Chapter 28: Access Control

Section 28.1: Basic Example using a Struct

Version ≥ 3.0

In Swift 3 there are multiple access-levels. This example uses them all except for `open`:

```
public struct Car {

    public let make: String
    let model: String //Optional keyword: will automatically be "internal"
    private let fullName: String
    fileprivate var otherName: String

    public init(_ make: String, model: String) {
        self.make = make
        self.model = model
        self.fullName = "\(make)\(model)"
        self.otherName = "\(model) - \(make)"
    }
}
```

Assume `myCar` was initialized like this:

```
let myCar = Car("Apple", model: "iCar")
```

Car.make (public)

```
print(myCar.make)
```

This print will work everywhere, including targets that import `Car`.

Car.model (internal)

```
print(myCar.model)
```

This will compile if the code is in the same target as `Car`.

Car.otherName (fileprivate)

```
print(myCar.otherName)
```

This will only work if the code is *in the same file* as `Car`.

Car.fullName (private)

```
print(myCar.fullName)
```

This won't work in Swift 3. `private` properties can only be accessed within the same `struct`/`class`.

```
public struct Car {

    public let make: String       //public
    let model: String             //internal
    private let fullName: String! //private

    public init(_ make: String, model model: String) {
        self.make = make
```

```
        self.model = model
        self.fullName = "\(make)\(model)"
    }
}
```

If the entity has multiple associated access levels, Swift looks for the lowest level of access. If a private variable exists in a public class, the variable will still be considered private.

Section 28.2: Subclassing Example

```
public class SuperClass {
    private func secretMethod() {}
}

internal class SubClass: SuperClass {
    override internal func secretMethod() {
        super.secretMethod()
    }
}
```

Section 28.3: Getters and Setters Example

```
struct Square {
    private(set) var area = 0

    var side: Int = 0 {
        didSet {
            area = side*side
        }
    }
}

public struct Square {
    public private(set) var area = 0
    public var side: Int = 0 {
        didSet {
            area = side*side
        }
    }
    public init() {}
}
```