Készítette: Zoltán Richárd Jánki
Dátum: 16.02.2017

Apple Swift Course
Practice 2

Variables, constants:

- `var: Type` - Type type variable
  e.g.:
```
var integer: Int = 10
var double: Double = 10.0
```
- `let` - constant which can be only initalized
  e.g.:
```
let integer = 10
let double = 10.0
```
- inference: infers the type of objects

Class:

- `class ClassName { }`

Object:

- everything is object
- instantiated from a class
  e.g.: `let person = Person()`
- by the instantiation the `init()` method gets called
- if an object gets `nil` value, the `deinit()` function gets called

Optional:

- object which has a value or is `nil`
- notation: **?** and **!**
  e.g.: `var Person: Person?`

Optional chaining:

- using `if let` together
- if the Optional on the left side is `nil`, it returns with true

e.g.:
```
if let personPhone = person.cellphone?.type {   //Optional Chaining
    print(personPhone)
} else {
    print("no cellphone found...")
}
```

Automatic Reference Counting:

- memory management
- all of the created objects has a `strong` reference in the heap
- an object exists till it's necessary
- the objects that aren't used, get deleted - `deinit()`
- if an object is `nil` (Optional), then it can be removed from the heap

Strong Reference Cycle:

- the objects' reference to each other is created with a `strong` reference
- an object can't be deleted till there's a `strong` referenc to it
- **problem:** the objects have references to each other
- **solution:** use `weak/unowned` reference instead of `strong`
- when the `strong` reference is removed, the object can be removed
  e.g.: *see: **02_Swift_ARC.swift** és **02_Swift_ARC2.swift***

Weak vs. unowned:

- **`weak`** variables can be any type (Optional is!)
- **`unowned`** variables must have a value (can't be Optional!)
- both references can be removed, if there's no **`strong`** in the heap

Typealias:

- alias can be used instead of types (works for user-defined types too!)

  e.g.:
```
typealias D = Double
var double: D = 10.0
```