Kotlin : les bases

🤾‍♀️ Une startup toulousaine qui organise des concerts sur Toulouse vient de vous recruter. Elle vous a confié des premières missions pour vous familiariser avec Kotlin. Vous allez manipuler des variables, des collections, des classes et des tests unitaires tout en utilisant git pour versionner votre code. À la fin de ce TP, vous aurez la responsabilité de développer une application interactive (via le terminal) pour gérer un festival de musique.

> **Note**
>
> À chaque fois que vous croisez une indication :
>
> ```
> fix #x.y message
> ```
>
> Commit + push sur votre dépôt git avec le message “fix #x.y : message”.

Exercice 0 : installation de l’environnement

Sur l’IDE IntelliJ IDEA, initialisez un projet Kotlin. Créez un fichier `Main.kt` et affichez “Hello, World!” dans la console. Lancez le programme pour vérifier que tout fonctionne correctement.

Exercice 1 : variables et types

Exercice 1.1

Dans le fichier `Main.kt`, déclarez trois variables immuables représentant des salles de concert à Toulouse :

- `bikini` : contenant “Le Bikini”
- `zenith` : contenant “Zenith Toulouse”
- `rex` : contenant “Le Rex”

Affichez ensuite ces variables avec la fonction `println()` au format `"Le Bikini - Zenith Toulouse - Le Rex"`.

Exercice 1.2

Toujours dans le fichier Main.kt, écrivez une fonction `checkGenre` prenant en paramètre une chaîne de caractères représentant un genre musical. Si le genre est `"rap"` ou `"techno"`, la fonction doit afficher `"Genre $genre supporté"`, sinon `"Genre $genre non supporté"`.

Testez cette fonction avec les genres `"rap"`, `"techno"`, et `"rock"`.

Bonus : les `enum` existent aussi en Kotlin, vous pouvez les utiliser pour définir les différents genres…Il faudra alors adapter la fonction `checkGenre` pour prendre en paramètre un `enum` à la place d’une chaîne de caractères.

> **Note**
>
> À partir de maintenant, vous pouvez créer un package `exos` dans lequel vous allez ajouter un fichier par exercice (2, 3). Dans chaque fichier (ex: `exo2.kt`), vous pouvez déclarer les sous-exercices (2.1, 2.2, etc.) puis appeler les fonctions correspondantes dans le fichier `Main.kt`.

Exercice 2 : collections et boucles

Exercice 2.1

Créez une liste mutable contenant les noms de trois artistes francophones (par exemple `"Houdi"`, `"Luther"`, `"Laylow"`).
Ajoutez deux autres noms à la liste et afficher tous les noms avec une boucle `for`.

Exercice 2.2

Créez une map mutable associant des salles de concert à leur capacité maximale :

- `"Le Bikini"` -> `1500`
- `"Zenith Toulouse"` -> `11000`
- `"Le Rex"` -> `800`

Ajoutez une salle supplémentaire avec une capacité de votre choix. Vous allez parcourir la map pour afficher chaque salle et sa capacité au format suivant :

```
Nom de la salle - Capacité
```

Exercice 3 : fonctions, classes et tests

Exercice 3.1

Déclarez une classe `Concert` avec les propriétés suivantes :

- `nomArtiste` (String)
- `date` (Date)
- `salle` (String)

Ajoutez une fonction `description` qui retourne une chaîne de caractères formatée comme suit : `"Le concert de {nomArtiste} aura lieu le {date} à la salle {salle}"`.

Exercice 3.2

Instanciez deux objets `Concert` avec des données fictives et appelez leur méthode `description`. Affichez le résultat dans la console.

Exercice 3.3

Ajoutez la propriété `prixTicket` de type Int qui peut être `null` à la classe `Concert`.
Modifiez la méthode `description` pour afficher le prix du ticket si celui-ci est défini. Sinon, affichez `"Prix non défini"`.

Exercice 3.4

Écrivez un test unitaire utilisant JUnit pour vérifier que la méthode `description` fonctionne correctement pour un objet `Concert`.

Exercice 4 : projet

Pour ce projet[1], l'équipe vous a confié la responsabilité du Weekend des Curiosités, un festival de musique avec des artistes émergents qui se déroule au Bikini. Vous allez construire une application interactive permettant de gérer les différents aspects du festival :

- Gérer les artistes et le planning des concerts
- Organiser les diverses animations et stands du festival
- Calculer le prix des tickets en fonction des demandes des artistes et des premières statistiques des réseaux sociaux

> **Note**
>
> Vous devez récupérer le repo git suivant : BUT3-projet-weekend-curiosites.
>
> Pour chaque exercice, il faut que des tests unitaires soient écrits pour valider les fonctionnalités développées. Pour rappel, les tests doivent être placés dans le dossier `src/test/kotlin` du projet.

Exercice 4.1 - Les artistes et les concerts

Étape 1 : Gestion des artistes
Créez une classe `Artiste` avec les propriétés suivantes :

- `nom` (String)
- `genre` (String)
- `popularite` (Int) : une valeur entre 1 et 100 représentant la popularité de l'artiste.

Ajoutez une méthode `description` qui retourne une chaîne de caractères comme suit : `"{nom} est un artiste de {genre} avec une popularité de {popularite}/100."`

[1] Source image: site web du Weekend des Curiosités

Implémentez un menu pour ajouter, afficher et supprimer des artistes dans une liste mutable.
Exemple d'options :
- `1. Ajouter un artiste`
- `2. Afficher tous les artistes`
- `3. Supprimer un artiste par son nom`

Étape 2 : Gestion des concerts
Créez une classe `Concert` avec les propriétés suivantes :
- `artiste` (Artiste)
- `date` (String) : au format "dd/MM/yyyy".
- `heure` (String) : au format "HH:mm".
- `duree` (Int) : durée en minutes.

Ajoutez une méthode `description` qui retourne une chaîne comme suit : `"Le concert de {artiste.nom} ({artiste.genre}) aura lieu le {date} à {heure} pour une durée de {duree} minutes."`

Ajoutez un menu interactif pour gérer une liste de concerts, avec des options comme :
- `1. Ajouter un concert`
- `2. Afficher tous les concerts`
- `3. Supprimer un concert par son artiste`

fix #4.1 (projet): artistes et concerts

Exercice 4.2 - Planning du festival

Étape 1 : Planification
Créez une classe `Planning` avec une liste mutable `concerts` (initialement vide). Ajoutez les méthodes suivantes :
- `ajouterConcertPlanning` pour ajouter un concert au planning.
- `afficherPlanning` pour afficher tous les concerts triés par date et heure.

Intégrez ces fonctionnalités dans un menu interactif :
- `1. Ajouter un concert au planning`
- `2. Afficher le planning trié`

Étape 2 : Vérification des conflits
Ajoutez une méthode `verifierConflits` à la classe `Planning` pour détecter les concerts qui se chevauchent (même date et horaires qui se croisent). Affichez les concerts en conflit dans le menu.

Cette fonction n'est pas triviale, n'hésitez pas à demander de l'aide si vous êtes bloqué.e

L'application commence à prendre forme, mais il manque une gestion des erreurs lors des entrées utilisateur. À l'aide d'un `try-catch` dans votre `Main.kt`, gérez les exceptions pour éviter les "crash" de l'application.

fix #4.2 (projet): planning

Exercice 4.3 - Animations et stands

Étape 1 : Gestion des stands
Créez une classe `Stand` avec les propriétés suivantes :
- `nom` (String)
- `type` (Enum à créer) : par exemple, “FOOD”, “MERCH”, “OTHER”.

Ajoutez une méthode `description` qui retourne une chaîne comme suit : `"Le stand {nom} propose des services de type {type}."`

Ajoutez un menu pour gérer une liste mutable de stands :
- `1. Ajouter un stand`
- `2. Afficher tous les stands`
- `3. Supprimer un stand par son nom`

Étape 2 : Gestion du site
Créez une classe `Site` qui regroupe les salles et les stands :
- Propriété `salles` (`List<String>`) : une liste de noms de salles avec par défaut les salles : “Le Bikini”, “Zenith Toulouse”, “Le Rex”.
- Propriété `stands` (`MutableList<Stand>`) : les stands disponibles.

Ajoutez une méthode `afficherSite` pour afficher les salles et les stands associés puis l’ajouter dans le menu.

```
fix #4.3 (projet): animations et stands
```

Exercice 4.4 - Calcul des prix et statistiques

Étape 1 : Calcul des prix
Ajoutez une méthode `calculerPrix` dans la classe `Concert`, basée sur la popularité de l’artiste :
- Moins de 50 : 1000€
- Entre 50 et 75 : 2000€
- Plus de 75 : 5000€

Ajoutez une option dans le menu principal pour afficher le prix des artistes.

Étape 2 : Analyse des données
Créez une classe `Statistiques` avec les fonctionnalités suivantes :
- `analyserDemandes` : Retourne le genre musical le plus populaire.
- `calculerPrixMoyen` : Retourne le prix moyen des tickets.

Intégrez un menu pour afficher ces statistiques.

```
fix #4.4 (projet): prix et statistiques
```