FABERNOVEL

Identifiers all the way

1 Unifying the concept of identified objects

```
struct Something {
    let id: Int
    /* other properties */
}
```

```swift
struct Something: Identifiable {
    let id: Int
    /* other properties */
}
```

```swift
protocol Identifiable {
    var id: Int { get }
}
```

```swift
struct Something: Identifiable {
    let id: Int
    /* other properties */
}
```

```swift
protocol Identifiable {
    var id: Int { get }
}
```

```swift
extension Collection where Element: Identifiable {
    func first(identifiedBy id: Int) -> Element? {
        first { $0.id == id }
    }
}
```

```swift
struct Something: Identifiable {
    let id: Int
    /* other properties */
}


struct SomethingPainful: Identifiable {
    let id: String
    /* other properties */
}


protocol Identifiable {
    var id: Int { get }
}


extension Collection where Element: Identifiable {
    func first(identifiedBy id: Int) -> Element? {
        first { $0.id == id }
    }
}
```

```swift
struct Something: Identifiable {
    let id: Int
    /* other properties */
}
```

Type 'SomethingPainful' does not conform to protocol 'Identifiable'

```swift
struct SomethingPainful: Identifiable {
    let id: String
    /* other properties */
}
```

```swift
protocol Identifiable {
    var id: Int { get }
}
```

```swift
extension Collection where Element: Identifiable {
    func first(identifiedBy id: Int) -> Element? {
        first { $0.id == id }
    }
}
```

```swift
struct Something: Identifiable {
    let id: Int
    /* other properties */
}


struct SomethingPainful: Identifiable {
    let id: String
    /* other properties */
}

protocol Identifiable {
    associatedtype Identifier: Hashable
    var id: Identifier { get }
}

extension Collection where Element: Identifiable {
    func first(identifiedBy id: Int) -> Element? {
        first { $0.id == id }
    }
}
```

```
protocol Identifiable {
    associatedtype Identifier: Hashable
    var id: Identifier { get }
}
```

```swift
protocol Identifiable {
    associatedtype Identifier: Hashable
    var id: Identifier { get }
}
```

```swift
func presentArticleDetail(forArticleId id: String)
```

```swift
protocol Identifiable {
    associatedtype Identifier: Hashable
    var id: Identifier { get }
}
```

```swift
func presentArticleDetail(forArticleId id: String)
```

```swift
func presentArticleDetail(for id: Article.ID)
```

2 ○ Safely manipulate identifier

```swift
struct Identifier<Value: Hashable>: Hashable {
    private let id: Value

    init(_ value: Value) {
        self.id = value
    }
}
```

```swift
struct Identifier<Value: Hashable>: Hashable {
    private let id: Value

    init(_ value: Value) {
        self.id = value
    }
}
```

```swift
struct Dog: Identifiable {
    let id: Identifier<String>
}
```

```swift
struct Cat: Identifier {
    let id: Identifier<String>
}
```

```swift
struct Identifier<Value: Hashable>: Hashable {
    private let id: Value

    init(_ value: Value) {
        self.id = value
    }
}


struct Dog: Identifiable {
    let id: Identifier<String>
}




struct Cat: Identifier {
    let id: Identifier<String>
}




func call(byItsName id: Cat.ID) -> Cat {}
func call(byItsName id: Dog.ID) -> Dog {}
```

```swift
struct Identifier<Value: Hashable, Model>: Hashable {
    private let id: Value

    init(_ value: Value) {
        id = value
    }
}


struct Dog: Identifiable {
    let id: Identifier<String, Dog>
}




struct Cat: Identifier {
    let id: Identifier<String, Cat>
}




func call(byItsName id: Cat.ID) -> Cat {}
func call(byItsName id: Dog.ID) -> Dog {}
```

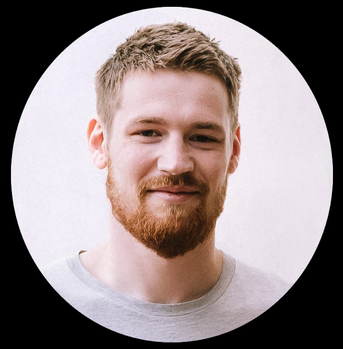
3 Identifying the right way

```swift
protocol Identifiable {
    var id: Int { get }
}
```

```swift
protocol Identifiable {
    associatedtype Identifier: Hashable
    var id: Identifier { get }
}
```

```swift
struct Identifier<Value: Hashable, Model>: Hashable {
    private let id: Value

    init(_ value: Value) {
        id = value
    }
}
```



Denis Poifol

FABERNOVEL

THANK YOU