MC322 - Object Oriented Programming
Lab 3
Estrutura de Dados e Enums

Instituto de Computação - UNICAMP

Sumário

Collection

Interfaces Derivadas de Collection

ArrayList

HashSet

Exercicio 1

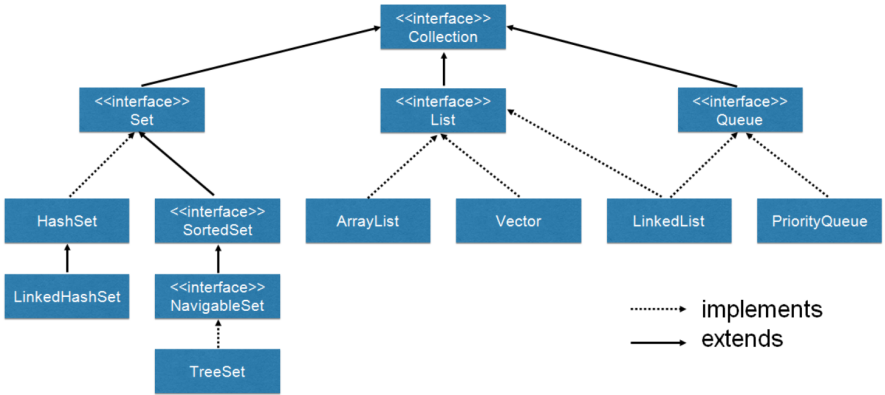
Enums

Exercicio 2

Collection

Collection

Collection é uma interface em Java que serve como raiz para a maior parte das estruturas de dados. Sua definição é ser um **grupo de elementos**



Collections

Essa interface estabelece a implementação de um certo conjuntom de métodos, a lsita inteira pode ser conferida na documentação, mas alguns dos principais são:

- **add(E e):** método opcional para adição de um certo elemento em uma coleção
- **contains(Object o):** operação booleana para saber se a coleção possui certo elemento
- **size():** retorna o tamanho em inteiro da coleção
- **isEmpty()**: retorna true se a coleção estiver sem nenhum elemento
- **remove(Object o):** método opcional, remove uma única instância de elemento na coelação

Interfaces Derivadas de Collection

Interfaces derivadas de Collection

Como mostrado na figura anterior existem algumas interfaces que extendem a interface Collection, sendo as principais:

1. **List:** coleção ordenada que pode conter elementos duplicados e que podem ser acessados por indez
2. **Set:** coleção sem elementos duplicados que corresponde a representação matemática de conjuntos, os seus métodos fornecem operações equivalentes as tradicionais de conjunto como união e interseção
3. **Queue:** Geralmente utilizadas para guardar elementos antes de um processamento, possuem métodos adicionais para inserção, remoção e inspeção. Costumeiramente possuem a ordenação em FIFO (First in, First Out)

ArrayList

ArrayList

Implementação da interface List que permite a adição de todos os tipos de elementos. Algumas características importantes são:

- Você pode especificar o tipo de ArrayList que vai ser implementado, mas se não for passado é permitido qualquer tipo.

```
ArrayList<String> list = new ArrayList<>();
```

ArrayList

Implementação da interface List que permite a adição de todos os tipos de elementos. Algumas características importantes são:

- Você pode especificar o tipo de ArrayList que vai ser implementado, mas se não for passado é permitido qualquer tipo.
- Tem como vantem em relação a um Array serem dinamicamente alocadas e terem uma tipagem mais segura

```
ArrayList<String> list = new ArrayList<>();
list.add("Hello")
String arr = new String[5]
arr[0] = "Hello"
```

ArrayList - Métodos Importantes

Alguns métodos relevantes para ArrayList em java são:

- **get(int index):** retorna o elemento na posição especificada
- **set(int index, E element):** troca o elemento na posição passada pelo contido no parametro
- **indexOf(Object o):** Retorna o indice da primeira ocorrência do objeto passado
- **add(int index, E element:** Adiciona um elemento na posição especificada, dando um shift dos elementos para à direita
- **sort(Comparator c)**: ordena a lista, é possível especificar pelo comparador a ordem de como vai ser ordenado

[Documentação]

HashSet

HashSet

Implementação da interface Set que armazena elementos únicos. Algumas características importantes são:

- Você pode especificar o tipo dos elementos assim como em ArrayList para ter tipagem segura
- Não permite elementos duplicados, ignorando caso tentando
- A ordem dos elementos não é garantida
- Desempenho eficiente em operações básicas como `add`, `remove` e `contains`

```
HashSet<String> frutas = new HashSet<>();
frutas.add("Maçã");
frutas.add("Banana");
frutas.add("Maçã"); // Nao tem efeito
System.out.println(frutas);
```

HashSet - Métodos Importantes

Principais métodos da implementação:

- **add(E e):** Adiciona elemento se não existir (retorna `true` se adicionado).
- **remove(Object o):** Remove elemento se presente (retorna `true` se removido).
- **contains(Object o):** Verifica existência em tempo constante.
- **size():** Retorna quantidade de elementos únicos.

[Documentação]

Exercicio 1

Exercício 1

Você tem um sistema de gerenciamento de filmes, que contem uma classe Filme com atributos: nota, gênero e título do filme. Além dessa classe existe também a SistemaFilmes, essa classe contém uma lista de filmes e um conjunto de títulos. Você vai precisar adicionais dois métodos para passar em testes unitários criados:

1. addTitutlo(Filme filme): Esse método deve adicionar um Filme novo a lsita de filmes do SistemaFilme, retornando `true` se deu certo ou false se o título já constava no sistema
2. medianaNotas(): Esse método deve retornar a mediana das notas dos filmes adicionados no sistema ou -1 se não exister nenhum filme adicionado,

Utilize ArrayList e HashSet para resolver esses problemas, lembre-se dos atributos mostrados nos slides. Como comparador fica a sugestção do `Comparator.naturalOrder()`.

Enums

Enums

Estrutura que define um conjunto fixo de constantes nomeadas com comportamentos e propriedades. Principais características:

- Como são constantes a conveção é que elas devem ser EM MAIÚSCULO
- Podem ter métodos, campos e construtores
- Implementam métodos como `values()` e `valueOf()` automaticamente
- Muito usados quando necessário trabalhar com constantes ou operadores switch

[Documentação]

Enums - Exemplo

```
public enum DiaDaSemana {

    QUARTA("Quarta-feira", 3),
    SABADO("Sábado", 6),
    DOMINGO("Domingo", 7);

    // Atributos
    private final String nome;
    private final int numero;

    // Construtor
    DiaDaSemana(String nome, int numero) {
        this.nome = nome;
        this.numero = numero;
    }
    //Método
    public boolean verificaDiaUtil() {
        return this != SABADO && this != DOMINGO;
    }
}
```

Enums - Exemplo

A instanciação e chamada de método do Enum mostrado seria:

```
DiaDaSemana dia = DiaDaSemana.QUARTA;
System.out.println("    dia útil? " + dia.verificaDiaUtil());
```

Enums - Métodos values() e e valueOf()

O método .values() retorna um array de Strigns com todas as constantes na ordem de declaração

```
DiaDaSemana.values() //{QUARTA, SABADO, DOMINGO}
```

Já o método `.valueOf(Class<T> enumType, String name)` retorna a constante do tipo e nome específicado, cuidado pois ele é CASE SENSETIVE

```
DiaDaSemana var = DiaDaSemana.valueOf(SEGUNDA)
```

Enums e Métodos Abstratos

É possivel para um Enum ter um método abstrado ou implementar um interface. Exemplo:

```
public enum DiaDaSemana {

    QUARTA("Quarta-feira", 3){
        @Override
        public boolean verificaDIaUtil() {
            return false;
        }
    };
    private final String nome;
    private final int numero;
    DiaDaSemana(String nome, int numero) {
        this.nome = nome;
        this.numero = numero;
    }
    public abstract boolean verificaDiaUtil();
}
```

Exercicio 2

Exercício 2

Você deve criar um Enum chamado Operacão que pra implementa o método calcula(int var1, int var2) para cada uma das operações aritméticas básicas (SOMA, SUBTRACAO, MULTIPLICACAO, DIVISAO). O objetivo é passar em todos os testes.

Referências

- Effective Java 3rd Edition.
- Oracle. (n.d.). Java Documentation, The Java Tutorials. Acessado em 16 de Março de 2025. Disponível em: https://docs.oracle.com/javase/tutorial/collections/index.html