Monitoria POO
Implementando Lista de Objetos

Monitor: Johnes Thomas

 /johnesthomas

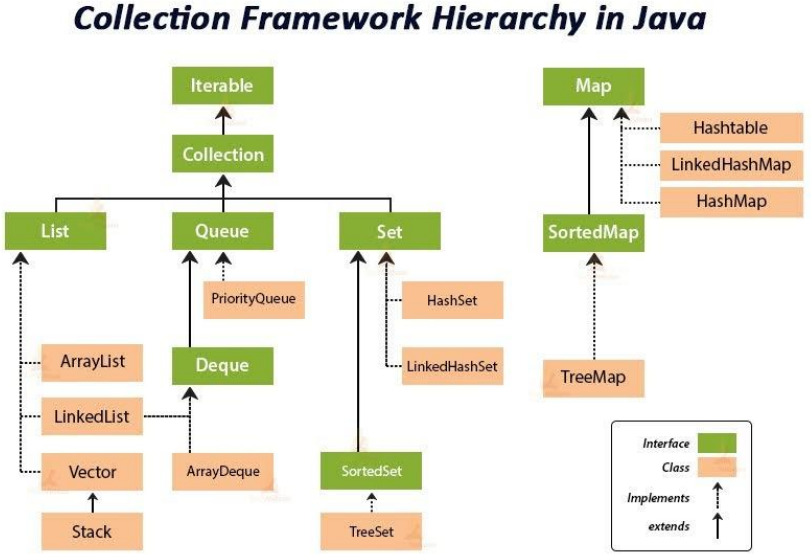
 /johnesthomas

2020 - ANO DO FOCO

GRUPO TIRADENTES
Inspirando o futuro

Collections

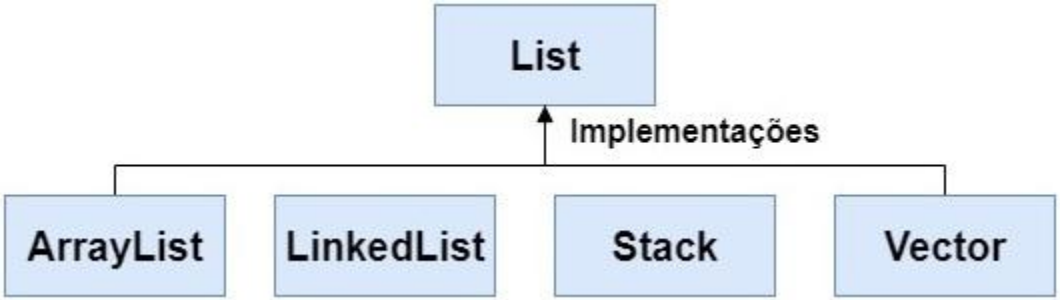
Uma Collection representa um grupo de Objects. Algumas Collections permitem Objects duplicados, outras não. Na hierarquia a classe Collection é uma subInterface derivada da superInterface **Iterable<E>**, que é uma subClasse da superClasse **Object**, e por conta disso alguns métodos são herdados, como o equals(), toString(), entre outros.



Fonte: http://www.mauda.com.br/?p=2300

List<E>

A **List** é uma subInterface da interface **Collection**. Nessa interface o usuário tem um controle preciso sobre onde na lista cada elemento é inserido. O usuário pode acessar os elementos por seu índice e pesquisar por elementos na lista. Essa interface herda alguns métodos da sua superInterface, como os métodos: *add*(E e), *remove*(int index), *size*(), *equals*(Object o), *contains*(Object o), entre vários outros.



ArrayList<E>

ArrayList é uma classe que implementa a interface List<E>. Sendo assim implementa todas as operações da interface e permite todos os elementos, inclusive **null**. Podemos considerar essa classe como um **Array** dinâmico. Todos ArrayList tem uma capacidade, a capacidade é o tamanho do array usado para armazenar os elementos da lista.

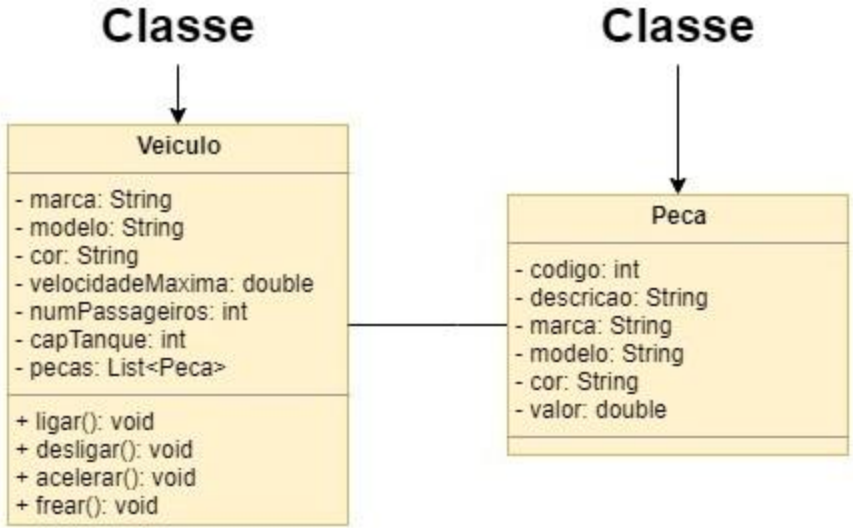
O "**<E>**" ao lado direito da classe é o parâmetro que o ArrayList vai receber, ou seja, é a especificação do tipo do elemento que será a lista. Só é aceito elementos do tipo classe, podendo ser uma lista de Objects, String, entre outros.

Ex.: ArrayList<String> ou ArrayList<Objects>

Alguns métodos do ArrayList

- void **add**(int index, **E** element) = Insere o elemento especificado na posição especificada nesta lista;
- boolean **add**(**E** e) = Acrescenta o elemento especificado ao final desta lista;
- void **clear**() = Remove todos os elementos desta lista;
- boolean **contains**(**Object** o) = Retorna true se esta lista contém o elemento especificado;
- **E get**(int index) = Retorna o elemento na posição especificada nesta lista;
- int **indexOf**(**Object** o) = Retorna o índice da primeira ocorrência do elemento especificado nesta lista, ou -1 se a lista não contiver elemento.
- boolean **isEmpty**() = Retorna true se esta lista não contiver elementos;
- **E remove**(int index) = Remove o elemento na posição especificada nesta lista;
- boolean **remove**(**Object** o) = Remove a primeira ocorrência do elemento especificado desta lista, se estiver presente;
- int **size**() = Retorna o número de elementos nesta lista.

Diagrama de classe com lista de objetos



Classe Veiculo com atributo do tipo ArrayList

```java
public class Veiculo {

    private String marca;
    private String modelo;
    private String cor;
    private double velocidadeMaxima;
    private int numPassageiros;
    private int capTanque;
    private ArrayList<Peca> pecas;
```

Implementando ArrayList:

- Instanciando;
- Adicionando;
- Imprimindo;
- Removendo;
- Recuperando.

Bora praticar !

```java
public class Programa {

    public static void main(String[] args) {

        // Veiculo automovel = new Veiculo();

        // automovel.setMarca("Renault");
        // automovel.setModelo("Kwid");
        // automovel.setCor("Branco");
        // automovel.setVelocidadeMaxima(156.00);
        // automovel.setNumPassageiros(5);
        // automovel.setCapTanque(38);

        // (1) Instanciando uma lista de peças

        ArrayList<Peca> listaPecas = new ArrayList<Peca>();

        // (2) Adicionando peças à lista

        listaPecas.add(new Peca(500, "Pneu", "Continental", "165/70 R14", "Preto", 200.00)); // 1
        listaPecas.add(new Peca(501, "Porta", "Fiat", "XX", "Branca", 800.00)); // 2

        // (3) Imprimindo a lista de peças

        for (Peca peca : listaPecas) {

            System.out.println("Codigo: " + peca.getCodigo());
            System.out.println("Descrição: " + peca.getDescricao());
            System.out.println("Marca: " + peca.getMarca());
            System.out.println("Modelo: " + peca.getModelo());
            System.out.println("Cor: " + peca.getCor());
            System.out.printf("Valor: %.2f %n", peca.getValor());
            System.out.println("================");

        }

        // (4) Remover peça da lista

        listaPecas.remove(1);

        System.out.println("Depois do remove()");
        System.out.println();

        for (Peca peca : listaPecas) {

            System.out.println("Codigo: " + peca.getCodigo());
            System.out.println("Descrição: " + peca.getDescricao());
            System.out.println("Marca: " + peca.getMarca());
            System.out.println("Modelo: " + peca.getModelo());
            System.out.println("Cor: " + peca.getCor());
            System.out.printf("Valor: %.2f %n", peca.getValor());
            System.out.println("================");

        }
```

Referências

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Object.html - **Object**

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Iterable.html - **Iterable**

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Collection.html# - **Collection**

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html - **List**

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html - **ArrayList**

http://www.mauda.com.br/?p=2300 - **img_hierarquia_collection_java**