Unit 4—Lesson 1: Protocoles

Protocoles

Un protocole défini un ensemble de méthodes, propriétés et autre prérequis qui sont nécéssaires pour accomplir une tâche, ou une partie de fonctionnalité particulière.

Protocoles

La bibliothèque standard de Swift défini plusieurs protocoles, dont :

`CustomStringConvertible`

`Equatable`

`Comparable`

`Codable`

Quand vous adoptez un protocole, vous devez implémenter toutes les méthodes requises.

Print avec CustomStringConvertible

```
let string = "Hello, world!"
print(string)

let number = 42
print(number)

let boolean = false
print(boolean)
```

```
Hello, world!
42
false
```

Print avec CustomStringConvertible

```
class Shoe {
  let color: String
  let size: Int
  let hasLaces: Bool

  init(color: String, size: Int, hasLaces: Bool) {
        self.color = color
        self.size = size
        self.hasLaces = hasLaces
    }
}

let myShoe = Shoe(color: "Black", size: 12, hasLaces: true)
print(myShoe)
```

```
__lldb_expr_1.Shoe
```

```
class Shoe: CustomStringConvertible {
    let color: String
    let size: Int
    let hasLaces: Bool

    init(color: String, size: Int, hasLaces: Bool) {
        self.color = color
        self.size = size
        self.hasLaces = hasLaces
    }




}
```

```
class Shoe: CustomStringConvertible {
    let color: String
    let size: Int
    let hasLaces: Bool

    init(color: String, size: Int, hasLaces: Bool) {
        self.color = color
        self.size = size
        self.hasLaces = hasLaces
    }

    var description: String {
        return "Shoe(color: \(color), size: \(size), hasLaces: \(hasLaces))"
    }
}
let myShoe = Shoe(color: "Black", size: 12, hasLaces: true)
print(myShoe)
```

```
Shoe(color: Black, size: 12, hasLaces: true)
```

Créer un protocole

```
protocol FullyNamed {
  var fullName: String { get }

  func sayFullName()
}

struct Person: FullyNamed {
  var firstName: String
  var lastName: String
}
```

Créer un protocole

```
struct Person: FullyNamed {
  var firstName: String
  var lastName: String

  var fullName: String {
    return "\(firstName) \(lastName)"
  }

  func sayFullName() {
    print(fullName)
  }
}
```

Délégation

Permet à une classe ou une structure de déléguer des responsabilités à une instance d'une autre type

```
protocol ButtonDelegate {
  func userTappedButton(_ button: Button)
}

class GameController: ButtonDelegate {

  func userTappedButton(_ button: Button) {
    print("User tapped the \(button.title) button.")
 }
}
```

Délégation
Exemple GameController (suite)

```
class Button {
  let title: String
  var delegate: ButtonDelegate? // Ajouter une propriété delegate au Button

  init(title: String) {
    self.title = title
  }

  func tapped() {
    self.delegate?.userTappedButton(self) // Si le delegate existe, appelle la fonction
                                          // déléguée `userTappedButton` sur le délégué
  }
}
```

Délégation
Exemple GameController (suite)

```
let startButton = Button(title: "Start Game")
let gameController = GameController()
startButton.delegate = gameController

startButton.tapped()
```

Unit 4—Lesson 1
Lab: Protocols

Open and complete the exercises in `Lab - Protocols.playground`