**FUNÇÕES**

**Flutter pub get = para trazer as dependências do projeto;**

**Flutter pub clean = Limpar as dependências do projeto;**

**Atividade 01.**

**a)**

import 'dart:io';

main() {

print('Calcule a média - ');

print('Digite a sua PRIMEIRA nota:');

var entradaNotaUm = stdin.readLineSync()!;

var notaUm = int.parse(entradaNotaUm);

print('Digite a sua SEGUNDA nota:');

var entradaNotaDois = stdin.readLineSync()!;

var notaDois = int.parse(entradaNotaDois);

var mediaAluno = (notaUm + notaDois) / 2;

print('Sua média é: $*mediaAluno*');

}

**Atividade 01.**

**b)**

import 'dart:io';

main() {

print('Calcule a média - ');

print('Digite a sua PRIMEIRA nota:');

var entradaNotaUm = stdin.readLineSync()!;

var notaUm = int.parse(entradaNotaUm);

print('Digite a sua SEGUNDA nota:');

var entradaNotaDois = stdin.readLineSync()!;

var notaDois = int.parse(entradaNotaDois);

if (notaUm >= 0 && notaDois <= 10) {

var mediaAluno = (notaUm + notaDois) / 2;

print(mediaAluno >= 6 ? 'Aluno Aprovado' : 'Aluno Reprovado');

} else {

print('Números Invalidos!');

}

}

**Atividade 01.**

**c)**

import 'dart:io';

main() {

print('Calcule a média - ');

print('Digite a sua PRIMEIRA nota:');

var entradaNotaUm = stdin.readLineSync()!;

var notaUm = int.parse(entradaNotaUm);

print('Digite a sua SEGUNDA nota:');

var entradaNotaDois = stdin.readLineSync()!;

var notaDois = int.parse(entradaNotaDois);

var mediaAluno = (notaUm + notaDois) / 2;

print('Sua média é: $*mediaAluno*');

print(mediaAluno >= 60 ? 'Aluno Aprovado' : 'Aluno Reprovado');

}

**Atividade 02.**

1. **Em programação o null é um problema? Explique.**

Sim, o **NULL** pode ser um problema em programação, especialmente em linguagens que permitem o seu uso.

**Exemplo:**

O uso indiscriminado de null pode tornar o código mais difícil de entender e dar origem a erros difíceis de rastrear. Em muitos casos, é difícil determinar se um valor null é um estado válido ou se é um indicador de erro ou condição especial.

**Atividade 02.**

1. **O que é null safety? para que serve? Quais são as vantagens e desvantagens?**

Null safety é um conceito em programação que se refere à capacidade de uma linguagem de programação ou de um sistema de tipos garantir que referências nulas (ou null) sejam tratadas de maneira segura, prevenindo erros de acesso a objetos nulos em tempo de execução.

**Para que serve? Facilitar a manutenção do código.**

Código que é null safe tende a ser mais fácil de manter, pois reduz a necessidade de verificações repetitivas de null e torna mais claro onde e como os valores nulos são tratados.

**Atividade 02.**

1. **Faça um exemplo do uso incorreto de um dos operadores Null Safety.**

function getUserName(user: User | null): string {

return user.name; // Uso incorreto do operador de null safety

}

interface User {

name: string;

age: number;

}

const user: User | null = null;

const userName = getUserName(user);

console.log(userName);

**Atividade 04.**

1. **Quais são os tipos de parâmetros em dart? Explique as diferenças, vantagens e desvantagens de cada uma.**

* **Parametros Posicionais:** 
  + **Diferença:** Parâmetros posicionais são aqueles em que a correspondência entre o argumento passado e o parâmetro formal é feita com base na posição.
  + **Vantagens:**
    - Simplicidade: São simples de usar e entender, pois a ordem dos argumentos geralmente corresponde diretamente à ordem dos parâmetros.
    - Compatibilidade: Funcionam bem com funções e métodos que têm um pequeno número de parâmetros.
  + **Desvantagens:**
    - Menos Flexibilidade: Se você precisar adicionar novos parâmetros no meio da lista, isso pode afetar todas as chamadas existentes, pois elas precisarão ser ajustadas para refletir a nova ordem.
* **Paramentros Nomeados:**
  + **Diferença:** Parâmetros nomeados são identificados pelo nome, o que significa que a correspondência entre o argumento passado e o parâmetro formal é feita explicitamente pelo nome do parâmetro.
  + **Vantagens:**
    - Clareza: A utilização de nomes torna a chamada do método ou função mais clara, pois você pode identificar facilmente qual argumento corresponde a qual parâmetro.
    - Flexibilidade: Adicionar ou reorganizar parâmetros não afeta as chamadas existentes, pois elas são identificadas pelos nomes.
  + **Desvantagens:**
    - Sintaxe adicional: Pode exigir mais digitação ao chamar a função devido à necessidade de especificar explicitamente os nomes dos parâmetros.
* **Parametros Opcionais:**
  + **Diferença:** Parâmetros opcionais permitem que você defina argumentos que não são necessários para chamar a função ou método.
  + **Vantagens:**
    - Flexibilidade: Permitem que você crie funções ou métodos com argumentos que podem ser omitidos se não forem necessários.
    - Redução da Sobrecarga de Sobrecargas: Você não precisa criar várias versões de uma função com diferentes números de parâmetros. Em vez disso, pode ter uma única função com parâmetros opcionais.
  + **Desvantagens:**
    - Menos clareza: Pode ser menos claro para quem chama a função entender quais argumentos são obrigatórios e quais são opcionais.
    - Potencial para Ambiguidade: O uso excessivo de parâmetros opcionais pode levar à ambiguidade e confusão sobre como usar a função corretamente.