Lecture 5: Type Classes

***Exercise 5.1** Read the documentation on `Ord`, and then provide a parsimonious instance of `Ord` for suitably type-constrained instances of `Pair`.

Exercise 5.2 Instead of deriving the default `Show` instance for `Pair`, define one that produces the following:

```
> Pair "Hello" 161
( "Hello" , 161 )
```

***Exercise 5.3** Write a `reverseTree` function for `BinaryTree a` which creates a mirror-image of the original tree, and verify that

```
> toList (reverseTree tree) == reverse (toList tree)
True
```

Note that the `toList` function is not in the `Prelude`, but it is in `Data.Foldable`, and so you'll need to either

```
import Data.Foldable
```

in your source, or

```
> :m +Data.Foldable
```

in `ghci`.