# Les attributs d'une classe dans Dart

## Objectis

- Savoir déclarer des attributs de différentes types (entiers, chaînes, virgule flottante, etc.).
- Comprendre comment initialiser les attributs lors de la déclaration ou via des constructeurs.

1. ## Présentation

    - Dans une classe, les attributs pourraient être considérés comme les caractéristiques de l'objet. Une personne possède un nom, un prénom, une date de naissance… Une voiture, elle, a une couleur, une marque, un modèle…
    - À l'intérieur de la classe, l'enjeu va être de déclarer ces attributs de la même façon que des variables quelconques. Ils possèderont un type et seront explicitement nommés. Toutefois, il existe une particularité à respecter, l'encapsulation.
    - Il s'agit de protéger ces éléments du contact extérieur afin de garantir que ce qu'ils contiennent ne subira pas de modifications sauvages.
    - Pour garantir cette intégrité, les attributs doivent être déclarés privés ; c'est-à-dire qu'ils ne seront exploitables que dans la classe où ils se trouvent. Dans beaucoup de langages, il existe des mots-clés pour cela comme private en Java.
    - En Dart, il n'y en a pas. En revanche, il est possible de mettre en préfixe du nom de l'attribut un underscore (tiret bas) qui produira le même effet.
    - Voici la classe créée précédemment à laquelle les attributs nom et prénom ont été ajoutés :

```
class Personne {
 String nom;
 String _prenom;
}
```

2. Ces deux attributs sont des chaînes de caractères. Ils s'appellent nom et _prenom et possèdent respectivement une portée publique et privée (due à la présence de l'underscore).
3. Théoriquement, les deux attributs auraient dû être privés. Mais dans la suite de l'exemple, il sera utile de constater la différence quand une personne sera créée dans un autre fichier.
4. Justement, comment créer une personne ? Il existe un outil autorisant cela et son nom est des plus explicite : le constructeur.

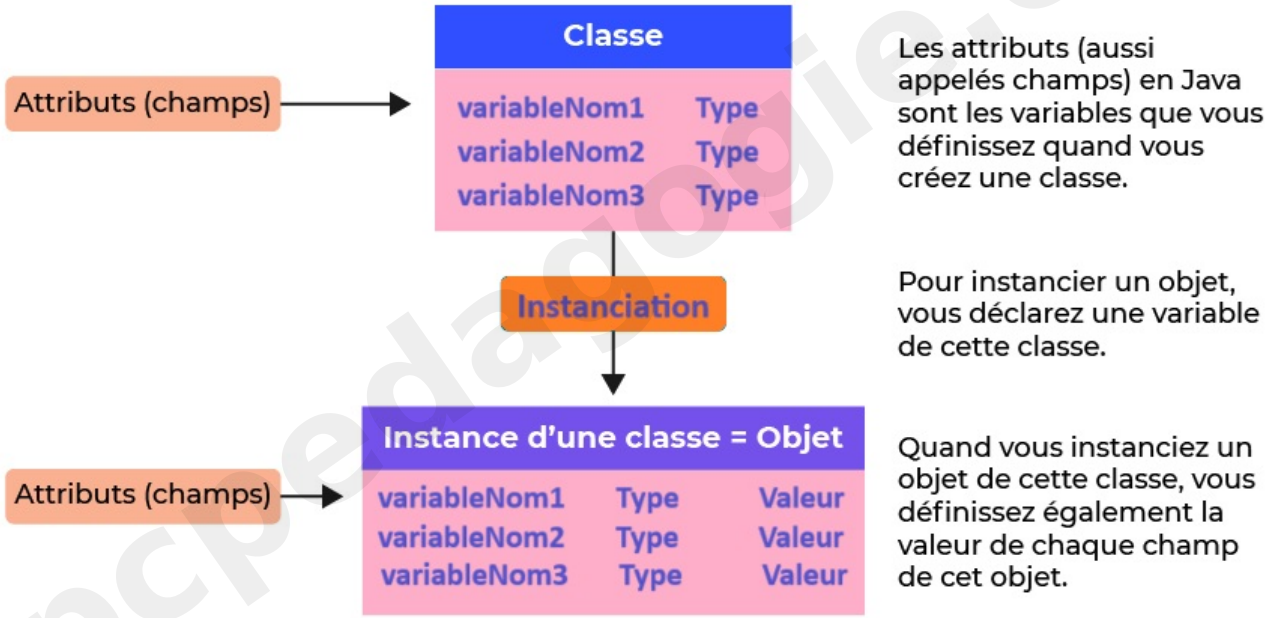
apcpedagogie.com

## 5. Déclaration d'attributs

- En Dart, vous déclarez les attributs à l'intérieur d'une classe. Vous pouvez utiliser différents types de données pour définir le type de chaque attribut.

```
class MaClasse {
  // Attributs
  int entier; // Exemple d'un attribut de type entier
  String chaine; // Exemple d'un attribut de type chaîne de caractères
  double virguleFlottante; // Exemple d'un attribut de type virgule flottante

  // Vous pouvez également initialiser les attributs lors de leur déclaration
  bool estVrai = true;
}
```




## 6. Comment définir et utiliser des attributs dans une classe Dart

```
class MaClasse {
  // Attributs de la classe
  String nom;
  int age;

  // Constructeur
  MaClasse(this.nom, this.age);
```

```
  // Méthode pour afficher les attributs
  void afficherInfos() {
    print('Nom: $nom, Age: $age');
  }
}

void main() {
  // Création d'une instance de MaClasse
  var objet = MaClasse('Alice', 30);

  // Utilisation des attributs et des méthodes de la classe
  print(objet.nom); // Affiche: Alice
  print(objet.age); // Affiche: 30
  objet.afficherInfos(); // Affiche: Nom: Alice, Age: 30
}
```

- **Dans cet exemple :**
  - **nom** et **age** sont les attributs de la classe **MaClasse**.
  - Ils sont définis à l'intérieur de la classe mais en dehors de toute méthode.
  - Le constructeur MaClasse permet d'initialiser ces attributs lors de la création d'une instance de la classe.
  - La méthode afficherInfos est une méthode de la classe qui permet d'afficher les valeurs des attributs.
  - Les attributs d'une classe peuvent avoir différents types de données, y compris les types primitifs comme int, double, String, ainsi que d'autres classes, et même des listes ou des maps.

1. **Attributs publics :**

   - Les **attributs publics** sont accessibles depuis n'importe où dans le code.

```
class MaClasse {
  String nom; // Attribut public

  MaClasse(this.nom);
}

void main() {
  var objet = MaClasse('Alice');
  print(objet.nom); // Accessible depuis l'extérieur de la classe
```

apcpedagogie.com

```
}
```

- Dans cet exemple, **nom** est un **attribut public** de la classe **MaClasse**. Il peut être accédé depuis l'extérieur de la classe, comme dans la **fonction main()** où nous imprimons sa valeur.
- Les attributs publics sont généralement utilisés lorsque vous souhaitez que les valeurs de ces attributs soient accessibles depuis d'autres parties de votre programme. Cependant, il est important de noter que cela peut également rendre votre code plus fragile, car d'autres parties du code peuvent accéder et modifier directement ces attributs sans contrôle supplémentaire.
- Si vous souhaitez restreindre l'accès à un attribut ou ajouter une logique supplémentaire lors de la lecture ou de l'écriture de cet attribut, vous pouvez utiliser des méthodes d'accès (getters et setters) pour encapsuler cet attribut.

2. **Attributs privés :**

- En Dart, vous pouvez déclarer des attributs privés en préfixant leur nom avec un tiret bas _. Cela signifie que ces attributs ne seront accessibles que depuis l'intérieur de la classe où ils sont définis. Voici un exemple :

```
class MaClasse {
  String _attributPrive; // Attribut privé

  MaClasse(this._attributPrive);

  void afficherAttributPrive() {
    print(_attributPrive); // Accessible à l'intérieur de la classe
  }
}

void main() {
  var objet = MaClasse('Valeur privée');
  objet.afficherAttributPrive(); // Mais il peut être utilisé à l'intérieur
des méthodes de la classe
}
```

- Dans cet exemple, **_attributPrive** est un attribut privé de la classe MaClasse. Il ne peut être accédé que depuis l'intérieur de cette classe. Si vous essayez d'accéder à cet attribut depuis l'extérieur de la classe (comme dans la fonction main()), cela générera une erreur.

apcpedagogie.com

- Les attributs privés sont souvent utilisés pour **encapsuler** les détails d'implémentation à l'intérieur d'une classe, ce qui les rend inaccessibles depuis l'extérieur. Cela favorise l'encapsulation et la modularité du code, car les détails internes de la classe ne sont pas exposés à d'autres parties du programme.

3. **Attributs en lecture seule dans une classe dart :**

- En Dart, vous pouvez définir des attributs en lecture seule en utilisant un constructeur sans un setter correspondant. Cela signifie que l'attribut ne peut être initialisé qu'une seule fois lors de la création de l'objet, et une fois initialisé, sa valeur ne peut pas être modifiée. Voici un exemple :

```
class MaClasse {
  final String attributEnLectureSeule; // Attribut en lecture seule

  MaClasse(this.attributEnLectureSeule);
}

void main() {
  var objet = MaClasse('Valeur initiale');
  //objet.attributEnLectureSeule = 'Nouvelle valeur'; // Cela génèrerait une
erreur car l'attribut est en lecture seule
  print(objet.attributEnLectureSeule); // Affiche: Valeur initiale
}
```

- Dans cet exemple, **attributEnLectureSeule** est un attribut en lecture seule de la classe MaClasse. Il est déclaré avec le mot-clé final, ce qui signifie qu'une fois initialisé dans le constructeur, sa valeur ne peut pas être modifiée.
- L'utilisation d'attributs en lecture seule peut être utile lorsque vous avez des valeurs qui doivent être définies une fois et ne doivent pas être modifiées par la suite, ce qui peut renforcer l'immutabilité de votre code et éviter les erreurs potentielles de modification accidentelle des valeurs.

4. **Attributs avec initialisation par défaut :**

- En Dart, vous pouvez initialiser les **attributs avec une valeur par défaut** directement dans la déclaration de la classe ou dans le constructeur. Voici quelques exemples pour chaque méthode :

1. **Initialisation dans la déclaration de la classe :**

apcpedagogie.com

```
class MaClasse {
  String attributAvecValeurParDefaut = 'Valeur par défaut'; // Attribut
avec valeur par défaut

  MaClasse();
}

void main() {
  var objet = MaClasse();
  print(objet.attributAvecValeurParDefaut); // Affiche: Valeur par
défaut
}
```

- Dans cet exemple, **attributAvecValeurParDefaut** est un attribut de la classe MaClasse qui est initialisé avec une valeur par défaut directement lors de sa déclaration.

2. **Initialisation dans le constructeur :**

```
class MaClasse {
  String attributAvecValeurParDefaut; // Attribut

  MaClasse([String valeurParDefaut = 'Valeur par défaut']) :
attributAvecValeurParDefaut = valeurParDefaut;
}

void main() {
  var objet = MaClasse();
  print(objet.attributAvecValeurParDefaut); // Affiche: Valeur par
défaut
}
```

- Dans cet exemple, **attributAvecValeurParDefaut** est un attribut de la **classe MaClasse** qui est initialisé avec une valeur par défaut dans le constructeur. La valeur par défaut est spécifiée comme un paramètre optionnel avec une valeur par défaut dans les **crochets []** du constructeur.

*Que vous initialisiez vos attributs avec une valeur par défaut dans la déclaration de la classe ou dans le constructeur dépend de vos besoins spécifiques et de la façon dont vous préférez organiser votre code.*

apcpedagogie.com

5. **Attributs avec initialisation par défaut :**

   - En Dart, vous pouvez déclarer des attributs statiques dans une classe en utilisant le mot-clé static. Les attributs statiques sont partagés par toutes les instances de la classe et peuvent être accédés sans avoir besoin d'instancier la classe. Voici un exemple :

```
class MaClasse {
  static int attributStatique = 0; // Attribut statique

  void incrementerAttributStatique() {
    attributStatique++;
  }
}

void main() {
  print(MaClasse.attributStatique); // Accès à l'attribut statique sans
instancier la classe

  var objet1 = MaClasse();
  var objet2 = MaClasse();

  objet1.incrementerAttributStatique();

  print(objet1.attributStatique); // Affiche: 1
  print(objet2.attributStatique); // Affiche: 1 (partagé entre toutes les
instances)
}
```

   - Dans cet exemple, **attributStatique** est un attribut statique de la classe MaClasse. Il est déclaré avec le mot-clé static. Vous pouvez y accéder en utilisant le nom de la classe suivi de . puis du nom de l'attribut.
   - L'avantage des attributs statiques est qu'ils sont partagés par toutes les instances de la classe, ce qui peut être utile pour stocker des informations partagées entre toutes les instances, telles que des compteurs, des configurations globales, etc.

6. **Attributs initialisés de manière différée :**

   - En Dart, le mot-clé **late** est utilisé pour déclarer des attributs qui sont initialisés de manière différée, c'est-à-dire des attributs qui ne sont pas immédiatement initialisés lors de leur déclaration, mais plus tard, avant leur utilisation. Voici un

apcpedagogie.com

exemple :

```
class MaClasse {
  late String attributLate; // Attribut late

  MaClasse() {
    attributLate = 'Valeur initiale'; // Initialisation tardive
  }

  void afficherAttributLate() {
    print(attributLate);
  }
}

void main() {
  var objet = MaClasse();
  objet.afficherAttributLate(); // Affiche: Valeur initiale
}
```

- Dans cet exemple, **attributLate** est un attribut de la classe MaClasse qui est déclaré avec le mot-clé late. Il n'est pas initialisé lors de sa déclaration mais avant sa première utilisation, ici dans le constructeur. L'initialisation tardive est souvent utilisée lorsque la valeur d'un attribut ne peut pas être déterminée lors de sa déclaration, mais plutôt à un moment ultérieur, comme lors de l'initialisation de l'objet ou lors de l'appel d'une méthode spécifique.
- Il est important de noter que si vous accédez à un attribut late avant qu'il ne soit initialisé, cela entraînera une exception LateInitializationError. Par conséquent, vous devez vous assurer que l'attribut est initialisé avant de l'utiliser pour éviter cette erreur.

## 7. Application

1. **App01**

apcpedagogie.com

1. Créer une classe « **Personne** » avec un attribut public « **nom**« , un attribut privé « **age** » et un attribut en lecture seule « **profession**« .
2. Définir un attribut statique « **ville** » qui est initialisé à « **Paris**« .
3. Implémenter un constructeur qui prend les valeurs du **nom** et de la profession et les assigne aux attributs correspondants.
4. Définir un getter pour l'attribut privé « **age** » afin de le rendre accessible depuis l'extérieur de la classe.
5. Implémenter un setter pour l'attribut privé « **age** » qui vérifie que la valeur est positive.
6. Ajouter une méthode « **anniversaire** » qui incrémente l'âge de la personne de 1.
7. Ajouter une méthode statique « **changerVille** » qui permet de modifier la valeur de l'attribut statique « ville ».

## Solution

```
class Personne {
  String nom; // Attribut public
  int _age; // Attribut privé
  final String profession; // Attribut en lecture seule
  static String ville = "Paris"; // Attribut statique

  // Ajoutez une initialisation pour _age dans le constructeur
  Personne(this.nom, this.profession, this._age);

  // Autre option : utilisez late pour _age si vous prévoyez de l'initialiser
plus tard
  // late int _age;

  int get age => _age; // Getter pour l'attribut privé _age

  set age(int value) {
    if (value >= 0) {
      _age = value;
    } else {
      throw ArgumentError("L'âge ne peut pas être négatif.");
    }
  }

  void anniversaire() {
    _age++;
  }

  static void changerVille(String nouvelleVille) {
    ville = nouvelleVille;
  }
}
void main() {
  // Votre code ici
```

apcpedagogie.com

```
  // Exemple d'utilisation de la classe Personne :
  var personne = Personne("Alice", "Ingénieur", 30);
  print("Nom : ${personne.nom}, Age : ${personne.age}");
}
```

2. **App02**

Définissez une classe `Rectangle` en Dart avec les attributs suivants :

- `longueur` : de type double, représentant la longueur du rectangle.
- `largeur` : de type double, représentant la largeur du rectangle.

La classe doit avoir les fonctionnalités suivantes :

- Un constructeur pour initialiser les valeurs de `longueur` et `largeur`.
- Des getters pour `longueur` et `largeur`.
- Une méthode `calculerSurface` qui retourne la surface du rectangle (produit de la longueur par la largeur).

Solution

```
// Définition de la classe Rectangle
class Rectangle {
  double longueur; // Attribut représentant la longueur du rectangle
  double largeur; // Attribut représentant la largeur du rectangle

  // Constructeur pour initialiser la longueur et la largeur
  Rectangle(this.longueur, this.largeur);

  // Getter pour récupérer la longueur du rectangle
  double getLongueur() {
    return longueur;
  }

  // Getter pour récupérer la largeur du rectangle
  double getLargeur() {
    return largeur;
  }

  // Méthode pour calculer la surface du rectangle
  double calculerSurface() {
    return longueur * largeur;
  }
}

// Fonction principale
```

apcpedagogie.com

```
void main() {
  // Création d'un rectangle avec une longueur de 5.0 et une largeur de 3.0
  Rectangle monRectangle = Rectangle(5.0, 3.0);

  // Affichage de la longueur et de la largeur du rectangle
  print("Longueur du rectangle : ${monRectangle.getLongueur()}");
  print("Largeur du rectangle : ${monRectangle.getLargeur()}");

  // Calcul de la surface du rectangle
  double surface = monRectangle.calculerSurface();
  print("Surface du rectangle : $surface");
}
```

---

## Cours Dart

Source : https://apcpedagogie.com/les-attributs-dune-classe-dans-dart · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com