

MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS

CNG242 Programming Language Concepts – Spring 2021 – Lab 6: Haskell

1. Modules

- A collection of values, data types, functions, etc.
- A module can import other modules, and export some of its resources to other modules.
- Prelude is a module which is imported by default.

Shapes.hs

module Shapes (ThreeDShape(Cube, Cylinder), volume, surfaceArea) where

data ThreeDShape = Cube Float | Cylinder Float Float deriving (Show, Eq. Ord)

volume (Cube length) = length^3
volume (Cylinder height radius) = pi * radius^2 * height

surfaceArea (Cube length) = 6 * length^2 surfaceArea (Cylinder height radius) = (2 * pi * height * radius) + (2 * pi * radius^2)

a. How to import

If you would like to import everything related to the Shapes module, you can use the following command.

import Shapes

If you would like to use only volume function of the Shapes module, you can use one of the following commands.

import Shapes (ThreeDShape(Cube, Cylinder), volume)

or

import Shapes hiding (surfaceArea)

b. Qualified imports

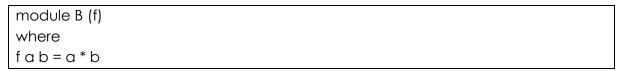
There may be some module which uses the same function names. This causes ambiguous occurrence.

A.hs

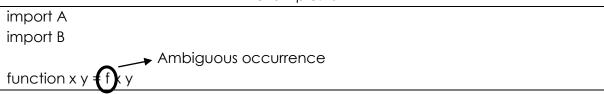
module A (f) where

fab = a + b

B.hs



example.hs



Two ways for the solution to this problem

Way 1:

example.hs

import A import B function x y = A.f x y

Way 2:

example.hs

import qualified A as MA import qualified B as MB

function x y = MA.f x y

2. Useful Modules

• To search functions, or to find out where they are located, use https://www.haskell.org/hoogle/

a. Data.List module

A way to import a module in GHCi

Prelude : module Data.List

Prelude Data.List> sort [3,5,7,2,1]

[1,2,3,5,7]

Prelude Data.List> words "some text delimited with blanks"

["some","text","delimited","with","blanks"]

Prelude Data.List> splitAt 5 "EnverEver"

("Enver","Ever")

Prelude Data.List> 5 `elemIndex` [8,5,2,1,7]

Just 1

b. Data.Char module

Prelude>:module Data.Char

Prelude Data.Char> isLower 'A'

False

Prelude Data.Char> isUpper 'A'

True

Prelude Data.Char> isNumber '1

True

Prelude Data.Char> toUpper 'a'

ΊΑ

Prelude Data.Char> toLower 'A'

'a'

Practical Exercises

1. Write a function that takes a sentence and converts all characters to upper case.

Sample Run:

makeUpper "make me upper"

"MAKE ME UPPER"

2. Write a function that takes a list of numbers and then creates another list which includes pairs with are created by selecting a minimum number and a maximum number until no number is left in the list.

Sample Run:

createTuples [3,6,1,8,7,9,12,4]

[(1,12),(3,9),(4,8),(6,7)]

3. Implement the "capitaliseEachWord" function that takes a sentence to capitalise the first letter of each word.

Sample Run:

capitaliseEachWord "some text delimited with blanks"

"Some Text Delimited With Blanks"

References:

Learn You a Haskell < http://learnyouahaskell.com/modules>

A Gentle Introduction to Haskell < http://www.haskell.org/tutorial/index.html>