Vorlesung "Fortgeschrittene funktionale Programmierung"

Boris Dudelsack

13. Oktober 2016

Aufgabe 1: Graphiken

1.1 Graphics.hs

```haskell
module Graphics where
    import Data.Char

    data Point = Point { x :: Double, y :: Double }
        deriving Show

    data Object =
        Rect { p1 :: Point, p2 :: Point, s :: Style }
        | Circle { p :: Point, r :: Double, s :: Style }
        deriving Show

    data Color =
        Black
        | Red
        | Green
        | Blue
        deriving Show

    data Style = Style { c :: Color }
        deriving Show

    colorToStr :: Color → String
    colorToStr(c) = map Data.Char.toLower (show c)

    styleToAttr :: Style → String
    styleToAttr(Style c) = "style=\"fill: " ++ colorToStr c ++ "; stroke: " ++ colorToStr c ++
         ";\"";

    defaultStyle :: Style
    defaultStyle = Style Black

    data Graphic = Nil | Cons Object Graphic
        deriving Show

    single :: Object → Graphic
    single(o) = Cons o Nil

    (<>) :: Graphic → Graphic → Graphic
    (<>) Nil g = g
    (<>) g Nil = g
    (<>) (Cons o1 g1) (Cons o2 g2) = Cons o1 (Cons o2 g1 <> g2)

    objToSVG :: Object → String
    objToSVG(Rect (Point x1 y1) (Point x2 y2) s) = "<rect x=\"" ++ show x1 ++ "\" y=\"" ++
        show y1 ++ "\" width=\""
        ++ show x2 ++ "\" height=\"" ++ show y2 ++ "\" " ++ styleToAttr s ++ "/>"
    objToSVG(Circle (Point x y) r s) = "<circle cx=\"" ++ show x ++ "\" cy=\"" ++ show y ++ "\" r=\"" ++ show r ++ "\" " ++ styleToAttr s ++ "/>"

    toSVG :: Graphic → String
    toSVG g = "<svg version=\"1.1\" xmlns=\"http://www.w3.org/2000/svg\">\n" ++ toSVG_ g ++ "\n</svg>";

    toSVG_ :: Graphic → String
    toSVG_ Nil = ""
    toSVG_ (Cons o Nil) = objToSVG o
    toSVG_ (Cons o g) = objToSVG o ++ "\n" ++ toSVG_ g

    rectangle :: Double → Double → Graphic
    rectangle d1 d2 = single (Rect (Point 0.0 0.0) (Point d1 d2) defaultStyle)

    circle :: Double → Graphic
    circle r = single (Circle (Point (0.0 + r) (0.0 + r)) r defaultStyle)

    colored :: Color → Graphic → Graphic
    colored c Nil = Nil
    colored c (Cons (Rect p1 p2 s) g) = Cons (Rect p1 p2 (Style c)) (colored c g)
    colored c (Cons (Circle p r s) g) = Cons (Circle p r (Style c)) (colored c g)
```

1.2 exercise02.hs

```
{-# OPTIONS_GHC -fno-warn-unused-binds -fno-warn-unused-matches #-}

import Graphics

graphic :: Graphic
graphic = rectangle 10.0 10.0 <> circle 5.0 <> rectangle 20.0 20.0 <> circle 10.0

main :: IO ()
main = writeFile "graphic.svg" (toSVG (colored Green graphic))
```

1.3 graphic.svg

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
<rect x="0.0" y="0.0" width="10.0" height="10.0" style="fill: green; stroke: green;"/>
<circle cx="10.0" cy="10.0" r="10.0" style="fill: green; stroke: green;"/>
<rect x="0.0" y="0.0" width="20.0" height="20.0" style="fill: green; stroke: green;"/>
<circle cx="5.0" cy="5.0" r="5.0" style="fill: green; stroke: green;"/>
</svg>
```