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

Nome: Link:

Filipe Augusto Parreira < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Marcos Shoji< Cópia de Aula 25 - POCO4A - Exercícios propostos >
Mabille < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Matheus e Thiago< Cópia de Aula 25 - POCO4A - Exercícios propostos >
Plinio < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Julio Farias< Cópia de Aula 25 - POCO4A - Exercícios propostos >
Angélica < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Thales Hasegawa, Wesley Zimmer < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Vitor Viana < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Raphael Uematsu < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Deivid da Silva Galvão < Cópia de Aula 25 - POCO4A - Exercícios propostos >
João Vitor N. Yoshida < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Rafael Kendy < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Séfora< Aula 25 - POO >
Fernando Rafael: Cópia de Aula 25 - POCO4A - Exercícios propostos
Gabriel Reis < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Roberto Furlani Neto < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Daniel Martins de Carvalho < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Henrique Cois < Cópia de Aula 25 - POCO4A - Exercícios propostos >
João Pedro Cavani Meireles< Cópia de Aula 25 - POCO4A - Exercícios propostos >
Felipe Lorusso, Carlos Eduardo e Luis Mendes <Aula 25>
Guilherme Conceição Ramalho < Cópia de Aula 25 - POCO4A - Exercícios propostos >
Vitor Hugo Leite A. de Oliveira < Aula 25 - Exercícios >
Gabriel Takeshi < Cópia de Aula 25 - POCO4A - Exercícios propostos >

1) Padrão de Projeto: Facade (Fachada).

   Motivação: Simplificar a chamada de vários métodos.

```
public class Principal {

    public abstract class Facade {
        public abstract void iniciar();
    }

    public class Supermercado extends Facade {
```

```
        List <Integer>  listaCaixas;
        List <Integer>  listaVendas;
        List <Integer>  listaDespesas;

        public void iniciar(){

            iniciarCaixas();
            iniciarVendas();
            iniciarDespesas();
        }//fim iniciar

        private void iniciarCaixas(){   this.listaCaixas = new ArrayList<>();  }
        private void iniciarVendas(){   this.listaVendas = new ArrayList<>();  }
        private void iniciarDespesas(){   this.listaDespesas = new ArrayList<>();  }

    }//fim classe

    public Principal(){

        Facade supermercado = new Supermercado();
        supermercado.iniciar();
        Facade comercioEletronico = new ComercioEletronico();
        comercioEletronico.iniciar();

    }

    public static void main(String [] args){
          new Principal();

    }
}

—---------
Exemplo 2:

package aula25;

public class Aula25 {

   public abstract class Servico {
      public abstract void tipoServico();

      public void iniciar(){
         tipoServico();
      }

   }
```

```
  public class Prefeitura extends Servico {

     public void tipoServico(){
        System.out.println("SERVICO DA PREFEITURA");
     }

  }
  public class SetorCompras extends Servico {
     public void tipoServico(){
        System.out.println("SERVICO DA COMPRAS");
     }

  }

  public Aula25(){

     iniciar();  //generico

  }
  public void iniciar(){

     Servico prefeitura = new Prefeitura();
     prefeitura.iniciar(); //especifico

     Servico setorCompras = new SetorCompras();
     setorCompras.iniciar(); //especifico

  }


  public static void main(String[] args) {

     /*Servico prefeitura = new Aula25().new Prefeitura();

     Servico setorCompras = new Aula25().new SetorCompras();

     prefeitura.tipoServico();
     setorCompras.tipoServico();*/

     new Aula25();

  }

}
```

2) (Online) Exercício: Acesse o link da atividade (Aula25prog1): https://codeboard.io/projects/357605

3) Padrão de Projeto: Delegação.

Motivação: o objeto poderá mudar de classe com o uso de polimorfismo, ou seja, o objeto da subclasse é tratado como um objeto de sua superclasse através de sobrescrita de métodos. Delegação é implementada pela composição de objetos.

```
public class Principal {
    public abstract class Fruta {
          public abstract String toString();
    }
    public class Maca extends Fruta {
        public String toString(){   return this.getClass().getSimpleName(); }
    }
    public class Abacate extends Fruta {
        public String toString(){   return this.getClass().getSimpleName(); }
    }

    public class Prateleira {
        private Fruta fruta;
        public Prateleira(Fruta fruta) {
              setFruta(fruta);
        }
        public void setFruta(Fruta fruta) {
             this.fruta = fruta;
        }
        public String toString(){ return this.fruta.toString(); }
    }


    public Principal(){

         Prateleira  prateleira = new Prateleira( new Maca() );
         System.out.println( prateleira );

        Prateleira prateleira2 = new Prateleira( new Abacate() );
        System.out.println( prateleira2 );

        prateleira.setFruta( new Abacate() );
        System.out.println( prateleira );
    }


    public static void main(String [] args ){
```

```
        new Principal();
    }

}


//—------------------
Exemplo 2:



package aula25;

public class Aula25 {

   public abstract class Funcionario {
      public abstract void pagamento();
   }

   public class Bolsista extends Funcionario {
      public void pagamento(){
         System.out.println("100");
      }
   }
   public class Estagiario extends Funcionario {
      public void pagamento(){
         System.out.println("10");
      }
   }
   public class Prefeitura {

      public void pagamento( Funcionario funcionario ){

         funcionario.pagamento();
      }

   }


   public Aula25(){

      Prefeitura prefeitura = new Prefeitura();

      Funcionario joao = new Bolsista();
      prefeitura.pagamento(joao);

      joao = new Estagiario();
```

```
        prefeitura.pagamento(joao);


    }


    public static void main(String[] args) {
        new Aula25();
    }



}
```

4) (Online) Exercício: Acesse o link da atividade (Aula25prog2): https://codeboard.io/projects/357606

```
///mabylly
import java.util.List;
import java.util.ArrayList;
public class Principal {

    public class SistemaPapelaria{
        List<MaterialEscolar> lm;
        List<Caixa> lc;
        List<Empregado> le;

        public void iniciar(){
            lm = new ArrayList<>();
            lc = new ArrayList<>();
            le = new ArrayList<>();

            MaterialEscolar materialEscolar = new Lapis();
            lm.add(materialEscolar);
            material.marca();
            material.tamanho();

            materialEscolar = new Borracha();
            lm.add(materialEscolar);
            material.marca();
            material.tamanho();
        }
    }
```

```
public abstract class MaterialEscolar{
    public abstract void marca();
    public abstract void tamanho();

}
public class Lapis extends MaterialEscolar{
   public void marca(){ System.out.println("Faber");}
   public void tamanho(){ System.out.println("8cm");}
}
public class Borracha extends MaterialEscolar{
   public void marca(){ System.out.println("Faber");}
   public void tamanho(){ System.out.println("5cm");}
}
```

5) Padrão de Projeto: Interface com Delegação.

Motivação: o objeto poderá mudar de classe com o uso de polimorfismo, ou seja, o objeto da subclasse é tratado como um objeto de sua superclasse. Interface com Delegação é implementada pela composição com uma interface.

```
public class Principal {
    public interface IFruta {
          public abstract String toString();
    }
    public class Maca implements IFruta {
        public String toString(){   return this.getClass().getSimpleName(); }
    }
    public class Abacate implements IFruta {
        public String toString(){   return this.getClass().getSimpleName(); }
    }

    public class Prateleira {
        private IFruta interfaceFruta;
        public Prateleira(IFruta interfaceFruta) {
              setFruta(interfaceFruta);
        }
        public void setFruta(IFruta interfaceFruta) {
```

```
            this.interfaceFruta = interfaceFruta;
        }
        public String toString(){ return this.interfaceFruta.toString(); }
    }


    public Principal(){

         Prateleira  prateleira = new Prateleira( new Maca() );
         System.out.println( prateleira );

        Prateleira prateleira2 = new Prateleira( new Abacate() );
        System.out.println( prateleira2 );

        prateleira.setFruta( new Abacate() );
        System.out.println( prateleira );
    }


    public static void main(String [] args ){

        new Principal();
   }

}
//—----------
```

Exemplo 2:

```
package aula25;

public class Aula25 {

   public interface Funcionario {
      public abstract void pagamento();
   }

   public class Bolsista implements Funcionario {
      public void pagamento(){
         System.out.println("100");
      }

   }
   public class Estagiario implements Funcionario {
      public void pagamento(){
         System.out.println("10");
      }
```

```
  }
  public class Prefeitura {

     public void pagamento( Funcionario funcionario ){

        funcionario.pagamento();
     }

  }


  public Aula25(){

     Prefeitura prefeitura = new Prefeitura();

     Funcionario joao = new Bolsista();
     prefeitura.pagamento(joao);

     joao = new Estagiario();
     prefeitura.pagamento(joao);


  }


  public static void main(String[] args) {
     new Aula25();
  }



}
```

6) (Online) Exercício: Acesse o link da atividade (Aula25prog3): https://codeboard.io/projects/357610

```
//Marcos


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

   public class SistemaPapelaria{
      List<MaterialEscolar> lm;
      List<Caixa> lc;
      List<Empregado> le;

      public void iniciar(){
         lm = new ArrayList<>();
         lc = new ArrayList<>();
         le = new ArrayList<>();

         MaterialEscolar materialEscolar = new Lapis();
         lm.add( materialEscolar );
         materialEscolar.marca();
         materialEscolar.tamanho();

         materialEscolar = new Borracha();
         lm.add( materialEscolar );
         materialEscolar.marca();
         materialEscolar.tamanho();
      }
   }
   public class Lapis implements MaterialEscolar {
      public void marca(){ System.out.println("FABER"); }
      public void tamanho(){ System.out.println("10cm"); }
   }
   public class Borracha implements MaterialEscolar {
      public void marca(){ System.out.println("RUBBER"); }
      public void tamanho(){ System.out.println("5cm"); }
   }
   public interface MaterialEscolar {
      public abstract void marca();
      public abstract void tamanho();
   }
   public class Caixa{
   }
   public class Empregado{
   }

   public void iniciar(){
      SistemaPapelaria sp = new SistemaPapelaria();
      sp.iniciar();
```

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

//Exemplo 5.1: Padrão de Projeto: Interface com Delegação (com classes não-relacionadas). Relacionar: Bicicleta, Carro, Predio

```
public class Principal {

    public interface IEmissaoCarbono {
        public abstract float getEmissaoCarbono();
    }

    public class Bicicleta implements IEmissaoCarbono {

        public float getEmissaoCarbono(){
```

```
            return 0.0f;
        }
    }
     public class Carro implements IEmissaoCarbono {

        public float getEmissaoCarbono(){
            return 50.0f;
        }
    }
   public class Predio implements IEmissaoCarbono {

        public float getEmissaoCarbono(){
            return 10.0f;
        }
    }

     public class Poluente {
          private IEmissaoCarbono  poluente;
          public Poluente(IEmissaoCarbono  poluente){
               setPoluente(poluente);
          }
          public void setPoluente(IEmissaoCarbono poluente){
                this.poluente = poluente;
          }
          public String toString(){
                 return this.poluente.getEmissaoCarbono();
           }
     }

     public Principal(){
             Poluente  poluente = new Poluente( new Carro() );
             System.out.println( poluente );

             poluente.setPoluente( new Bicicleta() );
             System.out.println( poluente );

     }

    public static void main(String [] args){
        new Principal();

    }
}

//—------
Exemplo 2:
```

```
package aula25;

public class Aula25 {

   public interface Projeto {

      public void imprimirNome();
      public void imprimirPagamento();

   }

   //----Codigo do GRUPO1
   public class Grupo1 implements Projeto {

      @Override
      public void imprimirNome() {
         System.out.println("GRUPO1");
      }

      @Override
      public void imprimirPagamento() {
         System.out.println("100");
      }

   }

   //----Codigo do GRUPO2
   public class Grupo2 implements Projeto {

            @Override
      public void imprimirNome() {
         System.out.println("GRUPO2");
      }

      @Override
      public void imprimirPagamento() {
         System.out.println("10");
      }
   }


   public void imprimir(Projeto projeto){
       projeto.imprimirNome(); //GRUPO1
       projeto.imprimirPagamento(); //100
   }

   public Aula25(){
```

```
        Projeto projeto = new Grupo1();

        imprimir(projeto);

        projeto = new Grupo2();

        imprimir(projeto);

    }


    public static void main(String[] args) {
        new Aula25();
    }



}
```

7) (Online) Exercício: Acesse o link da atividade (Aula25prog4): https://codeboard.io/projects/357611

```
//
/**
 * TODO 1: Principal: Crie a interface 'Produto' com o metodo 'fornecedor'
 * TODO 2: Lapis: implemente a interface 'Produto' para imprimir o
 *         nome do fornecedor.
 * TODO 3: SistemaPapelaria: imprima o nome do fornecedor no metodo 'iniciar'
 * TODO 4: Repita os passos 2 e 3 para a classe 'Borracha'
 */

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

    public interface Produto {

        public abstract void fornecedor();
    }
```

```
public class SistemaPapelaria{
   List<MaterialEscolar> lm;
   List<Caixa> lc;
   List<Empregado> le;

   public void iniciar(){
      lm = new ArrayList<>();
      lc = new ArrayList<>();
      le = new ArrayList<>();

      MaterialEscolar materialEscolar = new Lapis();
      lm.add( materialEscolar );
      materialEscolar.marca();
      materialEscolar.tamanho();
      //TODO3
      ((Produto)materialEscolar).fornecedor();

      materialEscolar = new Borracha();
      lm.add( materialEscolar );
      materialEscolar.marca();
      materialEscolar.tamanho();
      //TODO3
      ((Produto)materialEscolar).fornecedor();
   }
}
//TODO2
public class Lapis implements MaterialEscolar, Produto {
   public void marca(){ System.out.println("FABER"); }
   public void tamanho(){ System.out.println("10cm"); }
   public void fornecedor(){ System.out.println("FORNECEDOR DO LAPIS"); }
}
public class Borracha implements MaterialEscolar, Produto {
   public void marca(){ System.out.println("RUBBER"); }
   public void tamanho(){ System.out.println("5cm"); }
   public void fornecedor(){ System.out.println("FORNECEDOR DA BORRACHA"); }
}
public interface MaterialEscolar {
   public abstract void marca();
   public abstract void tamanho();
}
public class Caixa{
}
public class Empregado{
}

public void iniciar(){
   SistemaPapelaria sp = new SistemaPapelaria();
```

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

8) (Online) Exercício: Acesse o link da atividade (Aula25prog5): https://codeboard.io/projects/357613