Travaux Pratiques Kotlin
Variables – Collections – POO – Héritage

Pr. Lamia ZIAD

EST Essaouira

I. Travaux Pratiques : Kotlin — Variables, Listes, Tableaux et Classes

Exercice 1 : Manipulation des Variables

Objectif :

— comprendre les types de variables en Kotlin ;

— effectuer des opérations simples.

Consignes :

1. Déclarer une variable `nom` de type `String` contenant votre prénom.
2. Déclarer une variable `age` de type `Int`.
3. Afficher : *"Je m'appelle [nom] et j'ai [age] ans."*
4. Créer une variable `anneeNaissance` calculée automatiquement.

Exemple de code :

```
val nom: String = "Ahmed"
val age: Int = 20
println("Je␣m'appelle␣$nom␣et␣j'ai␣$age␣ans.")

val anneeNaissance = 2025 - age
println("Annee␣de␣naissance␣:␣$anneeNaissance")
```

Exercice 2 : Listes et Tableaux

Consignes :

1. Créer une liste mutable contenant trois noms de villes.
2. Ajouter une ville à la liste.
3. Afficher chaque ville avec une boucle.
4. Créer un tableau contenant : 10, 20, 30, 40, 50.
5. Calculer et afficher la somme des éléments.

Exemple de code :

```
val villes = mutableListOf("Rabat", "Casablanca", "Agadir")
villes.add("Essaouira")

for (v in villes) {
    println(v)
}

val tableau = arrayOf(10, 20, 30, 40, 50)
val somme = tableau.sum()
println("Somme = $somme")
```

Exercice 3 : Classes et Objets

Consignes :

1. Créer une classe `Personne(nom, age)`.
2. Ajouter une méthode `sePresenter()`.
3. Instancier deux objets et appeler la méthode.

Exemple de code :

```
class Personne(val nom: String, val age: Int) {
    fun sePresenter() {
        println("Bonjour, je m'appelle $nom et j'ai $age ans.")
    }
}

val p1 = Personne("Sara", 22)
val p2 = Personne("Yassine", 30)

p1.sePresenter()
p2.sePresenter()
```

Exercice 4 : Héritage et Interfaces

Consignes :

- Classe `Animal` avec propriété `nom` et méthode `faireDuBruit()`.
- Classe `Chien` : afficher “Ouaf Ouaf !”.
- Classe `Chat` : afficher “Miaou !”.
- Tester les deux objets.

Exemple de code :

```
open class Animal(val nom: String) {
    open fun faireDuBruit() {
        println("Bruit inconnu")
    }
}

class Chien(nom: String) : Animal(nom) {
    override fun faireDuBruit() {
        println("Ouaf Ouaf !")
    }
}
```

```
class Chat(nom: String) : Animal(nom) {
    override fun faireDuBruit() {
        println("Miaou!")
    }
}

val c1 = Chien("Rex")
val c2 = Chat("Mimi")

c1.faireDuBruit()
c2.faireDuBruit()
```

II. Travaux Pratiques : Programmation Orientée Objet en Kotlin

Titre : Gestion des véhicules

Objectifs :

— comprendre les classes et l'héritage ;

— manipuler les interfaces ;

— créer et utiliser des objets.

Consignes — Architecture demandée

1. Classe Vehicule :

— propriétés : `marque`, `modele`, `vitesseMax` ;

— méthode : `afficherDetails()`.

2. Interface Reparable :

— méthode : `reparer()`.

3. Héritage :

— `Voiture` hérite de `Vehicule` + propriété `nombrePortes` ;

— `Moto` hérite de `Vehicule` + propriété `cylindree` ;

— les deux implémentent `Reparable`.

4. Programme principal :

— créer une voiture et une moto ;

— afficher les détails ;

— simuler une réparation.

Exemple de code (structure attendue)

```
open class Vehicule(
    val marque: String,
    val modele: String,
    val vitesseMax: Int
) {
    open fun afficherDetails() {
        println("Vehicule: $marque $modele, Vitesse max: $vitesseMax km/h"
            )
    }
}

interface Reparable {
    fun reparer()
}

class Voiture(
    marque: String,
    modele: String,
    vitesseMax: Int,
    val nombrePortes: Int
) : Vehicule(marque, modele, vitesseMax), Reparable {
```

```
    override fun afficherDetails() {
        super.afficherDetails()
        println("Nombre de portes: $nombrePortes")
    }

    override fun reparer() {
        println("La voiture $marque $modele est en reparation.")
    }
}

class Moto(
    marque: String,
    modele: String,
    vitesseMax: Int,
    val cylindree: Int
) : Vehicule(marque, modele, vitesseMax), Reparable {

    override fun afficherDetails() {
        super.afficherDetails()
        println("Cylindree: ${cylindree}cc")
    }

    override fun reparer() {
        println("La moto $marque $modele est en reparation.")
    }
}

fun main() {
    val voiture = Voiture("Toyota", "Corolla", 180, 4)
    val moto = Moto("Yamaha", "R1", 299, 1000)

    println("---- Details des vehicules ----")
    voiture.afficherDetails()
    moto.afficherDetails()

    println("\n---- Reparation ----")
    voiture.reparer()
    moto.reparer()
}
```