**Vetor Dinâmico Genérico em Dart**

Desenvolva, em Dart, uma classe chamada VetorDinamico<T> que represente um vetor genérico e dinâmico, ou seja, que possa armazenar qualquer tipo de dado e crescer automaticamente quando precisar.

Após criar a estrutura do vetor, deverá implementar todas as funções obrigatórias e, na main(), testar todas as funções implementadas, demonstrando que o vetor funciona corretamente.

**Regras de negócio**

1. O vetor deve ser **genérico**: deve funcionar com int, double, String, ou qualquer outro tipo.
2. Deve possuir uma **capacidade inicial** (por padrão, 4 elementos).
3. Sempre que faltar espaço para uma nova inserção, a capacidade deve **dobrar** (ex.: 4 → 8 → 16).
4. A classe deve armazenar os elementos em uma lista interna (List<T?>) e controlar a quantidade real de elementos inseridos com um contador (\_tamanho).
5. O acesso aos elementos deve respeitar os limites do vetor. Qualquer tentativa de acessar uma posição inválida deve lançar um “RangeError”.
6. As operações de adição, remoção e inserção devem manter o vetor consistente, deslocando elementos quando necessário.
7. O método toString() deve imprimir apenas os elementos válidos.

**Funções obrigatórias a serem implementadas**

**Construtor**

VetorDinamico([int capacidadeInicial = 4])

* Cria o vetor com capacidade inicial indicada (mínimo 1).

**Consultas**

int tamanho();

// Retorna a quantidade de elementos realmente armazenados.

bool estaVazio();

// Retorna true se não houver nenhum elemento.

int capacidade();

// Retorna a capacidade interna atual.

**Acesso por índice**

T operator [](int indice);

// Retorna o elemento da posição indicada. Se for inválido, lança RangeError.

void operator []=(int indice, T valor);

// Atualiza o elemento da posição indicada. Se for inválido, lança RangeError.

**Inserção e remoção**

void adicionar(T valor);

// Adiciona o valor no fim do vetor, aumentando a capacidade se necessário.

void inserir(int indice, T valor);

// Insere na posição indicada (0 <= indice <= tamanho).

// Desloca os elementos à direita para abrir espaço.

T removerEm(int indice);

// Remove e retorna o valor na posição indicada (0 <= indice < tamanho).

// Desloca os elementos à esquerda para preencher o espaço.

void limpar();

// Remove todos os elementos, mas mantém a capacidade alocada.

**Busca/utilidades**

int indiceDe(T valor);

// Retorna o índice da primeira ocorrência do valor, ou -1 se não encontrar.

bool contem(T valor);

// Retorna true se o valor existir no vetor.

@override

String toString();

// Retorna os elementos no formato [e1, e2, e3], apenas até o tamanho real.

**Suporte interno (privado)**

void \_garantirCapacidade(int capacidadeMinima);

// Se a capacidade atual for menor que a necessária, dobra repetidamente

// até que caiba o novo elemento. Copia os dados para a nova lista interna.

**O que entregar**

* A classe VetorDinamico<T> completa, com todos os métodos acima implementados.
* Um main() que:
  + Crie um vetor de inteiros e outro de strings para testar.
  + Teste **todas as funções obrigatórias** (adição, remoção, inserção, acesso, busca, limpar, etc.).
  + Mostre a saída no console, validando que o comportamento está correto.

**Critérios de avaliação**

1. **Corretude**: todos os métodos implementados de acordo com as regras.
2. **Tratamento de erros**: índices inválidos devem lançar RangeError.
3. **Uso adequado de genéricos e null safety**: a classe deve ser genérica (<T>) e não vazar null para fora da interface pública.
4. **Cobertura de testes**: o main() deve chamar e demonstrar o uso de todas as funções obrigatórias.
5. **Organização do código**.