Solution Review: Implement Miner Interface

This lesson discusses the solution to the challenge given in the previous lesson.

Environment Variables

Key:	Value:
GOROOT	/usr/local/go
GOPATH	//root/usr/local/go/src
PATH	//root/usr/local/go/src/bin:/usr/local/go...

```
package min
type Miner interface {
        Len() int
        ElemIx(ix int) interface{}
        Less(i, j int) bool
}

func Min(data Miner) interface{}  {
        min := data.ElemIx(0)
        for i:=1; i < data.Len(); i++ {
                if data.Less(i, i-1) {
                                min = data.ElemIx(i)
                }
        }
        return min
}

type IntArray []int
func (p IntArray) Len() int                      { return len(p) }
func (p IntArray) ElemIx(ix int) interface{}  { return p[ix] }
func (p IntArray) Less(i, j int) bool            { return p[i] < p[j] }

type StringArray []string
func (p StringArray) Len() int                   { return len(p) }
func (p StringArray) ElemIx(ix int) interface{}  { return p[ix] }
func (p StringArray) Less(i, j int) bool         { return p[i] < p[j] }
```

In **min.go** in folder **min**, we implement the `Min` function (from **line 8** to **line 16**). At **line 9**, we take `min` to be the first element of the collection. Then in the for loop, we traverse the collection comparing the element at `i` index with the previous element at index `i-1`. When `data.Less(i, i-1)` is true, this means the i^{th} element is smaller, and we set `min` to the i^{th} element. After the for loop, `min` is the smallest item in the collection and we return it.

`Min` is the smallest item in the collection and we return it.

`ElemIx(ix int) interface{}` returns the element at the `ix` index. The function `Min` and the method `ElemIx` return an empty interface{}. This means that they can return an element of any type: we don't know beforehand for which collections we want to use the `Miner` interface.

At **line 18**, we define an alias `IntArray` for `[]int`, and at **line 23**, we define an alias `StringArray` for `[]string`.

From **line 19** to **line 21**, we implement `Miner` for an array of ints. From **line 24** to **line 26**, we do the same for an array of strings. Now all the work is done!

In **main.go** we simply test our work. In function `ints()`, we make an int array data at **line 8**. We convert it to a variable `a` of type `IntArray` at **line 9**. Then we can call the `Min` function from package **min** on `a`, returning the smallest element `m` from the int array. From **line 14** to **line 19**, the same steps are repeated for an array of strings.

That is it about the solution. In the next lesson, we'll come back to Go's `reflect` package but this time it will have more functionalities.