C206 - POO

Arthur Openheimer

Inatel

Informações Gerais

- Atendimento → Terça-feira 17:30-19:10, prédio 1, sala 19
- Email → arthur.openheimer@ges.inatel.br
- Github →https://github.com/ArthurOpenheimer/C206-Monitoria

Inatel

Collections

Para facilitar o trabalho com arrays, existem classes e interfaces disponíveis no pacote java.util que implementam diversos métodos e estruturas, dentre elas:

- ArrayList
- List
- Collections
- LinkedList
- HashMap
- HashSet

ArrayList

Funciona de forma similar ao array padrão, por isso, vamos repetir as operações de array que já implementamos anteriormente no seguinte cenário:

Uma sistema de jogo de cartas onde um jogador tem uma mão, que consiste em uma lista de cartas

ArrayList

Importar a coleção

Declarar o ArrayList nos atributos da classe e no construtor(composição)

Métodos do ArrayList

```java
import java.util.ArrayList;

public class Jogador {
    private String nome;
    private int idade;
    private ArrayList<Carta> mao;

    public Jogador(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
        this.mao = new ArrayList<Carta>();
    }

    public void adicionarCarta(Carta carta) {
        mao.add(carta);
    }

    public void removerCarta(Carta carta) {
        mao.remove(carta);
    }

    public void mostrarMao() {
        for (Carta carta : mao) {
            System.out.println(carta);
        }
    }
}
```

Inatel

Métodos mais utilizados do ArrayList

- set: Substitui o elemento em uma posição específica da lista
- get: Retorna o elemento na posição especificada da lista
- add: Adiciona um elemento ao final da lista
- addAll: Adiciona todos os elementos de uma coleção à lista
- remove: Remove o elemento ou o elemento na posição especificada
- removeAll: Remove todos os elementos que existem em uma coleção da lista
- indexOf: Retorna o índice da primeira ocorrência de um elemento na lista
- forEach: Executa uma ação para cada elemento da lista
- contains: Verifica se a lista contém o elemento especificado
- isEmpty: Verifica se a lista está vazia
- lastIndexOf: Retorna o índice da última ocorrência de um elemento na lista
- size: Retorna o número de elementos na lista

Classe Collections

A classe Collections possui vários métodos estáticos para auxiliar o trabalho com listas, porém, para utilizar alguns deles precisamos fazer algumas implementações antes, vamos ver a implementação no cenário do jogador:

Classe Collections

Importar a classe

Main

```java
import java.util.Collections;

public class Main {
    Run | Debug
    public static void main(String[] args) {

        Jogador jogador = new Jogador(nome:"João", idade:20);
        Carta carta1 = new Carta(nome:"Ás", valor:1);
        Carta carta2 = new Carta(nome:"Rei", valor:13);

        jogador.adicionarCarta(carta1);
        jogador.adicionarCarta(carta2);

        Collections.sort(jogador.getMao());
    }
}
```

Chamando o método abstrato sort para ordenar as cartas na mão do jogador

Mas como a classe Collections sabe qual parâmetro utilizar como comparação na ordenação?

Inatel

Classe Collections

Sobrescrever o método compareTo e implementar a lógica necessária para definir a regra de comparação

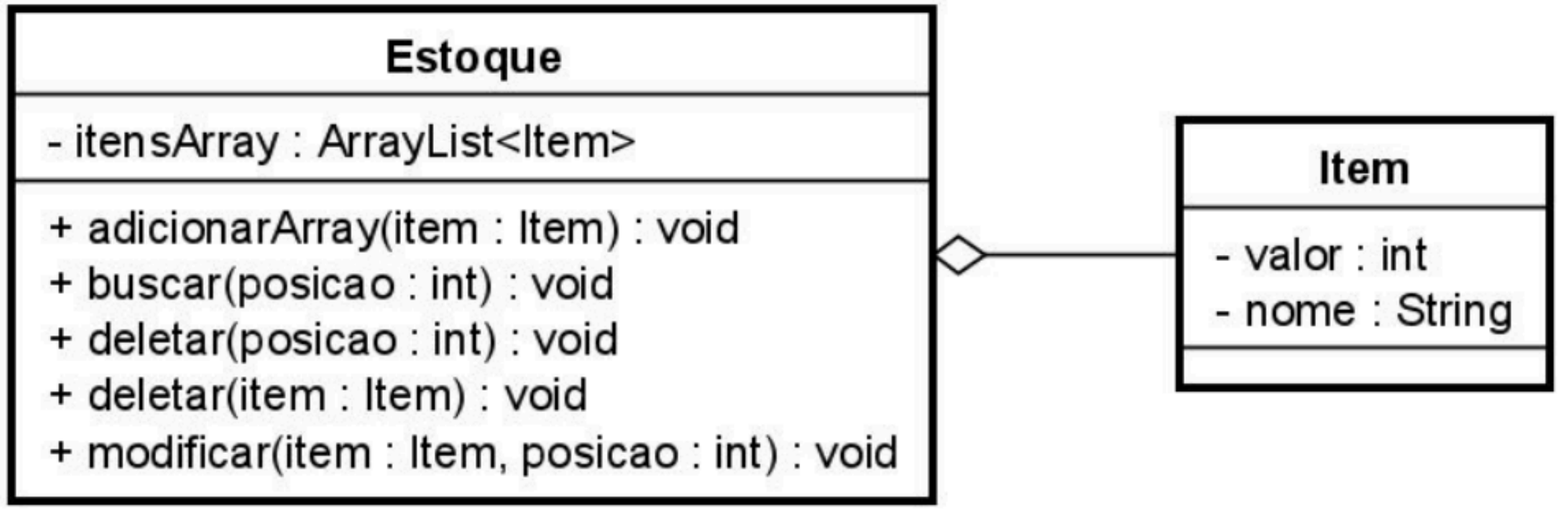
```java
public class Carta implements Comparable<Carta> {

    public String nome;
    public int valor;

    public Carta(String nome, int valor) {
        this.nome = nome;
        this.valor = valor;
    }

    @Override
    public int compareTo(Carta carta) {
        return Integer.compare(this.valor, carta.valor);
    }
}
```

Implementar a interface Comparable e definir a classe do objeto que será comparado

Nesse caso, utilizamos a classe Integer para comparar os valores inteiros do valor da carta

Inatel

Exercício



- Teste o método sort da classe Collections na main, fazendo a ordenação por ordem decrescente de valor;
- TODOS os métodos do estoque devem utilizar os métodos da coleção ArrayList;
- Chame todos os métodos na main e mostre as informações do estoque e dos itens;
- Opcional: construtores e método para mostrar informações

Inatel