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

—----------TURMA B—----------
Nome e link:
Andrei Fernandes Zani: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Fernando Rafael: Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Rafael Zaupa Watanabe: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Alexandre Calisto: Aula 26
Iago Macarini: Aula 26 - POCO4A

Vinicius Letroche Felix: Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
—----------TURMA A—----------
Nome e link:
Vitor Batista Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Gabriel Candelária: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Camila Costa: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Matheus H. A. Pereira: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Lucas Ribeiro P. Maroja: Aula 26 - POCO4A - 17/11/2022
Erik Noda: aula26
Lucas Prado: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Michael Pariz: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Gustavo Nunes: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Arnald Souza: Aula 26 - 17/11
Leonardo G. Fagote Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Maria Eduarda: Pedroso - Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Raphael Uematsu: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Mariana Pedroso Naves: Aula 26 - POO
Guilherme Conceição Ramalho:
Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Ruan Mateus Trizotti: Aula 26
Felipe Antonio Magro: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
William Eizo Hatakeyama:
Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Julio Farias: Cópia de Aula 26 - POCO4A - 17/11/2022 - Exercícios propostos
Lucas Santana: Aula 26 - POCO4A - 17/11/2022 - Lucas Santana

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 MeuItem {
	private String nome;
	public MeuItem(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<MeuItem> lista;
	private int pos=0;

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

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

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

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();
	}
}
```

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

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
```

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

```
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;
   }
   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 + " " +
            this.preco + " " +
            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", 1.5f, "JOAO"));
      lista.add(new MaterialEscolar("Borracha", 3.5f, "MARIA"));

      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();

	}
}
```

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