03 - Trabalhando com Arrays e Listas

O Dart tem suporte integrado para coleções de listas, conjuntos e mapas.

List

Talvez a coleção mais comum em quase todas as linguagens de programação seja o array, ou grupo ordenado de objetos. Em Dart, arrays são objetos List, então a maioria das pessoas os chama apenas de listas.

```
// Dart infere que a lista tem o tipo List<int>
var list = [1, 2, 3];

// Dart infere que a lista tem o tipo List<String>
var list = [
  'Car',
  'Boat',
  'Plane',
];
```

Listas usam indexação baseada em zero, onde 0 é o índice do primeiro valor e list.length - 1 é o índice do último valor. Você pode obter o comprimento de uma lista usando a propriedade .length e acessar os valores de uma lista usando o operador subscrito ([])

```
var list = [1, 2, 3];
assert(list.length == 3);
assert(list[1] == 2);

list[1] = 1;
assert(list[1] == 1);
```

Para criar uma lista que seja uma constante de tempo de compilação, adicione const antes do literal da lista:

Sets

Um conjunto (set) em Dart é uma coleção não ordenada de itens únicos. O suporte do Dart para conjuntos (sets) é fornecido por literais de conjunto (set) e o tipo Set.

A implementação padrão do Set, LinkedHashSet, considera objetos indistinguíveis se eles forem iguais em relação a Object.== e Object.hashCode.

A iteração sobre elementos de um conjunto (set) pode ser desordenada ou ordenada de alguma forma. Exemplos:

- Um `HashSet` não é ordenado, o que significa que sua ordem de iteração não é especificada

- `LinkedHashSet` itera na ordem de inserção de seus elementos e
- um conjunto (set) classificado como `SplayTreeSet` itera os elementos em ordem classificada.

```
// Criando um conjunto (set) de números
  Set<int> numbers = {1, 3, 5, 7, 3, 5, 9, 2, 1};

// Criando um conjunto (set) de Strings
var nomes = {'Florentina', 'Tiririca', 'Ronald McDonalds'};
```

Neste exemplo, estamos criando um conjunto (set) que contém os números inteiros 1, 3, 5, 7, 3 (repetido), 5 (repetido), 9, 2 e 1 (repetido). Como um conjunto (set) só aceita elementos únicos, as duplicatas serão automaticamente removidas.

Para criar um conjunto (set) vazio, use {} precedido por um argumento de tipo ou atribua {} a uma variável do tipo Set:

```
var nomes = <String>{};
// Set<String> nomes = {}; // Isso também funciona.
// var nomes = {}; // Cria um map, não um conjunto (set).
```

Para adicionarmos itens a um conjunto existente, podemos utilizar os métodos add() ou addAll():

```
var elementos = <String>{};
elementos.add('ET Bilu');
elementos.addAll(nomes);
```

Use .length para obter o número de itens no conjunto:

```
elementos.length
print(elementos.length) //Saída: 4
```

Para criar um conjunto (set) que seja uma constante de tempo de compilação, adicione const antes do literal do conjunto (set)

```
final constanteSet = const {
  'Bruno Lima'
}

// constanteSet.add('Samara Ogata'); // Esta linha causará um erro.
```

Maps

Os mapas funcionam como dicionários, associando chaves a valores. Eles oferecem uma grande flexibilidade em termos de organização de dados.

Tanto chaves quanto valores podem ser qualquer tipo de objeto. Cada chave ocorre apenas uma vez, mas você pode usar o mesmo valor várias vezes.

O suporte Dart para mapas é fornecido por literais de `map` e o `Map Type`.

Mapas, e suas chaves e valores, podem ser iterados. A ordem de iteração é definida pelo tipo individual de mapa. Exemplos:

- O `HashMap` simples não é ordenado (nenhuma ordem é garantida), o
- `LinkedHashMap` itera na ordem de inserção da chave,
- e um mapa ordenado como o `SplayTreeMap` itera as chaves na ordem ordenada.

```
// Dart infere que pessoas tem o tipo Map<String, String>
var pessoas = {
  // Key:    Value
  '01': 'Bruno Lima',
  '02': 'Samara Ogata',
  '03': 'João Miguel'
};
```

Você pode criar os mesmos objetos usando um construtor Map:

```
var pessoas = Map<String, String>();

pessoas['01'] = 'Bruno Lima';
pessoas['02'] = 'Samara Ogata';
pessoas['03'] = 'João Miguel';

//Adicione um novo par de chave-valor a um map existente usando o operador
de atribuição de subscrito ([]=)
pessoas['04'] = 'Edi Ogata'; // Adicione um par chave-valor
```

Manupule os elementos do seu Map

```
// Recuperar um valor de um map usando o operador subscrito ([])
pessoas['03'] //Saída: 'João Miguel'

//Se você procurar uma chave que não esteja em um map, receberá null em
troca
pessoas['05'] //Saída: null

// Use .length para obter o número de pares de chave-valor no map
print(pessoas.length) //Saída: 4
```

Para criar um `map` que seja uma constante de tempo de compilação, adicione const antes do literal do `map`

```
final constantMap = const {
  2: 'helium',
  10: 'neon',
  18: 'argon',
};

// constantMap[2] = 'Helium'; // Esta linha causará um erro.
```