













,Compartilhar o seu link com: luciorocha @ professores.utfpr.edu.br

Nome:

Link:

Sefora Davanzo <  Aula 26 - POO >
Filipe Augusto Parreira <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Angélica <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Roberto Furlani Neto <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Rafael Kendy <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Matheus e Thiago<  Cópia de Aula 26 - POCO4A - Exercícios propostos >
João Vitor N. Yoshida <  Cópia de Aula 26 - POCO4A - Exercícios propostos >.
Deivid da Silva Galvão<  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Isabella Melo <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Raphael Uematsu <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Plínio <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Marcos Tadao Shoji<  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Gabriel Takeshi <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Julio Farias <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
João Pedro Cavani Meireles<  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Thales Hasegawa <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Mabyly <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Daniel Martins de Carvalho <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Vitor Viana<  Cópia de Aula 26 - POCO4A - Exercícios propostos >
João Pedro de Paula e Lucas dos Reis Viana:<
 Cópia de Aula 26 - POCO4A - Exercícios propostos >.
Vitor Hugo Leite A. de Oliveira <  Aula 26 - Exercícios >
Fernando Rafael:  Cópia de Aula 26 - POCO4A - Exercícios propostos
Gabriel Reis, Alexandre Aparecido e Matheus Maeda <
 Cópia de Aula 26 - POCO4A - Exercícios propostos >
Wesley Zimmer <  Cópia de Aula 26 - POCO4A - Exercícios propostos >
Gustavo Nunes :  Cópia de Aula 26 - POCO4A - Exercícios propostos
Victor Ramos e Thiago Tieghi:  Cópia de Aula 26 - POCO4A - Exercícios propostos

1) Padrão de Projeto: Iterator.

Motivação: Percorrer objetos personalizados.

Passo 1: Definição dos métodos para iterar os elementos:

```
public interface Iterator {  
    public abstract boolean temProximo();  
    public abstract Object proximo();  
}
```

Passo 2: Definição do elemento que será percorrido:

```
public class MeulItem {  
    private String nome;  
    public MeulItem(String nome){  
        this.nome=nome;  
    }  
    public String toString(){  
        return this.nome;  
    }  
}
```

Passo 3: Implementar os métodos da interface para indicar como percorrer os elementos:

```
import java.util.ArrayList;  
public class ItemIterator implements Iterator {  
  
    private ArrayList<MeulItem> lista;  
    private int pos=0;  
  
    public ItemIterator(ArrayList<MeulItem> lista){  
        this.lista = lista;  
    }  
  
    public boolean temProximo() {  
        boolean result=false;  
        if ( pos < this.lista.size() )  
            result=true;  
        return result;  
    }  
  
    public Object proximo() {  
        MeulItem meulItem = this.lista.get(pos);  
        pos++;  
        return meulItem;  
    }  
}
```

Passo 4: Percorrer os elementos personalizados:

```
import java.util.ArrayList;
public class Principal {

    public Principal(){
        ArrayList<MeuItem> lista = new ArrayList<>();
        lista.add(new MeuItem("Abacate"));
        lista.add(new MeuItem("Morango"));

        ItemIterator item = new ItemIterator(lista);

        while ( item.temProximo() )
            System.out.println( (MeuItem)item.proximo() );
    }

    public static void main(String[] args) {

        new Principal();

    }
}
```

//Exemplo de Sala1

```
package aula26;

import java.util.ArrayList;
import java.util.Iterator;
public class Aula26 {

    public class Prefeitura {
        private String cidade;
        public Prefeitura(String cidade){
            this.cidade = cidade;
        }
        public String toString(){
            return this.cidade;
        }
    }

    public class Veiculo {
        private String tipo;
        public Veiculo(String tipo){
            this.tipo = tipo;
        }
    }
}
```

```

    public String toString(){
        return this.tipo;
    }
}

public Aula26(){

    ArrayList<Veiculo> lista = new ArrayList<>();
    lista.add(new Veiculo("TIPO1"));
    lista.add(new Veiculo("TIPO2"));
    Iterator iterator = lista.iterator();
    while( iterator.hasNext() ){

        System.out.println(iterator.next()); //Exibe noh

    }
}

public static void main(String[] args) {

    new Aula26();

}
}

```

//Exemplo de Sala: Iterator personalizado

```

package aula26;

import java.util.ArrayList;
import java.util.Iterator;
public class Aula26 {

    //Passo1: interface
    public interface Meulterator {
        public abstract boolean temProximo();
        public abstract Object proximo();
    }

    //Passo2: elemento (noh da lista)
    public class Prefeitura {
        private String nome;
    }
}

```

```

    public Prefeitura(String nome){
        this.nome=nome;
    }
    public String toString(){
        return this.nome;
    }
}
//Passo3: implementar a consulta
public class PrefeituraNoh implements Meulterator {

    private ArrayList<Prefeitura> lista;
    private int pos=0;

    public PrefeituraNoh(ArrayList<Prefeitura> lista){
        this.lista = lista;
    }

    public boolean temProximo() {
        boolean result=false;
        if ( pos < this.lista.size() )
            result=true;
        return result;
    }

    public Object proximo() {
        Prefeitura meultem = this.lista.get(pos);
        pos++;
        return meultem;
    }
}

public Aula26(){

    ArrayList<Prefeitura> lista = new ArrayList<>();
    lista.add(new Prefeitura("CIDADE1"));
    lista.add(new Prefeitura("CIDADE2"));

    PrefeituraNoh iterator = new PrefeituraNoh(lista);
    while( iterator.temProximo() )
        System.out.print(iterator.proximo());

}

public static void main(String[] args) {

    new Aula26();
}

```

```

    }

}

```

- 2) (Online) Exercício: Percorrer o item da lista a partir do primeiro. Acesse o link da atividade (Aula26prog1): <https://codeboard.io/projects/359820>

```

//Julio Farias

import java.util.ArrayList;

public class Principal {
    ArrayList<MaterialEscolar> lista;
    public void iniciar(){
        lista = new ArrayList<>();
        lista.add(new MaterialEscolar("Lapis"));
        lista.add(new MaterialEscolar("Borracha"));

        Iterator i = new ItemIterator( lista );
        while( i.temProximo() )
            System.out.println( (MaterialEscolar) i.proximo() );
    }

    public static void main(String[] args) {
        Principal p = new Principal();
        p.iniciar();
    }
}

-----

public class MaterialEscolar{
    private String nome;
    public MaterialEscolar(String nome){
        this.nome = nome;
        //TODO 2
    }
    public String toString(){
        return this.nome;
    }
}

```

```

}
-----

public interface Iterator{
    public boolean temProximo();
    public Object proximo();
}
-----

import java.util.ArrayList;
public class ItemIterator implements Iterator {

    private ArrayList<MaterialEscolar> lista;
    private int pos=0;

    public ItemIterator(ArrayList<MaterialEscolar> lista){
        this.lista = lista;
    }

    public boolean temProximo() {
        boolean result=false;
        if ( pos < this.lista.size() )
            result=true;
        return result;
    }

    public Object proximo() {
        Object item = this.lista.get(pos);
        pos++;
        return item;
    }
}

public interface Iterator {
    public abstract boolean temProximo();
    public abstract Object proximo();
}

import java.util.ArrayList;
public class ItemIterator implements Iterator {

    private ArrayList<MaterialEscolar> lista;
    private int pos=0;

```

```

public ItemIterator(ArrayList<MaterialEscolar> lista){
    this.lista = lista;
}

public boolean temProximo() {
    boolean result=false;
    if ( pos < this.lista.size() )
        result=true;
    return result;
}

public Object proximo() {
    Object item = this.lista.get(pos);
    pos++;
    return item;
}
}

```

- 3) (Online) Exercício: Percorrer o item da lista a partir do último. Acesse o link da atividade (Aula26prog2): <https://codeboard.io/projects/359825>

- 4) Padrão de Projeto: Adapter.

Motivação: redefinição de métodos mantendo as mesmas assinaturas.

```

public class Principal2s2022 {

    public interface IFrutas {
        public void operacao1();
    }

    public class Abacate implements IFrutas{
        public void operacao1(){
            System.out.println("ABACATE");
        }
    }

    public class Maca implements IFrutas{
        public void operacao1(){
            System.out.println("MACA");
        }
    }
}

```



```

        }
    }

    public void iniciar(){

        IFrutas fruta = new Abacate();
        fruta.operacao1();

        fruta = new Maca();
        fruta.operacao1();

    }

    public static void main(String [ ] args){
        new Principal2s2022().iniciar();
    }

} //fim classe

```

//Exemplo de Sala 3: Padrão de Projeto Adapter

```

package aula26;

import java.util.ArrayList;
import java.util.Iterator;
public class Aula26 {

    //RELEASE

    //DEBUG

    public interface ITipo {
        public abstract void tipo();
    }

    public class Release implements ITipo {

        public void tipo() {
            System.out.println("RELEASE");
        }
    }

    public class Debug implements ITipo {

        public void tipo() {

```

```

        System.out.println("DEBUG");
    }
}

public Aula26(){

    ITipo sistema = new Release();
    sistema.tipo();

    sistema = new Debug();
    sistema.tipo();

}

public static void main(String[] args) {

    new Aula26();

}

}

```

- 5) (Online) Exercício: Acesse o link da atividade (Aula26prog3): <https://codeboard.io/projects/359827>

//Proposta de solução:

```

/**
 * TODO1: Classe Principal: implemente a interface 'IMaterialEscolar'
 *        com os metodo abstratos 'setPreco(float)' e 'setFornecedor(String)'.
 *
 * TODO2: Classe MaterialEscolar: adicione um construtor sobrecarregado
 *        que receba o nome, o preco, e o fornecedor.
 *
 * TODO3: Classe MaterialEscolar: implemente a interface 'IMaterialEscolar'.
 *
 * TODO4: Classe Principal: No metodo iniciar, imprima o nome, o preco e o
 *        fornecedor de cada item da lista, a partir do ultimo item.
 */

import java.util.ArrayList;

public class Principal {

```

```

public interface IMaterialEscolar{
    public abstract void setPreco(float preco);
    public abstract void setFornecedor(String fornecedor);
}

public class MaterialEscolar implements IMaterialEscolar{
    private String nome;
    private float preco;
    private String fornecedor;

    public MaterialEscolar(String nome){
        this.nome=nome;
        this.preco=0.0f;
        this.fornecedor="fornecedor";
    }
    public MaterialEscolar(String nome, float preco, String fornecedor){
        this.nome=nome;
        setPreco(preco);
        setFornecedor(fornecedor);
    }
    public void setPreco(float preco){
        this.preco = preco;
    }
    public void setFornecedor(String fornecedor){
        this.fornecedor = fornecedor;
    }
    public String toString(){
        return this.nome+
            " preco: " + this.preco +
            " fornecedor: " + this.fornecedor;
    }
}

ArrayList<MaterialEscolar> lista;

public interface Iterator {
    public boolean temAnterior();
    public Object anterior();
}

public class ItemIterator implements Iterator {

    private ArrayList< MaterialEscolar > lista;
    private int pos=0;

    public ItemIterator(ArrayList<MaterialEscolar> lista){

```

```

        this.lista = lista;

        pos=this.lista.size()-1;
    }

    public boolean temAnterior() {
        boolean result=false;
        if ( pos >= 0 )
            result=true;
        return result;
    }

    public Object anterior() {
        MaterialEscolar item = this.lista.get(pos);
        pos--;
        return item;
    }

    public boolean temProximo() {
        boolean result=false;
        if ( pos < this.lista.size() )
            result=true;
        return result;
    }

    public Object proximo() {
        MaterialEscolar item = this.lista.get(pos);
        pos++;
        return item;
    }
}

public void iniciar(){
    lista = new ArrayList<>();
    lista.add(new MaterialEscolar("Lapis",10f,"FORNECEDOR1"));
    lista.add(new MaterialEscolar("Borracha",5f,"FORNECEDOR2"));

    Iterator i = new ItemIterator( lista );
    while( i.temAnterior() )
        System.out.println( (MaterialEscolar) i.anterior() );
}

public static void main(String[] args) {
    Principal p = new Principal();
    p.iniciar();
}
}

```



6) (Online) Exercício: Acesse o link da atividade (Aula26prog4): <https://codeboard.io/projects/359834>

7) Padrão de Projeto: Singleton.

Motivação: manter uma única instância ativa de um objeto de determinada classe.

Passo 1: Criar classe com construtor private e variável de classe private.

```
public class Prateleira {  
  
    private static Prateleira estoque;  
  
    private Prateleira () {  
        System.out.println("Singleton iniciado.");  
    }  
  
    public static Prateleira iniciar(){  
        if(estoque==null)  
            estoque = new Prateleira();  
        return estoque;  
    }  
  
}
```

Passo 2: Instanciar o singleton em outra classe.

```
public class Principal {  
  
    public void iniciar(){  
  
        Prateleira adm = Prateleira.iniciar();  
        Prateleira adm2 = Prateleira.iniciar();  
  
    }  
  
    public static void main(String[] args) {  
        new Principal().iniciar();  
    }  
}
```

```

    }
}

```

```

//Exemplo de Sala

package aula26;

public class Aula26 {

    public Aula26(){
        Janela janela1 = Janela.iniciar();
        janela1.terminar();

        Janela janela2 = Janela.iniciar();

    }

    public static void main(String[] args) {

        new Aula26();

    }
}

//=====

package aula26;

public class Janela {

    //Passo 2
    public static Janela janela;

    //Passo1
    private Janela(){
        System.out.println("JANELA CRIADA");
    }

    //Passo3
    public static Janela iniciar(){

```

```
        if(janela==null)
            janela = new Janela();

        return janela;
    }

    public static void terminar(){
        janela=null;
    }

    public static void main(String[] args) {

        new Janela();

    }
}
```

- 8) (Online) Exercício: Acesse o link da atividade (Aula25prog5):
<https://codeboard.io/projects/359841>
- 9) (Online) Exercício: Acesse o link da atividade (Aula25prog6):
<https://codeboard.io/projects/359856>