Exercices d'initiation au langage Dart

TP

Consignes

Consultez la documentation du langage Dart pour résoudre les exercices (https://dart.dev/guides).

01 - String

1. Affichez ***"Hello, World !"*** dans la console de débogage,

2. Créez ***une variable de type String*** nommée ***message*** avec la valeur ***"Hello, World !"***.
   Affichez la valeur de la variable ***message*** dans la console.

3. Créez 2 constantes de type String : ***hello*** et ***world***, ayant respectivement pour valeur ***"Hello"*** et ***"World"***.
   Sans recourir à une concaténation, affichez ***"Hello, World !"*** dans la console, à l'aide des 2 constantes et d'une **interpolation sur une chaîne de caractères**.

4. Affichez ***"HELLO, WORLD !"*** dans la console à l'aide des constantes ***hello*** et ***world*** précédemment déclarées, via une **concaténation**.

5. A partir de la variable ***message*** (définie précédemment), sans en modifier le contenu manuellement, affichez ***"Hello"*** dans la console.

6. Créez une constante de type String, nommée ***message*** contenant la phrase ***"Hello, World !"***.
   Affichez le nombre de caractères contenus dans ***message***.

7. A partir d'une constante ***welcome*** de type String, ayant pour valeur ***"Hello, World !"***, par programmation affichez ***"H3118, W8R1D !"*** dans la console.

8. Créez une constante ***welcome*** de type String avec la valeur ***"Hello World"***.
   A partir de la constante ***welcome***, générez une liste nommée **words**, ne pouvant contenir que des valeurs de type String.

   La liste ***words*** devra contenir les 2 items suivants ***"Hello"*** et ***"World"***.
   La liste ***words*** doit être immuable.

9. Créez une variable de type String nommée ***pwd***.
   A l'aide d'une condition ternaire, affichez le message *"Mot de passe manquant"* si la chaîne de caractères est vide et *"Mot de passe fourni"* dans le cas contraire.

10. Créez une variable ***email*** avec la valeur *"john@doe.com"*. Pour vous assurer que la valeur est bien une adresse mail, vérifiez que la variable ***email*** contient un "@" et un ".".
    A l'aide d'une condition ternaire, indiquez si la valeur de l'email est valide ou non (en vous basant sur la présence des 2 caractères évoqués ci-avant).

02-Number

1. Créez 2 variables typées (avec la valeur de votre choix) :
    a. un entier nommé ***price1***,
    b. un nombre à virgule nommé ***price2***.

    Affichez la valeur de ***price1*** et de ***price2*** dans la console.

2. Créez une variable ***sum*** qui aura pour valeur la somme de ***price1*** et ***price2***.

3. Affichez le type de la variable ***sum*** dans la console.

4. Convertissez la variable ***sum*** en entier et affichez sa valeur dans la console.

5. Créez une valeur immuable ***strSeven,*** de type chaîne de caractères, contenant la valeur ***"7"***.
    Puis, créez une valeur immuable ***numSeven*** de type entier, correspondant à la variable ***strSeven*** convertie en entier.
    Affichez la valeur de ***numSeven*** dans la console.

03 - Collections (list, set et map) + boucles

1. Créez une liste typée nommée ***planets***, pouvant uniquement contenir des chaînes de caractères. Insérez dans ce tableau les éléments suivants :
    - *Terre,*
    - *Mars,*
    - *Mercure,*
    - *Saturne,*
    - *Vénus,*
    - *Neptune,*
    - *Uranus,*
    - *Jupiter.*

    Affichez dans la console la valeur de la liste ***planets*** triée dans l'ordre alphabétique descendant.

2. A l'aide d'une boucle ***for in***, affichez dans la console, un à un chaque item de la liste ***planets*** en majuscules.

3. A l'aide d'une boucle ***while***, affichez la première lettre de chaque item de la liste.

4. A l'aide d'une boucle ***do while***, affichez chaque item de la liste précédé de son index + 1.

    Ex :
    1 - Terre
    2 - Mars
    …

5. A l'aide d'une boucle ***while***, affichez dans la console, uniquement les items de la liste dont la dernière lettre est une voyelle.

6. Par programmation, ajoutez à la liste ***planets*** un nouvel item dont la valeur est *"Pluton"*.
    Affichez la valeur de la liste ***planets*** dans la console.

7. Créez une classe ***Planet*** disposant des attributs :
    - ***name*** de type String,
    - ***distanceFromEarth*** de type double,

    Créez 7 instances de Planet en vous basant sur le tableau suivant.

Planète	**Distance de la Terre en Mkm**
Mercure	91,69
Saturne	1275
Neptune	4351,40
Jupiter	628,73
Mars	78,34
Venus	41,40
Uranus	2723,95

    Créez une liste de type ***Planet***, contenant les 7 instances de ***Planet*** créés précédemment.

    Affichez la liste ***planets*** triée par distance de la Terre (par ordre ascendant).

8. Créez une collection de type ***map***, nommée ***apollo*** contenant, les clés / valeurs suivantes :
    - 07_1969 : Apollo 11
    - 11_1969 : Apollo 12
    - 02_1971 : Apollo 14
    - 07_1971 : Apollo 15
    - 04_1972 : Apollo 16
    - 12_1972 : Apollo 17

    Affichez dans la console la valeur de ***apollo***.

    Affichez dans la console la valeur de la clé ***07_1971***.

    Affichez dans la console les clés de la map ***apollo***.

Affichez dans la console les valeurs de la map ***apollo***.

9. Mettez à jour la valeur de la clé ***07_1969*** de la map ***apollo*** : *"Neil Armstrong + Buzz Aldrin"*.

   Affichez la valeur de la clé ***07_1969*** de la map ***apollo*** dans la console.

10. Créez un enum ***Kind*** contenant 5 valeurs de type ***String*** : ***planet, star, satellite, asteroid et comet.***

    Créez une classe ***SolarSystemElement*** disposant d'un attribut ***name*** de type ***String*** et d'un attribut ***kind*** *de type* ***Kind***.

    Instanciez 4 occurrences de type ***SolarSystemElement*** :
    - ***sun*** (star),
    - ***earth*** (planet),
    - ***moon*** (satellite),
    - ***pluton*** (satellite).

    Affichez ces objets dans la console.

04 - Functions, lambdas & Null Safety

1. Créez une fonction ***sayHello***, disposant d'un paramètre de type String et permettant d'obtenir la chaîne de caractères *"Hello, <param> !"*,

2. Créez une fonction ***sum*** permettant d'obtenir la somme de 2 nombres passés en paramètres.

3. Créez une fonction ***sumMany*** permettant d'obtenir la somme d'une quantité indéfinie de nombres passés en paramètres.

4. Créez une fonction ***sumAndPrint***, ne retournant aucune valeur, et disposant de 2 paramètres obligatoires de type ***num*** (***param1, param2***) et d'un 3ème paramètre optionnel de type booléen ***shouldPrint***.

   La fonction ***sumAndPrint*** additionne ***param1*** et ***param2***, et stocke le résultat dans une variable ***result*** de type ***num***.

   Si la valeur de ***shouldPrint*** est ***true***, la fonction ***sumAndPrint*** affiche l'opération et son résultat dans la console (ex : "1+1=2").

5. Créez une fonction ***sumAndFormat***, retournant le résultat de l'addition de 2 nombres et disposant d'un paramètre nommé optionnel **ft** permettant de retourner la valeur sous forme de nombre entier (***int***) ou de nombre à virgule (***double***).

   Si le paramètre ***ft*** n'est pas renseigné, ***sumAndFormat*** n'applique pas de formatage au résultat.

6. Créez une lamba (*fonction anonyme*) nommée ***strReverse*** permettant d'inverser l'ordre des lettres d'une chaîne de caractères.

   Créez une variable ***hello*** de type String, avec la valeur *"Hello"*.

   Utilisez la lambda ***strReverse*** pour inverser l'ordre des lettres de ***hello*** et stockez le résultat dans une variable nommée ***reversed***.

   Affichez la valeur de **reversed** dans la console.

7. Programmez une fonction permettant d'afficher les 17 premiers nombres de la ***"Suite de Fibonacci"*** où chaque terme est la somme des deux termes qui le précèdent (https://fr.wikipedia.org/wiki/Suite_de_Fibonacci).

   0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987

05 - POO

1. Créez un dossier ***classes***, à l'intérieur de ce dossier créez un fichier ***User.dart***.
   Ecrivez une classe ***User***, disposant des 3 attributs publics suivants :
   - **firstname (String),**
   - **lastname (String),**
   - **email (String),**

2. Créez un constructeur de classe permettant de renseigner ces 3 attributs,

3. A la racine, créez un fichier ***main.dart*** et instanciez une constante nommée **johndoe**, de type ***User*** avec les valeurs suivantes :
   - ***firstname : "John",***
   - ***lastname : "Doe",***
   - ***email : "john@doe.com".***

4. Créez une méthode **toString()**, permettant de retourner l'objet User sous forme de chaîne de caractères. Attention, puisqu'il s'agit d'une surcharge, cette méthode doit être annotée avec la mention ***@override***.

5. Ajoutez un ***champ privé _password*** (String), doté d'un ***setter (mutateur)***.
   Attention, la méthode ***toString()*** précédemment écrite, ne doit pas révéler la valeur de l'attribut _password. Remplacez la valeur par ********.*** Affichez dans la console la valeur de la variable ***johndoe***.

6. Créez une méthode sayHello, permettant d'afficher dynamiquement la chaîne de caractères suivante : ***"Hello, I'm <firstname> <lastname> !"***. Testez la méthode dans le fichier main.dart.

7. Si vous n'avez pas utilisé un projet Dart (via la commande dart create), ajoutez un fichier **pubspec.yaml**, renseignez les informations suivantes :
   - ***name: oop***
   - ***environment:***
     sdk: '>= 2.10.0 <3.0.0'

   Importez le package externe ***crypto***

(https://pub.dev/documentation/crypto/latest/crypto/crypto-library.html)

dart pub add crypto
dart pub get

et le package interne **dart:convert** dans **User.dart**

Procédez aux implémentations et adaptations nécessaires pour que l'utilisation du setter de ***_password*** génère un mot de passe chiffré, stocké dans un attribut privé ***_hash***.

8. Ajoutez une méthode ***authenticate*** à la classe User permettant de vérifier la valeur d'un mot de passe. Testez les 2 cas de figure possibles dans la fonction main.

9. Créez une classe ***Admin*** héritant de ***User***. Ajoutez un attribut privé **grade**, de type int, par défaut égal à 1.
   Ajoutez un ***getter (assesseur)*** et un ***setter (mutateur)*** pour l'attribut ***grade***.

10. Créez les classes suivantes :
    - ***Lesson*** avec l'attribut privé ***_name***,
    - ***Student*** avec les attributs privés ***_firstname***, ***_lastname***.

    Une occurrence de ***Student*** possède de 0 à N à ***Lesson***. Ajoutez un attribut privé ***_lessons*** à ***Student*** permettant de gérer cette association.

    Programmez :
    - le constructeur et les ***getters*** / ***setters*** pour chaque attribut,
    - les surcharges de méthodes ***toString()*** permettant d'afficher les occurrences de classe ***User*** et ***Student*** en détail dans la console.

    - Créez un objet Student nommé johnDoe (firstname = John, lastname = Doe),
    - Créez un objet de type ***Lesson*** nommé ***italian*** (name = "Italian"),
    - Associez ***johnDoe*** à ***italian***,
    - Affichez l'objet ***johnDoe*** dans la console,
    - Créez un nouvel objet de type ***Lesson*** nommé ***french*** (name = "French") et ajoutez-le aux associations de l'objet ***johnDoe*** (sans supprimer l'association précédente avec **italian**).

- Affichez l'objet ***johnDoe*** dans la console.
- Supprimez l'objet ***italian*** des ***lessons*** de ***johnDoe***.
- Affichez l'objet ***johnDoe*** dans la console.

06 - Tests Unitaires

Reprenez la fonction ***sum*** implémentée dans l'exercice sur les types **num**.

1. Utilisez l'expression ***assert*** pour vérifier en mode ***debug*** que les valeurs retournées par la fonction ***sum*** sont correctes.

2. Créez une fonction nommée ***sumFails*** retournant une valeur de type ***Never***.
   Cette fonction permettra de générer une exception de type ArgumentError personnalisée.
   Utilisez la fonction ***sumFails*** pour générer une erreur en mode Debug lorsque la valeur retournée par la fonction ***sum*** retourne une valeur différente de celle attendue.

3. Créez une variable **dynamic** nommée ***something*** sans lui attribuer de valeur.
   Donnez la valeur 7 à la variable. Affichez son type dans la console.
   Donnez la valeur "seven" à la variable. Affichez son type dans la console.
   A l'aide d'un test avec ***assert***, vérifiez que le type de something est conforme à la valeur attendue après chaque évolution de son contenu.

Annexes

Dart

Documentation officielle du langage Dart : https://dart.dev/

Exécuter un programme Dart

- sur votre machine (après avoir installé Dart),
    - Créer un projet Dart (solution à privilégier) :
      ex :
      `dart create <nom-du-projet>`

      OU
    - Exécuter vos fichiers dans votre terminal avec le CLI de Dart :
      ex :
      `dart run <nom-du-fichier>.dart`

  OU

- en ligne, sans installation préalable sur votre machine, utilisez l'application web ***Dart Pad*** (https://dartpad.dev/) ou le simulateur sur le site officiel de Dart https://dart.dev/#try-dart

Un programme Dart doit toujours contenir une fonction ***"main"*** pour pouvoir être exécuté.

Dans un programme Dart (hors Flutter), la méthode ***"print"*** ne peut être appelée qu'au sein de la fonction ***"main"***.

```
//code ici

void main(){
     //code à tester ici
}
```

Dans Visual Studio Code, cliquez sur le bouton ***Run*** ou ***Debug***.

Seul le mode ***Debug*** prend en compte les tests de type ***assert***.

En alternative, le résultat d'un assert peut être affiché si le fichier Dart est exécuté ainsi :

dart run --enable-asserts

Création d'un projet Dart

La création d'un projet Dart permet de disposer d'une configuration prête à l'emploi, facilitant l'import de dépendances (packages Dart) et l'exécution de tests (assert).

Création d'un projet Dart (le dossier racine portera le nom du projet) :

```
dart create <nom-du-projet>
```

Déplacement dans le dossier nouvellement créé, reprenant le nom du projet :

```
cd <nom-du-projet>
```

Exécution du fichier main.dart :

```
dart run
```

Exécution des tests :

```
dart test
```

Suggestion d'organisation du projet

Créez un dossier (ou un fichier) par série d'exercices et une méthode par exercice.

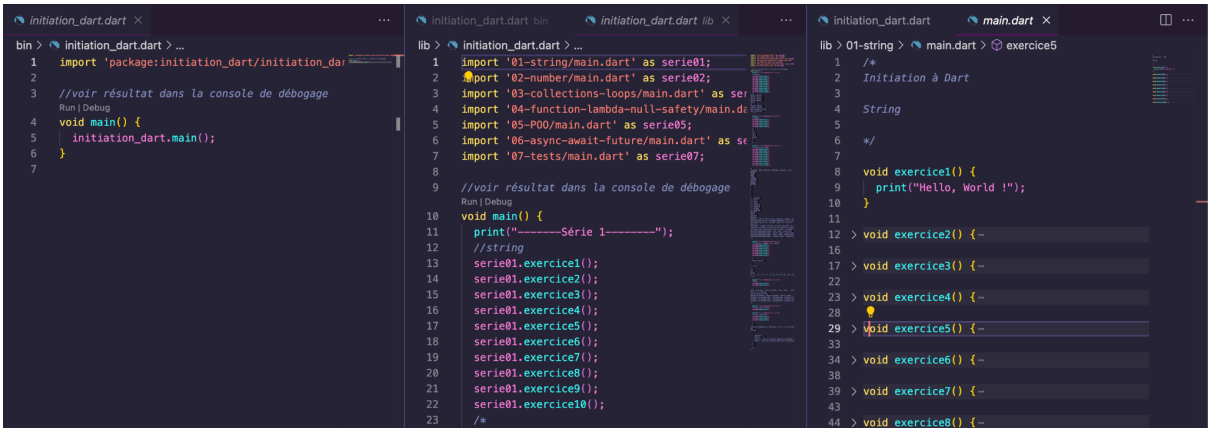
Fichier exécuté par la commande : ***dart run***
./**bin**/initiation_dart.dart

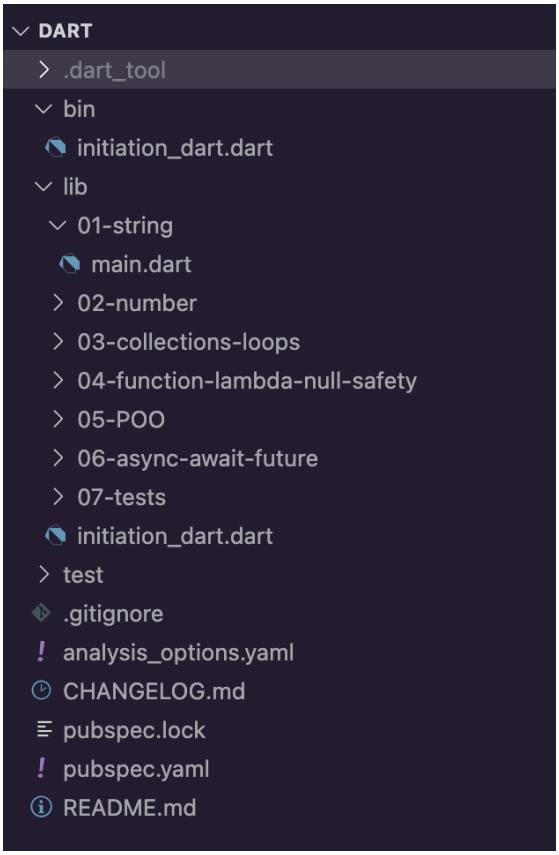
Point d'entrée du projet appelé par le fichier ./bin/initiation_dart.dart
./**lib**/initiation_dart.dart

./lib/**01-string**/main.dart
./lib/**02-numbe**r/main.dart
…
./lib/**07-tests**/main.dart

Ex : méthode ***exercice1*** du fichier ***./lib/01-string/main.dart***

```dart
void exercice1() {
 print("Hello, World !");
}
```




DART
.dart_tool
bin
initiation_dart.dart
lib
01-string
main.dart
02-number
03-collections-loops
04-function-lambda-null-safety
05-POO
06-async-await-future
07-tests
initiation_dart.dart
test
.gitignore
analysis_options.yaml
CHANGELOG.md
pubspec.lock
pubspec.yaml
README.md