CSc 372

Comparative Programming Languages

19 : Haskell — Modules

Department of Computer Science
University of Arizona

collberg@gmail.com

main.hs

```
module Main where
import TVL
import MyData.List
import Data.List as DL
import Shape

put a = putStr ((show a) ++ "\n")

main =
 do
    put (DL.sort [False',True',Unknown,False'])
    put (circumference (Triangle 3 4 5))
    put (Cons 1 (Cons 2 (Cons 3 Nil)))
    return ()
```

Modules

- The main program should be in the file `main.hs` and be called `Main`.
- There should be a function `main` that is called in startup.
- Module names start with a capital letter.
- There is one module in each file.
- The file name is the module name with a `.hs` extension.
- Dots in the module become sub-directories.

```
module Main where
import ...

main :: IO ()
main = ...
```

Imports

- Definitions can come in any order, except `import` statements which have to appear at the beginning.

```
module Main where
import ...
import Data.List as DL

main :: IO ()
main = ...
```

Imports

```
module Mod where
x = ...
y = ...
z = ...
```

What is brought into scope?

import Mod: `x, y, z, Mod.x, Mod.y, Mod.z`

import Mod (): `(Nothing!)`

import Mod (x,y): `x, y, Mod.x, Mod.y`

import qualified Mod: `Mod.x, Mod.y, Mod.z`

import qualified Mod (x,y): `Mod.x, Mod.y`

import Mod hiding (x,y): `z, Mod.z`

import qualified Mod hiding (x,y): `Mod.z`

import Mod as Foo: `x, y, z, Foo.x, Foo.y, Foo.z`

import Mod as Foo (x,y): `x, y, Foo.x, Foo.y`

import qualified Mod as Foo: `Foo.x, Foo.y, Foo.z`

tvl.hs

```
module TVL where
import Data.List
data TVL = True' | False' | Unknown
instance Eq TVL where
    True' == True'   = True
    False' == False' = True
    _ == _           = False
instance Show TVL where
    show True' = "T"
    ...
instance Enum TVL where
    fromEnum True'   = 0
    ...
instance Ord TVL where
  c <= c' = fromEnum c <= fromEnum c'
```

shape.hs

```
module Shape where
class Shape a where
    area :: a -> Float
    circumference :: a -> Float
data Poly = Triangle Float Float Float
          | Rectangle Float Float
          deriving Show
instance Shape Poly where
    area (Triangle a b c) =
        sqrt(p*(p-a)*(p-b)*(p-c))
            where p = (a+b+c)/2
    area (Rectangle a b)  = a*b
    circumference (Triangle a b c) = a+b+c
    circumference (Rectangle a b) = a+b
```

list.hs

- This module goes in the file `MyData/List.hs` in the sub-directory `MyData`.

```
module MyData.List where
data List a =
    Cons a (List a) |
    Nil

instance Show a => Show (List a) where
    show Nil          = "[]"
    show (Cons x xs) = show x ++
                       ":(" ++ show xs ++ ")"
```

Acknowledgments

- http://www.haskell.org/haskellwiki/Import
- http://en.wikibooks.org/wiki/Haskell/Modules