Les opérateurs dans le langage Dart

1. Présentation

- Un opérateur est un symbole utilisé pour manipuler les valeurs ou effectuer des opérations sur son opérande.
- Dart fournit un ensemble complet d'opérateurs intégrés pour accomplir divers types d'opérations. Les opérateurs peuvent être unaires ou binaires, ce qui signifie qu'unaire ne prend qu'un opérande et que le binaire prend deux opérandes avec des opérateurs.

2. Types d'opérateurs en langage Dart

- Il existe de nombreux opérateurs que nous pouvons utiliser, dans tous les langages de programmation l'utilisation d'opérateurs a le même objectif, seule la syntaxe est différente.

1. Opérateurs arithmétiques

- Les **opérateurs arithmétiques** sont les opérateurs les plus courants utilisés pour effectuer des additions, des soustractions, des multiplications, des divisions, etc. Prenons la variable a contient 20 et la variable b contient 10, puis -
- Dans le tableau suivant, la valeur **X** est définie sur **35**.

Type d'opération	Description	Symbole	Exemple	Résultat
Calculer	Addition- Il ajoute l'opérande gauche à l'opérande droit.	Ajout(**+**)	var X + 2;	37
Soustraction	Soustraction-Il soustrait l'opérande droit de l'opérande gauche.	Soustraction(**-**)	var X - 2;	33
Multiplication	Multiplication- Il multiplie un opérande par un autre opérande.	Multiplication(*****)	var X * 3;	105

apcpedagogie.com

Type d'opération	Description	Symbole	Exemple	Résultat
Division	Division- Il divise le premier opérande par le deuxième opérande et renvoie le quotient.	Division(/)	var X / 7;	5
Division de nombres entiers	Division de nombres entiers- Il divise le premier opérande par le deuxième opérande et renvoie un quotient entier.	~/	var X ~/ 7;	5

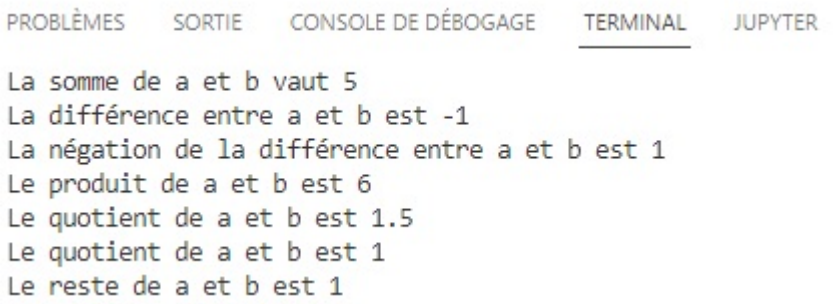
Exemple : Utilisation d'opérateurs arithmétiques dans un programme

Programme Dart:

```
Run | Debug
void main() {
  int a = 2;
  int b = 3;
  // Ajout de a et b
  var c = a + b;
  print("La somme de a et b vaut $c");
  // Soustraire a et b
  var d = a - b;
  print("La différence entre a et b est $d");
  // Utilisation du moins unaire
  var e = -d;
  print("La négation de la différence entre a et b est $e");
  // Multiplication de a et b
  var f = a * b;
  print("Le produit de a et b est $f");
  // Division de a et b
  var g = b / a;
  print("Le quotient de a et b est $g");
  // Utilisation de ~/ pour diviser a et b
  var h = b ~/ a;
  print("Le quotient de a et b est $h");
  // Reste de a et b
  var i = b % a;
  print("Le reste de a et b est $i");
}
```

Production:

apcpedagogie.com



2. **Opérateurs d'affectation**

   - L'opérateur d'affectation est un opérateur permettant d'affecter une valeur à une variable. Cet opérateur utilise le signe égal (=). en fait, cela se fait de droite à gauche et non de gauche à droite comme en mathématiques.

   1. **Addition avec affectation (+=)**

      - Cet opérateur ajoute la valeur d'une expression à une variable et affecte le résultat à la variable.

      ```
      int a = 5;
      a += 3; // Équivaut à a = a + 3;
      print(a); // Affiche 8
      ```

   2. **Soustraction avec affectation (-=)**

      - Cet opérateur soustrait la valeur d'une expression de la variable et affecte le résultat à la variable.

      ```
      int b = 10;
      b -= 4; // Équivaut à b = b - 4;
      print(b); // Affiche 6
      ```

   3. **Multiplication avec affectation (*=)**

      - Cet opérateur multiplie la variable par la valeur d'une expression et affecte le résultat à la variable.

apcpedagogie.com

```
int c = 7;
c *= 2; // Équivaut à c = c * 2;
print(c); // Affiche 14
```

4. **Division avec affectation (`/=`)**

   - Cet opérateur divise la variable par la valeur d'une expression et affecte le résultat à la variable.

```
double d = 20.0;
d /= 5; // Équivaut à d = d / 5;
print(d); // Affiche 4.0
```

5. **Modulo avec affectation (`%=`)**

   - Cet opérateur calcule le reste de la division de la variable par la valeur d'une expression et affecte le résultat à la variable.

```
int e = 17;
e %= 4; // Équivaut à e = e % 4;
print(e); // Affiche 1
```

6. **ET bit-à-bit avec affectation (`&=`)**

   - Cet opérateur effectue un ET bit-à-bit entre la variable et la valeur d'une expression, puis affecte le résultat à la variable.

```
int flags = 0b1101; // 13 en binaire
flags &= 0b1010;   // 10 en binaire
print(flags); // Affiche 0b1000, soit 8 en décimal
```

7. **OU bit-à-bit avec affectation (`|=`)**

   - Cet opérateur effectue un OU bit-à-bit entre la variable et la valeur d'une expression, puis affecte le résultat à la variable.

apcpedagogie.com

```
int flags = 0b1101; // 13 en binaire
flags |= 0b1010;   // 10 en binaire
print(flags); // Affiche 0b1111, soit 15 en décimal
```

8. **OU exclusif bit-à-bit avec affectation (^=)**

   - Cet opérateur effectue un OU exclusif bit-à-bit entre la variable et la valeur d'une expression, puis affecte le résultat à la variable.

```
int flags = 0b1101; // 13 en binaire
flags ^= 0b1010;   // 10 en binaire
print(flags); // Affiche 0b0111, soit 7 en décimal
```

9. **Décalage à gauche avec affectation (<<=)**

   - Cet opérateur effectue un décalage à gauche de la valeur de la variable par un certain nombre de positions, puis affecte le résultat à la variable.

```
int number = 5; // En binaire, 5 est 0b0101
number <<= 2;   // Effectue un décalage à gauche de 2 positions
print(number);  // Affiche 20 en décimal, 0b10100 en binaire
```

10. **Décalage à droite avec affectation (>>=)**

    - Cet opérateur effectue un décalage à droite de la valeur de la variable par un certain nombre de positions, puis affecte le résultat à la variable.

```
int number = 32; // En binaire, 32 est 0b100000
number >>= 3;    // Effectue un décalage à droite de 3 positions
print(number);   // Affiche 4 en décimal, 0b100 en binaire
```

    - La variable number est initialisée à 32, qui est représentée en binaire comme 0b100000. L'opérateur >>= effectue un décalage à droite de 3 positions, ce qui signifie que le nombre est divisé par 2^3, soit 8. Donc, 32 divisé par 8 équivaut à 4, ce qui est le résultat affiché. Le

apcpedagogie.com

résultat binaire 0b000100 est également correct, représentant 4 en binaire.

3. **Opérateurs relationnels**

Symbole de l'opérateur	Nom de l'opérateur	Description de l'opérateur
>	Plus grand que	Vérifiez quel opérande est le plus grand et donnez le résultat sous forme d'expression booléenne.
<	Moins que	Vérifiez quel opérande est plus petit et donnez le résultat sous forme d'expression booléenne.
>=	Plus grand ou égal à	Vérifiez quel opérande est supérieur ou égal à l'autre et donnez le résultat sous forme d'expression booléenne.
<=	moins qu'égal à	Vérifiez quel opérande est inférieur ou égal à l'autre et donnez le résultat sous forme d'expression booléenne.
==	Égal à	Vérifiez si les opérandes sont égaux ou non et donnez le résultat sous forme d'expression booléenne.
!=	Pas égal à	Vérifiez si les opérandes ne sont pas égaux ou non et donnez le résultat sous forme d'expression booléenne.

Exemple : Utilisation d'opérateurs relationnels dans un programme

Programme Dart:

apcpedagogie.com

```
Run | Debug
void main() {
  int a = 2;
  int b = 3;
  // Plus grand entre a et b
  var c = a > b;
  print("a est plus grand que b est $c");
  // Plus petit entre a et b
  var d = a < b;
  print("a est plus petit que b est $d");
  // Supérieur ou égal à entre a et b
  var e = a >= b;
  print("a est plus grand que b est $e");
  // Inférieur ou égal à entre a et b
  var f = a <= b;
  print("a est plus petit que b est $f");
  // Égalité entre a et b
  var g = b == a;
  print("a et b sont égaux est $g");
  // Inégalité entre a et b
  var h = b != a;
  print("a et b ne sont pas égaux est $h");
}
```

Production:

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  JUPYTER

a est plus grand que b est false
a est plus petit que b est true
a est plus grand que b est false
a est plus petit que b est true
a et b sont égaux est false
a et b ne sont pas égaux est true
```

4. **Opérateurs d'essai de type**

- En Dart, les opérateurs **`as`**, **`is`** et **`is!`** sont utilisés pour **tester le type d'une variable**. Si la condition est exacte , elle retourne **true** sinon **false** est retourné.

    1. **`Opérateur is`**

        - Cet opérateur est utilisé pour vérifier si une variable est d'un type particulier. Il retourne true si la variable est du type spécifié

et false sinon. Voici un exemple :

```
var variable = "Hello";
if (variable is String) {
  print("La variable est une chaîne de caractères.");
} else {
  print("La variable n'est pas une chaîne de caractères.");
}
```

2. **Opérateur is!**

   - Cet opérateur est l'inverse de is. Il vérifie si une variable n'est pas d'un type spécifique et retourne true si c'est le cas, sinon false. Voici un exemple :

```
var variable = 42;
if (variable is! String) {
  print("La variable n'est pas une chaîne de caractères.");
} else {
  print("La variable est une chaîne de caractères.");
}
```

3. **Opérateur as**

   - L'opérateur as est utilisé pour effectuer une conversion de type (cast) en Dart. Si la conversion est possible, il retourne la variable avec le type demandé, sinon il renvoie null. Voici un exemple :

```
var variable = "42";
var nombre = variable as int?;
if (nombre != null) {
  print("La variable peut être convertie en int : $nombre");
} else {
  print("La conversion en int n'est pas possible.");
}
```

apcpedagogie.com

> *Ces opérateurs sont utiles pour gérer et tester les types de variables en Dart, ce qui peut être particulièrement important lors de la programmation orientée objet et de la gestion de différentes structures de données.*

4. **Application**

Enoncé:

- Développer un programme en Dart qui interagit avec l'utilisateur. Le programme doit demander à l'utilisateur d'indiquer son âge. Cependant, une attention particulière doit être portée à la saisie : le programme doit s'assurer que l'utilisateur fournit un nombre entier valide comme réponse. Si la saisie est invalide, le programme doit inviter l'utilisateur à ressaisir son âge jusqu'à obtenir une réponse valide.
- Une fois que vous avez une saisie valide, le programme devra déterminer si l'utilisateur est majeur (c'est-à-dire âgé de 18 ans ou plus) et afficher un message approprié en conséquence.

Solution

```
import 'dart:io';

void main() {
  int? age;

  while (age == null) {
    stdout.write("Quel est votre âge ? ");
    String saisie = stdin.readLineSync()!;

    age = int.tryParse(saisie);

    if (age == null) {
      print("Veuillez saisir un nombre entier
valide.");
    }
  }
```

apcpedagogie.com

```
  if (age >= 18) {
    print("Vous êtes majeur.");
  } else {
    print("Vous êtes mineur.");
  }
}
```

- `int?` age est une déclaration de variable en Dart qui indique que la variable age est de type int, mais elle peut également avoir la valeur null. Cette syntaxe est possible grâce à l'utilisation de la notation de non-nullabilité introduite dans Dart 2.12.
- En Dart, le `?` après le type indique que la variable peut être null. Cela signifie que age peut contenir soit une valeur entière, soit la valeur null.
- Par **exemple**, si vous déclarez `int? age;`, vous pouvez affecter à age une valeur entière, comme `age = 25;`, ou vous pouvez lui donner la valeur null, comme `age = null;`.
- L'utilisation de `int?` est utile lorsque vous travaillez avec des valeurs qui peuvent être potentiellement absentes (c'est-à-dire nulles), ce qui vous permet de gérer plus efficacement les cas où la valeur de age n'est pas définie.

5. **Opérateurs logiques**

- Les opérateurs logiques sont utilisés pour combiner deux ou plusieurs conditions.

Nom de l'opérateur	Symbole de l'opérateur	Description de l'opérateur
Opérateur Et	**&&**	Renvoie true uniquement si toutes les expressions spécifiées renvoient **true**. Si l'une des expressions est **false**, l'ensemble de l'expression && renverra **false**.
Opérateur Ou	**||**	Utilisé pour combiner deux ou plusieurs conditions, et renverra **true** dès qu'au moins l'une d'entre elles est vraie. Si aucune des conditions n'est vraie, l'ensemble de l'expression || renverra **false**.

apcpedagogie.com

Nom de l'opérateur	Symbole de l'opérateur	Description de l'opérateur
Opérateur Pas	!	Il est utilisé pour inverser le résultat d'une expression booléenne. Si une expression est true, l'opérateur ! la convertira en **false**, et inversement.

Exemple : Utilisation d'opérateurs logiques dans un programme

Programme Dart:

```
Run | Debug
void main() {
  int a = 5;
  int b = 7;
  // Utiliser l'opérateur Et
  bool c = a > 10 && b < 10;
  print(c);
  // Utiliser l'opérateur Ou
  bool d = a > 10 || b < 10;
  print(d);
  // Utiliser l'opérateur Not
  bool e = !(a > 10);
  print(e);
}
```

Production:

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  JUPYTER

false
true
true
```

6. **Opérateurs conditionnels**

- En Dart, il existe deux opérateurs conditionnels spéciaux qui simplifient les expressions conditionnelles, permettant d'effectuer des opérations en fonction d'une condition. Voici une description de ces opérateurs :

1. **Opérateur ternaire**

   - **`condition ? expression1 : expression2`**: Cet opérateur permet d'exprimer une condition de manière concise. Si la condition est vraie, il renvoie la valeur de expression1, sinon il renvoie la

valeur de expression2. C'est une forme abrégée d'une instruction **if-else**. Exemple :

```
var age = 25;
var message = age >= 18 ? "Majeur" : "Mineur";
print(message); // Affiche "Majeur"
```

2. **Opérateur de coalescence nulle**

   - `expression1 ?? expression2`: Cet opérateur permet de renvoyer la valeur de **expression1** si elle n'est pas nulle (non-null), sinon il renvoie la valeur de **expression2**. Cela est utile pour fournir une valeur de secours lorsque l'expression principale peut être nulle. Exemple :

```
String? nom; // Une variable pouvant être nulle
var nomAffiche = nom ?? "Anonyme";
print(nomAffiche); // Si nom est nulle, affiche "Anonyme",
sinon affiche la valeur de nom.
```

> *Ces opérateurs conditionnels simplifient le code en permettant des évaluations conditionnelles plus courtes et plus lisibles. Ils sont largement utilisés pour effectuer des opérations conditionnelles de manière concise en Dart.*

3. **Exemple : Utilisation d'opérateurs conditionnels dans un programme**

apcpedagogie.com

```dart
Run | Debug
main() {
  int? firstNum = 10;
  int? secondNum;
  var first = firstNum != 10 ? 'Not $firstNum' : '$firstNum';
  print(first); // 10
  /// la longue version avec if-else
  if (firstNum != 10) {
    print('Not $firstNum');
  } else {
    print('$firstNum'); // 10
  }
  var second = secondNum ?? 10;
  print('$second'); // 10
}
```

7. **Opérateurs de cascade (..)**

- La notation en cascade (`. .`) dans Dart vous permet d'effectuer une séquence d'opérations sur le même objet (y compris les appels de fonction et l'accès aux champs). Cette notation permet de garder le code Dart compact et supprime le besoin de créer des variables temporaires pour stocker les données.
- En Dart, la notation d'opérateurs de cascade (`. .`) est utilisée pour simplifier l'appel répété de méthodes sur le même objet. Elle permet d'appeler plusieurs méthodes sur un objet sans avoir à répéter le nom de l'objet à chaque fois. Voici comment cela fonctionne :
- Supposons que vous ayez un objet monObjet et que vous souhaitiez appeler plusieurs méthodes sur cet objet, voici comment vous pouvez utiliser l'opérateur de cascade :
- **Utilisation d'opérateur de cascade(..)**

```dart
class Person {
  String nom = "";
  int age = 0;

  void sePresenter() {
    print("Je m'appelle $nom et j'ai $age ans.");
  }
}

void main() {
```

apcpedagogie.com

```
  var personne = Person()
    ..nom = "Alice"
    ..age = 30;

  personne
    ..sePresenter()
    ..nom = "Bob"
    ..age = 25
    ..sePresenter();
}
```

- Dans cet exemple, nous créons une instance de la classe **Person** et utilisons l'opérateur de cascade pour définir le nom et l'âge de la personne, ainsi que pour appeler la méthode **sePresenter()** pour afficher une présentation.
- L'opérateur de `cascade (..)` est utile pour rendre le code plus lisible et éviter la répétition du nom de l'objet lorsque vous effectuez plusieurs opérations successives sur cet objet.
- **Sans utilisation d'opérateur de cascade(..)**

```
class Person {
  String nom = "";
  int age = 0;

  void sePresenter() {
    print("Je m'appelle $nom et j'ai $age ans.");
  }
}

void main() {
  var personne = Person();
  personne.nom = "Alice";
  personne.age = 30;
  personne.sePresenter();

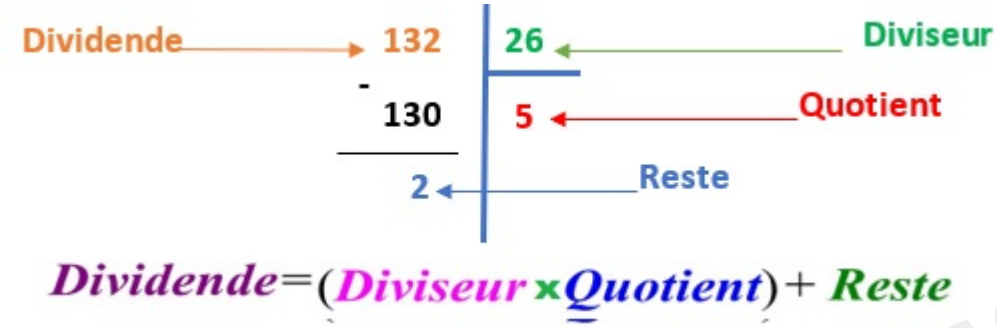
  personne.nom = "Bob";
  personne.age = 25;
  personne.sePresenter();
}
```

- Cependant, notez que sans l'opérateur de cascade, vous devez répéter le nom de l'objet (personne) à chaque fois que vous souhaitez appeler une méthode ou affecter une propriété. Cela peut rendre le code plus verbeux,

apcpedagogie.com

en particulier lorsque vous effectuez plusieurs opérations sur le même objet. L'opérateur de cascade est conçu pour simplifier ces scénarios en éliminant la répétition du nom de l'objet.

3. ## Division et modulo



- En arithmétique, la division euclidienne (dite aussi division) entière est une opération qui, à deux entiers naturels appelés dividende et diviseur, associe deux entiers appelés quotient et reste.
- Selon le format: **`Dividende=diviseur * quotient + Reste: Dividende=diviseur * quotient + Reste`**.
- Le signe "`/`" représente la division et le signe "`%`" représente le modulo (reste de la division Euclidienne)
- **Remarque**: le modulo est une fonction très importante en informatique (nous l'utiliserons dans la session relative aux dates), elle peut être décrite comme suit: **`x % y = x – y * (x/y).floor()`**

4. ## Les fonctions mathématiques basiques `ABS`, `ROUND`, `CEIL`, `FLOOR` et `TRUNCATE`

1. **La fonction abs**

   - La fonction **`abs`** rend la valeur absolue ce qui reviens en gros a enlever le signe " - " s'il existe. En langage mathématique cela correspond à " x " si x est positif et " -x " si x est négatif

```
// -4 -> 4 , 5 -> 5
print (a.abs());
print (bb.abs());
```

2. **La fonction `arrondi`**

   - La fonction round `arrondi` à la valeur inférieur si le nombre après la virgule est strictement inférieur à 0.5 et arrondi à la valeur supérieur si le nombre après la virgule et supérieur ou égale à 0.5

```
// 4.3 -> 4 , 4.5 -> 5 , 4.9 -> 5 , -1.51 -> -2
// -1.49 -> -1 , -1.50 -> -1 , -1.51 -> -2
print (pi.round());
```

3. **La fonction `ceil`**

   - La fonction `ceil` donne un arrondissement toujours à la valeur supérieure

```
// -4.3 -> -4 , 4.3 -> 5 , -1.50 -> -1
print (pi.ceil());
```

4. **La fonction `floor`**

   - La fonction `floor` renvoie la valeur inférieure

```
// -4.3 -> -5 , 4.3 -> 4 , -1.50 -> -2
print (pi.floor());
```

5. **La fonction `truncate`**

   - La fonction `truncate` quant à elle renvoie la valeur sans la partie après la virgule

```
// -1.49 -> -1 , 1.50 -> 1
print (pi.truncate());
```

5. Applications

1. **App01**

   - Quelle seraient les résultats de ces opérations ?

apcpedagogie.com

```dart
Run | Debug
main() {
  var operation1 = '13/2 = ${13 ~/ 2} r ${13 % 2}';

  var x = 'OK';
  int y = 10;

  var res = !((x == 'ok') && (y > 12));

  var operation2 = (res == true) ? null : y *= 2;

  print(operation1);
  print(res);
  print(operation2);
}
```

2. **App02: Vérification des notes d'étudiants**

1. **Enoncé**

- Écrivez un programme en Dart qui demande à l'utilisateur de saisir les notes d'un étudiant pour trois matières : mathématiques, physique et chimie. Le programme doit ensuite calculer la moyenne des notes et déterminer si l'étudiant est admis ou non, en utilisant les règles suivantes :
  - Pour être admis, l'étudiant doit avoir une moyenne d'au moins 50 dans chacune des trois matières.
  - Si la moyenne dans l'une des matières est inférieure à 50, l'étudiant est recalé.
  - Utilisez des opérateurs logiques pour effectuer ces vérifications. Voici un exemple de ce à quoi pourrait ressembler le code :

Solution

```dart
import 'dart:io';

void main() {
  stdout.write("Note en mathématiques : ");
  double math = double.parse(stdin.readLineSync()!);

  stdout.write("Note en physique : ");
```

apcpedagogie.com

```
  double physique = double.parse(stdin.readLineSync()!);

  stdout.write("Note en chimie : ");
  double chimie = double.parse(stdin.readLineSync()!);

  bool admission = (math >= 50) && (physique >= 50) && (chimie >= 50);

  if (admission) {
    print("L'étudiant est admis.");
  } else {
    print("L'étudiant est recalé.");
  }
}
```

3. **App03: Vérification d'éligibilité à un permis de conduire**

    1. **Enoncé**

        - L'objectif de cet exercice est de créer un programme en Dart qui vérifie l'éligibilité d'un candidat à un permis de conduire en fonction de son âge, de sa nationalité, et de la possession d'un permis existant. Le programme demandera à l'utilisateur de saisir ces informations et indiquera si le candidat est éligible ou non.
        - Voici les critères pour être éligible à un permis de conduire :
            - Le candidat doit avoir au moins 18 ans.
            - Le candidat doit être citoyen du pays (nationalité).
            - Le candidat doit déjà posséder un permis de conduire d'une catégorie spécifique (par exemple, moto ou voiture).
            - Utilisez les opérateurs relationnels (<, >, <=, >=, ==, !=), les opérateurs d'essai de type (is, as, is!), et les opérateurs logiques (&&, ||, !) pour effectuer ces vérifications.

        Solution

```
import 'dart:io';

void main() {
  stdout.write("Âge du candidat : ");
  int age = int.parse(stdin.readLineSync()!);
```

apcpedagogie.com

```
  stdout.write("Est citoyen du pays (true/false) : ");
  bool estCitoyen = (stdin.readLineSync()!.toLowerCase() == "true");

  stdout.write("Possède un permis de conduire existant (true/false) : ");
  bool aUnPermis = (stdin.readLineSync()!.toLowerCase() == "true");

  if (age >= 18 && estCitoyen && aUnPermis) {
    print("Le candidat est éligible à un permis de conduire.");
  } else {
    print("Le candidat n'est pas éligible à un permis de conduire.");
  }
}
```

2. **App04: Une mini-calculatrice interactive**

    1. **Enoncé**

        - Créer un programme en Dart qui simule une mini-calculatrice interactive.
        - Le programme doit effectuer les tâches suivantes :
            - Saluer l'utilisateur avec le message "Bonjour !".
            - Demander à l'utilisateur de saisir deux nombres entiers, en redemandant la saisie en cas de non-saisie d'un entier.
            - Réaliser les opérations d'addition, de soustraction, de multiplication et de division sur ces deux nombres avec une précision au millième.
            - Afficher les résultats de chaque opération.
            - Dire "Au revoir !" pour clôturer l'interaction.

        Exemple d'utilisation

        Bonjour ! Veuillez saisir le premier nombre : 4 Veuillez saisir le deuxième nombre : abc Ce n'est pas un nombre entier. Veuillez réessayer. Veuillez saisir le deuxième nombre : 7 Calculs : 4 + 7 = 11 4 - 7 = -3 4 * 7 = 28 4 / 7 = 0.571 Au revoir !

    Solution

```
import 'dart:io';
```

apcpedagogie.com

```
void main() {
  print('Bonjour !');

  int nombre1 = 0, nombre2 = 0;
  bool estEntier = false;

  while (!estEntier) {
    stdout.write('Veuillez saisir le premier nombre : ');
    var input1 = stdin.readLineSync();

    if (int.tryParse(input1!) != null) {
      nombre1 = int.parse(input1);
      estEntier = true;
    } else {
      print('Ce n'est pas un nombre entier. Veuillez réessayer.');
    }
  }

  estEntier = false;

  while (!estEntier) {
    stdout.write('Veuillez saisir le deuxième nombre : ');
    var input2 = stdin.readLineSync();

    if (int.tryParse(input2!) != null) {
      nombre2 = int.parse(input2);
      estEntier = true;
    } else {
      print('Ce n'est pas un nombre entier. Veuillez réessayer.');
    }
  }

  // Effectuer les calculs
  int addition = nombre1 + nombre2;
  int soustraction = nombre1 - nombre2;
  int multiplication = nombre1 * nombre2;
  double division = nombre1 / nombre2;

  print('Calculs :');
  print('$nombre1 + $nombre2 = $addition');
  print('$nombre1 - $nombre2 = $soustraction');
  print('$nombre1 * $nombre2 = $multiplication');
  print('$nombre1 / $nombre2 = ${division.toStringAsFixed(3)}');
```

apcpedagogie.com

```
  print('Au revoir !');
}
```

Cours Flutter

Source : https://apcpedagogie.com/les-operateurs-dans-le-langage-dart · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com