TP 1
Découverte d'Xcode

L'objectif de ce TP est de se familiariser avec Xcode et Interface Builder, en réalisant des changements sur un projet et en lançant l'application sur le simulateur.

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 1 – Prénom Nom`
- Si le TP est fait à plusieurs, préciser les noms et adresses mail de chacun

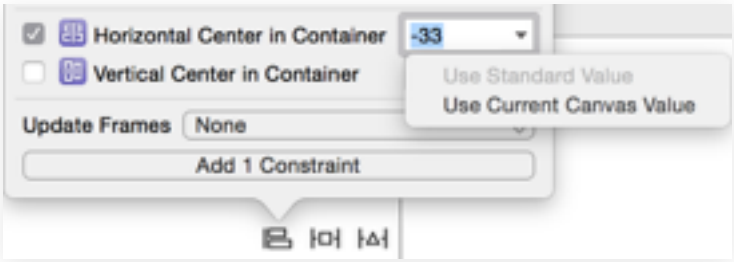
Exercice 1

- Ouvrir et lancer (⌘**R**) le projet **WordCollage**.
- En utilisant le *Project Navigator* (⌘**1**), explorer **Main.storyboard**.
- En utilisant le bouton *Show Document Outline* (▭) en bas à gauche du canvas, s'assurer que les détails du document sont visibles.
- Double-cliquer sur un Label dans le collage pour changer son contenu.
- Lancer l'application (⌘**R**) et visualiser les changements dans le Simulateur iOS.
- Expérimenter différents changements de contenu et de forme sur les labels.
- Lancer l'application et visualiser les changements dans le Simulateur iOS.

Exercice 2

- Afficher **Main.storyboard**.
- Lancer l'application et observer comment le layout change entre le Storyboard et l'aperçu dans le Simulateur.
- Avec la Librairie (⌥⌘**L**), placer un nouveau Label dans l'interface. Changer le contenu du Label (par ex. « Licence DANT ») et utiliser l'inspecteur (⌥⌘**4**) pour changer sa police, sa taille et sa couleur.
- Utiliser les contrôles du Label pour étendre sa taille et ajuster la position du Label.
- Lancer l'application et observer comment la position du Label apparait different dans le Simulateur.
- Avec le Label sélectionné, utiliser le bouton Pin (en bas à droite) pour sélectionner une contrainte *Vertical Space* relative à la vue (voir image ci-dessous).
  Les contraintes sont représentées par une ligne bleue. Si une contrainte est ambiguë ou manquante, elle est représentée en orange.

- Avec le Label sélectionné, utiliser le bouton Align pour sélectionner une contrainte d'alignement *Horizontal Center* basée sur la position courante du Label.



- En utilisant le bouton *Show Document Outline* (▭) en bas à gauche du canvas, s'assurer que le contenu du document est toujours visible.
- Remarquer comment Interface Builder affiche une contrainte manquante en orange et utiliser l'*Issue Navigator* (⌘**4**) ou le bouton *Document Outline* (🟠) pour comprendre ce qui manque dans le layout.
- Avec le Label sélectionné, utiliser le menu *Editor* > *Resolve Auto Layout Issues* > *Update Frames* (⌥⌘=) pour que la frame s'adapte aux contraintes. Il est aussi possible d'utiliser le menu *Editor* > *Resolve Auto Layout Issues* > *Update Constraints* (⇧⌘=) pour que les contraintes s'adaptent à la frame.
- Lancer l'application et observer comment la position du Label s'adapte.
- Utiliser Interface Builder pour sélectionner la *size class* Compact Width | Regular Height.
- Tout en visualisant le canvas dans Interface Builder, ouvrir l'Assistant (⌥⌘↩) et utiliser la barre du haut pour sélectionner le Preview.
- Supprimer l'iPhone 4" par défaut et utiliser le bouton *Add* en bas à gauche du *Preview* pour ajouter un iPhone 4,7".
- Dans le canvas Interface Builder, sélectionner le Label ajouté précédemment. Ajuster sa position, mettre à jour ses contraintes (⇧⌘=) et observer comment le preview se met automatiquement à jour pour refléter ces changements.
- Lancer l'application et observer comment le label apparait comme souhaité dans le Simulateur.
- Faire tourner le simulateur (⌘⇢) et observer comment la position du Label évolue en orientation horizontale.
- En utilisant Interface Builder, sélectionner Any Width | Compact Height comme *size class* et changez l'orientation du preview.

- Sélectionner le Label ajouté récemment et ajuster sa position. Mettre à jour les contraintes et observer les changements dans le preview.
- Lancer l'application et observer comment le label apparait comme souhaité dans le Simulateur.

Exercice 3

- Avec Interface Builder, revenir sur la *size class* Any Width | Compact Height et ajouter un bouton sur l'interface
- Modifier le text du bouton pour afficher "Changer l'arrière plan »
- Lancer l'application et observer comment le bouton apparait sur le Simulateur.
- Avec Interface Builder, faire un drag-drop avec clic-droit du Bouton vers la vue en dessous et sélectionner *Bottom Space to Bottom Layout Guide* pour créer une contrainte. Reproduire l'opération et sélectionner *Bottom Space to Bottom Layout Guide.*
- Avec le bouton toujours sélectionné, utiliser *Align control* et sélectionner *Horizontal Center in Container* pour créer une autre contrainte.
- Lancer l'application, taper sur le bouton et observer le résultat *(spoiler : aucun résultat)*.
- Ouvrir l'assistant d'Interface Builder (⌥⌘↩).
- Avec la section Document Outline, cliquer-droit sur le bouton et tirer une connection vers le controller, pour créer un Action. Utiliser le nom `changeBackgroundColor` et le type `UIButton`.

```
@IBAction func changeBackgroundColor(sender: UIButton) {
}
```

- Essayer de retirer l'attribut `@IBAction` et remarquer que la connection avec le bouton disparait. Annuler le changement, la connection réapparait.
- Implémenter la méthode `changeBackgroundColor:`

```
@IBAction func changeBackgroundColor(sender: UIButton) {
    view.backgroundColor = UIColor.blackColor()
}
```

- Avec la documentation Xcode et la référence de l'API (⇧⌘**0**), trouver plus d'informations sur la classe `UIColor` et les couleurs « faciles » à utiliser.
- Lancer l'application et tester le bouton.

Exercice 4

- Changer le label du bouton en « Black ».
- Avec Interface Builder et la Librairie, ajouter un bouton « White »
- Placer les contraintes pour que le bouton se positionne correctement, quel que soit l'orientation de l'application.
- Ajouter un autre bouton « Magenta », ajouter les contraintes correspondantes.
- Etablir les connections avec deux nouvelles méthodes sur le controller, `changeBackgroundColorToWhite:` et `changeBackgroundColorToMagenta:`.

```
@IBAction func changeBackgroundColorToWhite(sender: UIButton) {
}

@IBAction func changeBackgroundColorToMagenta(sender: UIButton) {
}
```

• Implementer ces deux méthodes.

```
@IBAction func changeBackgroundColorToWhite(sender: UIButton) {
    view.backgroundColor = UIColor.whiteColor()
}

@IBAction func changeBackgroundColorToMagenta(sender: UIButton) {
    view.backgroundColor = UIColor.magentaColor()
}
```

• Renommer `changeBackgroundColor:` en `changeBackgroundColorToBlack:`, observer le résultat sur la connection de cette méthode.

• Lancer l'application, taper sur le bouton *Black*, et observer le résultat.

• L'application a crashé parce qu'Interface Builder essaye toujours de connecter le bouton avec la méthode `changeBackgroundColor:` qui n'existe plus.

• Supprimer l'ancienne connexion et en établir une nouvelle avec `changeBackgroundColorToBlack:`. Observer la connection réapparaitre.

• Lancer l'application et tester les différents boutons.