Optional Binding

```
/// 옵셔널 값의 강제 추출 예시 코드.

var myName: String? = "Kobe"

// 옵셔널이 아닌 변수에는 옵셔널 값이 들어갈 수 없습니다. 추출해서 할당해주어야 합니다.
var kobe: String = myName!

myName = nil
kobe = myName! // 런타임 오류!

// if 구문 등 조건문을 이용해서 조금 더 안전하게 처리해볼 수 있습니다.
if myName != nil {
  print("My name is \(myName!)")
} else {
  print("myName == nil")
}
// myName == nil
```

위에서 사용한 if 구문을 통해 myName이 nil인지 먼저 확인하고 옵셔널 값을 강제 추출하는 방법은 다른 프로그래밍 언어에서 NULL 값을 체크하는 방식과 비슷합니다.

그렇다면 옵셔널을 사용하는 의미도 사라지게 됩니다.🤦‍♂️

그래서 스위프트는 조금 더 안전하고 세련된 방법으로 옵셔널 바인딩(Optional Binding)을 제공합니다.

`옵셔널 바인딩은 옵셔널에 값이 있는지 확인할 때 사용합니다.`

만약 옵셔널에 값이 있다면 옵셔널에서 추출한 값을 일정 블록 안에서 사용할 수 있는 상수나 변수로 할당해서 옵셔널이 아닌 형태로 사용할 수 있도록 해줍니다.

옵셔널 바인딩은 if 또는 while 구문 등과 결합하여 사용할 수 있습니다.

```
/// 옵셔널 바인딩을 사용한 옵셔널 값의 추출
var myName: String? = "kobe"

// 옵셔널 바인딩을 통한 임시 '상수' 할당
if let name = myName {
  print("My name is \(name)")
} else {
  print("myName == nil")
}
// My name is kobe

// 옵셔널 바인딩을 통한 임시 '변수' 할당
if var name = myName {
  name = "minseong"
}
// My name is minseong
```

위의 코드의 예제에서는 if 구문을 실행하는 블록 안쪽에서만 name이라는 임시 상수를 사용 할 수 있습니다.

즉 if 블록 밖에서는 사용할 수 없고 else 블록에서도 사용할 수 없습니다.

그렇기 때문에 위와 아래에서 모두 별도로 name을 사용했지만 충돌이 일어나지 않았습니다.

또, 상수로 사용하지 않고 변수로 사용하고 싶다면 if var를 통해 임시 변수로 할당할 수도 있습니다.

위 코드에서는 if와 else 블록만을 사용했지만, else if 블록도 추가할 수 있습니다.

`옵셔널 바인딩을 통해 한 번에 여러 옵셔널의 값을 추출할 수도 있습니다.`

쉼표(,)를 사용해 바인딩 할 옵셔널을 나열하면 됩니다.

단, 바인딩하려는 옵셔널 중 하나라도 값이 없다면 해당 블록 내부의 명령문은 실행되지 않습니다.

```
/// 옵셔널 바인딩을 사용한 여러 개의 옵셔널 값의 추출
var myName: String? = "kobe"
var yourName: String? = nil

// friend에 바인딩이 되지 않으므로 실행되지 않습니다.
if let name = myName, let friend = yourName {
  print("We are friend! \(name) & \(friend)")
}

yourName = "mark"

if let name = myName, let friend = yourName {
  print("We are friend! \(name) & \(friend)")
}
// We are friend! kobe & mark
```

옵셔널 바인딩은 옵셔널 체이닝과 환상의 결합을 이룹니다.