TP 7
Fiches

À la fin de ce TP :

- Faire une archive contenant les projets Xcode des exercices
- Envoyer l'archive à ahumiliere@captaintrain.com avec l'objet : `[DANT] TP 7 – Prénom Nom`
- Si le TP est fait à plusieurs, préciser les noms et adresses mail de chacun

Le TP se base sur un nouveau projet Xcode à créer, de type « Single View Application ».

Exercice 1

- L'application va d'abord présenter la couverture d'une fiche (Flashcard en anglais), avec un terme particulier.
- Ajouter un label à la vue et changer son texte en **Flashcard Term**. Ajouter les contraintes autolayout.
- Sur iOS, le composant UINavigationController permet de naviguer simplement entre différents view controllers. Il agit comme un conteneur, qui gère la navigation entre ses view controllers.
- Dans Interface Builder, ajouter un Navigation Controller au Storyboard. IB ajoute automatiquement un table view controller. Le supprimer.
- Déplacer la flèche pointant sur le premier view controller vers le navigation controller, pour indiquer qu'il s'agit du view controller initial de l'application.
- Glisser-Déposer avec le bouton droit du navigation controller vers le view controller et sélectionner le *Relationship Segue* `rootViewController`.
- Sélectionner la navigation bar en haut du view controller et, avec l'Attributes Inspector, changer le titre en **Term**.
- Renommer le view controller en **Term Controller**.
- Ajouter un nouveau Bar Button Item dans la navigation bar de Term Controller et changer son titre en **Definition**.
- Lancer l'application et observer le résultat.
- Ajouter un autre view controller au Storyboard. Le renommer en **Definition Controller**. Lui ajouter une Text View pouvant contenir plusieurs lignes de texte. Ajouter les contraintes autolayout.
- Glisser-Déposer avec le bouton droit du bouton Definition vers Definition Controller. Sélectionner le segue **show**. et observer comment Interface Builder représente ce nouveau lien. Les segue représentent des transitions entre deux view controllers.
- Sélectionner la navigation bar en haut du view controller et, avec l'Attributes Inspector, changer le titre en **Definition**.

- Lancer l'application et observer le résultat.

Exercice 2

- Créer une nouvelle classe `Flashcard` qui permettra d'encapsuler un terme et sa définition. L'objet Flashcard sera initialisé avec un terme et une définition. Prévoir un initialiser vide, affectant des textes par défaut pour le terme et sa définition.
- Ajouter un outlet pour le label *Flashcard Term* dans `ViewController`.
- Mettre à jour le label dans viewDidLoad, en utilisant les valeurs par défaut de `Flashcard`.

```
override func viewDidLoad() {
    super.viewDidLoad()
    let flashcard = Flashcard()
    termLabel.text = flashcard.term
}
```

- Lancer l'application et observer le résultat.

Exercice 3

- Créer une nouvelle classe `Deck`, qui représentera une collection d'objets `Flashcard`. Plutôt que d'accéder directement à une collection d'objets `Flashcard`, le controller utilisera des méthodes pour les demander à l'objet `Deck`.
- Ajouter une propriété privée `[Flashcard]`. Elle est privée masquée comment la classe `Deck` gère en interne sa collection d'objets.

```
private var cards = [Flashcard]()
```

- Implémenter une méthode `init` sans paramètres, initialisant `Deck` avec des données par défaut :

```
init() {
   let cardData = [
      "controller outlet" : "A controller view property, marked with
         IBOutlet.",
      "controller action": "A controller method, marked with IBAction,
         that is triggered by an interface event."
   ]
   for (term, definition) in cardData {
      cards.append(Flashcard(term: term, definition: definition))
   }
}
```

- Remplacer la boucle `for in` par un appel à la fonction `map()`. Aboutir à une syntaxe courte et élégante.
- Modifier la propriété cards pour en faire une constante, sans valeur par défaut.

Exercice 4

- Par soucis de cohérence, renommer **ViewController.swift** en **TermController.swift** et mettre à jour le nom de la classe en `TermController`.
- Dans Interface Builder, sélectionner le Term Controller et utiliser l'Identity Inspector pour changer la **Custom Class** à `TermController`.
- Ajouter une propriété `Deck` à la classe `TermController`.

```
let deck = Deck()
```

- La méthode `viewDidLoad` de `TermController` va afficher une `Flashcard` aléatoire parmi celles de `deck`. Ajouter une méthode `randomCard` à la classe `Deck`, renvoyant aléatoirement un des objets de la collection. Si le deck est vide, cette méthode devra renvoyer `nil`.
- Dans `TermController`, modifier l'implémentation de `viewDidLoad` pour utiliser la méthode `randomCard`.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let flashcard = deck.randomCard {
      termLabel.text = flashcard.term
   }
}
```

- Lancer l'application plusieurs fois et observer le résultat.

Exercice 5

- Ajouter au projet une nouvelle classe `DefinitionController`, héritant de `UIViewController`.
- Dans Interface Builder, sélectionner le Definition Controller et utiliser l'Identity Inspector pour changer la **Custom Class** à `DefinitionController`.
- Lancer l'application et naviguer jusqu'à la définition. Vérifier que le texte par défaut s'affiche toujours.
- Term controller a besoin de transmettre un objet `Flashcard` à `DefinitionController`.
- Ajouter une propriété de type `Flashcard` à la classe `DefinitionController`. Marquer cette propriété comme optionnelle : elle pourra contenir un objet de type `Flashcard` ou `nil`.

- Ajouter un `IBOutlet` pour la text view dans `DefinitionController`.
- Implementer `viewDidLoad` dans `DefinitionController`, pour utiliser le contenu de la propriété `flashcard`. Ici, `if let` permet de s'assurer que flashcard ne contient pas `nil`.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let card = flashcard {
      definition.text = card.definition
   }
}
```

- Dans la documentation Apple, étudier la méthode `prepareForSegue:sender:` de `UIViewController` et les propriétés `sourceViewController` et `destinationViewController` de `UIStoryboardSegue.`
- Ajouter une propriété de type `Flashcard` à la classe `TermController`.
- Mettre à jour la méthode `viewDidLoad` pour stocker la carte affichée dans cette propriété.

```
override func viewDidLoad() {
   super.viewDidLoad()
   if let flashcard = deck.randomCard {
      self.flashcard = flashcard
      termLabel.text = flashcard.term
   }
}
```

- Implementer la méthode `prepareForSegue:sender:` de la classe `TermController`.

```
override func prepareForSegue(segue: UIStoryboardSegue,
   sender: AnyObject?) {
   if let definitionController =
      segue.destinationViewController as? DefinitionController {
      definitionController.flashcard = flashcard
   }
}
```

- Lancer l'application plusieurs fois et observer le résultat.

Exercice 6

- On souhaite maintenant afficher un nouveau terme à chaque fois que le view controller TermController apparait. En utilisant la documentation de UIViewController, trouver une solution pour le faire et l'implémenter.