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

Nome: Link:

Roberto Furlani Neto < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Rafael Kendy Naramoto Lopes < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Angélica < Cópia de Aula 16 - POCO4A - Exercícios propostos >
mabylly< Cópia de Aula 16 - POCO4A - Exercícios propostos >
Séfora< Aula 16 - POO >
Daniel Martins de Carvalho < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Matheus Mazali Maeda < Copia >
Marcos Tadao Shoji< Cópia de Aula 16 - POCO4A - Exercícios propostos >
Isabella < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Thiago Cristovão de Souza < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Henrique Cois < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Guilherme Ramalho < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Filipe Augusto Parreira Almeida < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Gabriel Takeshi Abe< Cópia de Aula 16 - POCO4A - Exercícios propostos >
Matheus Hirata < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Plinio Koiama e João Pedro Cavani Meireles <
Cópia de Aula 16 - POCO4A - Exercícios propostos >
Vitor Luiz de Castro Viana < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Alexandre Aparecido < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Deivid da Silva Galvao < Cópia de Aula 16 - POCO4A - Exercícios propostos >
João Vitor N. Yoshida < Cópia de Aula 16 - POCO4A - Exercícios propostos >
João Pedro de Paula:< Cópia de Aula 16 - POCO4A - Exercícios propostos >
Julio Farias< Cópia de Aula 16 - POCO4A - Exercícios propostos >
Vitor Hugo Leite A. de Oliveira < Aula 16 - Exercícios propostos >
Wesley Zimmer, Thales Hasegawa < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Felipe Antonio Magro: Cópia de Aula 16 - POCO4A - Exercícios propostos
Rodrigo Leandro Benedito: Cópia de Aula 16 - POCO4A - Exercícios propostos
Thiago Tieghi: < Cópia de Aula 16 - POCO4A - Exercícios propostos >
Victor R
Lucas Viana, Carlos Eduardo e Luiz Henrique: <
Cópia de Aula 16 - POCO4A - Exercícios propostos >

Parte 1

Exercícios propostos:

1) (Online) Acesse o link da atividade online e realize as tarefas propostas: https://codeboard.io/projects/347925

```
//mabylly
/**
 * TODO 1: No metodo iniciar, defina um objeto do tipo MembroUniversitario que seja instanciado
 *         com um tipo Bolsista.
 * TODO 2: Imprima o salario do objeto do item anterior com o metodo 'toString()'.
 * TODO 3: Atribua um tipo Estudante para o mesmo objeto do item anterior.
 * TODO 4: Imprima o salario do objeto do item anterior com o metodo 'toString()'.
 */
public class Principal {

   public abstract class MembroUniversitario {
      protected float salario;
      public abstract float getSalario();
      public MembroUniversitario(float salario){
         this.salario=salario;
      }
      public String toString(){
         return this.getClass().getSimpleName() + " - Salario: " + this.salario;
      }
   }

   //Bolsista eh subclasse de MembroUniversitario
   public class Bolsista extends MembroUniversitario {
      public Bolsista(float salario){
         super(salario);
      }
      public float getSalario(){
         return this.salario;
      }
   }

   //Estudante eh subclasse de MembroUniversitario
   public class Estudante extends MembroUniversitario {
      public Estudante(){
         super(0.0f);
      }
      public float getSalario(){
         return this.salario;
      }
   }

   public void iniciar(){
```

```
   //TODO 1
   Principal.MembroUniversitario mu1 = new Principal.Bolsista(400.00f);


      //TODO 2
      System.out.println(  mu1.toString());

      //TODO 3
      mu1 = new Estudante();

      //TODO 4
     System.out.println(  mu1.toString());

   }

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

2) (Online) Acesse o link da atividade online e realize as tarefas propostas: https://codeboard.io/projects/347928

```
//Roberto
/**
 * TODO 1: No metodo iniciar, crie uma lista dinamica de objetos
 *         do tipo MembroUniversitario.
 * TODO 2: Defina um objeto do tipo MembroUniversitario que seja instanciado
com
 *         um tipo Bolsista.
 * TODO 3: Adicione ah lista o objeto do item anterior.
 * TODO 4: Atribua um tipo Estudante para o mesmo objeto do TODO 2.
 * TODO 5: Adicione ah lista o objeto do item anterior.
 * TODO 6: Imprima o salario de cada objeto da lista com o metodo polimorfico
'toString()'
 */
import java.util.List;
import java.util.ArrayList;
```

```
public class Principal {

   public interface MembroUniversitario {
      public abstract float getSalario();
      public abstract String toString();
   }

   public class Bolsista implements MembroUniversitario {
         private float salario;
      public Bolsista(float salario){
         this.salario=salario;
      }
      public float getSalario(){
         return this.salario;
      }
      public String toString(){
          return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
      }
   }
   public class Estudante implements MembroUniversitario {
         private float salario;
      public Estudante(){
          this.salario=0.0f;
      }
      public float getSalario(){
         return this.salario;
      }
      public String toString(){
          return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
      }
   }

   public void iniciar(){

      ArrayList<MembroUniversitario> lista = new ArrayList<>();


      MembroUniversitario membro1 = new Bolsista(33.0f);


      lista.add(membro1);


      membro1 = new Estudante();


      lista.add(membro1);
```

```
        for( MembroUniversitario membro : lista)
            System.out.println(membro.toString());



    }

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

3) (Online) Acesse o link da atividade online e realize as tarefas propostas: https://codeboard.io/projects/347930

```
//Marcos Shoji
/**
 * TODO 1: Classe Principal: crie um metodo publico
 *         'void adicionar(MembroUniversitario membro)'.
 * TODO 2: Classe Principal: crie uma variavel de instancia 'lista'
 *         que seja uma lista dinamica do tipo MembroUniversitario.
 * TODO 3: Classe Principal: inicialize a lista dinamica do item anterior
 *         no construtor padrao sem argumentos.
 * TODO 4: No metodo iniciar, defina um objeto do tipo MembroUniversitario que
 *         seja instanciado com um tipo Bolsista.
 * TODO 5: Adicione o objeto ah lista, com o metodo do TODO 1.
 * TODO 6: Atribua um tipo Estudante para o mesmo objeto do TODO 4.
 * TODO 7: Adicione o objeto ah lista, com o metodo do TODO 1.
 * TODO 8: No metodo iniciar, invoque o metodo 'imprimir'.
 */
import java.util.List;
import java.util.ArrayList;
public class Principal {

    ArrayList<MembroUniversitario> lista;//TODO 2

    public Principal(){
        lista = new ArrayList<>();//TODO 3
    }
    public interface MembroUniversitario {
        public abstract float getSalario();
```

```
    public abstract String toString();
}

public class Bolsista implements MembroUniversitario {
        private float salario;
    public Bolsista(float salario){
       this.salario=salario;
    }
    public float getSalario(){
       return this.salario;
    }
    public String toString(){
        return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
    }
}
public class Estudante implements MembroUniversitario {
        private float salario;
    public Estudante(){
        this.salario=0.0f;
    }
    public float getSalario(){
       return this.salario;
    }
    public String toString(){
        return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
    }
}

public void adicionar(MembroUniversitario membro){

                lista.add(membro);//TODO 5

} //TODO 1



public void imprimir(){
    for(MembroUniversitario membro : lista)
      System.out.println( membro );
}

public void iniciar(){

    Principal.MembroUniversitario mu1 = new Principal.Bolsista(500.0f);
    //TODO 4

    adicionar( mu1 );
```

```
        mu1 = new Principal.Estudante();//TODO 6

        adicionar( mu1 );

        imprimir();

        //for( MembroUniversitario membro : lista)
        //    System.out.println(mu1.toString());//to do 8
    }

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

4) (Online) Acesse o link da atividade online e realize as tarefas propostas: https://codeboard.io/projects/347933

```
//CARLOS EDUARDO E LUCAS VIANA

/**
 * TODO 1: Classe Principal: crie uma nova classe interna 'Tecnico' que
 *         implemente a interface 'MembroUniversitario'
 * TODO 2: Metodo iniciar: Atribua ao objeto 'joao' o tipo 'Tecnico' com
polimorfismo.
 * TODO 3: Metodo iniciar: Invoque o metodo 'imprimir'
 */
import java.util.List;
import java.util.ArrayList;
public class Principal {

    private List<MembroUniversitario> lista;

    public Principal(){
        lista = new ArrayList<>();
    }

    public interface MembroUniversitario {
        public abstract float getSalario();
        public abstract String toString();
    }
```

```
//TODO 1
public class Tecnico implements MembroUniversitario{
   private float salario;
   public Tecnico(float salario){
      this.salario=salario;
   }
   public float getSalario(){
      return this.salario;
   }
   public String toString(){
       return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
   }

}

public class Bolsista implements MembroUniversitario {
       private float salario;
   public Bolsista(float salario){
      this.salario=salario;
   }
   public float getSalario(){
      return this.salario;
   }
   public String toString(){
       return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
   }
}
public class Estudante implements MembroUniversitario {
       private float salario;
   public Estudante(){
       this.salario=0.0f;
   }
   public float getSalario(){
      return this.salario;
   }
   public String toString(){
       return this.getClass().getSimpleName() + "\tSalario: R$" + getSalario();
   }
}

public void adicionar(MembroUniversitario membro){
   lista.add( membro );
}

public void imprimir(){
   for(MembroUniversitario membro : lista)
     System.out.println( membro );
```

```
    }

    public void iniciar(){

        //
        lista = new ArrayList<>();

        //
        MembroUniversitario joao = new Bolsista(400.0f);

        //
        adicionar( joao );

        //
        joao = new Estudante();

        //
        adicionar( joao );

        //TODO 2
        joao = new Tecnico(20.00f);

        adicionar( joao );

        //TODO 3
        imprimir();

    }

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

5) (Online) Acesse o link da atividade online e realize as tarefas propostas: https://codeboard.interface/projects/347927

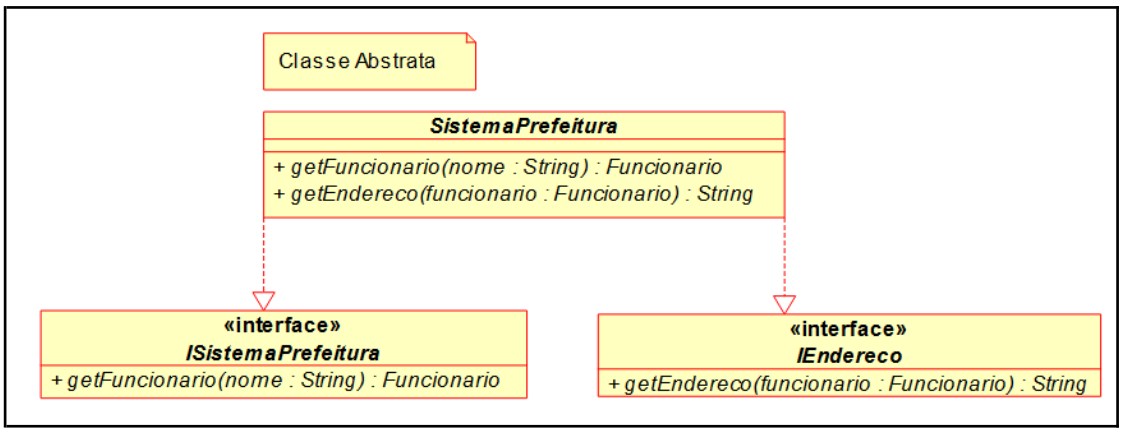
```
//Julio Farias
/**
 * TODO 1: Modifique a classe MembroUniversitario para que ela se torne uma interface.
 *       Faca as modificacoes necessarias nas demais classes.
```

```
 * TODO 2: Defina um objeto do tipo MembroUniversitario que seja instanciado
com
 *         um tipo Bolsista.
 * TODO 2: Imprima o salario do objeto do item anterior com o metodo 'toString()'.
 * TODO 3: Atribua um tipo Estudante para o mesmo objeto do item anterior.
 * TODO 4: Imprima o salario do objeto do item anterior com o metodo 'toString()'.
 */
public class Principal {

   //TODO 1
   public interface MembroUniversitario {
      mepublic abstract float getSalario();
      public abstract String toString();
   }

   public class Bolsista implements MembroUniversitario {
       private float salario;

      public Bolsista(float salario){
         //super(salario);
         this.salario = salario;
      }
      public float getSalario(){
         return this.salario;
      }
      public String toString() {
            return this.getClass().getSimpleName() + " salario: " + getSalario() );
      }


   }
   public class Estudante implements MembroUniversitario {
       private float salario;

      public Estudante(){
         //super(0.0f);
         this.salario = 0.0f;
      }
      public float getSalario(){
         return this.salario;
      }
      public String toString() {
            return this.getClass().getSimpleName() + " salario: " + getSalario() );
      }
   }

   public void iniciar(){
```

```
        //TODO 1


        //TODO 2
        MembroUniversitario membro = new Bolsista(500.f);
        System.out.println(membro.toString());

        //TODO 3
        membro = new Estudante();

        //TODO 4
        System.out.println(membro.toString());
    }

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

Parte 2

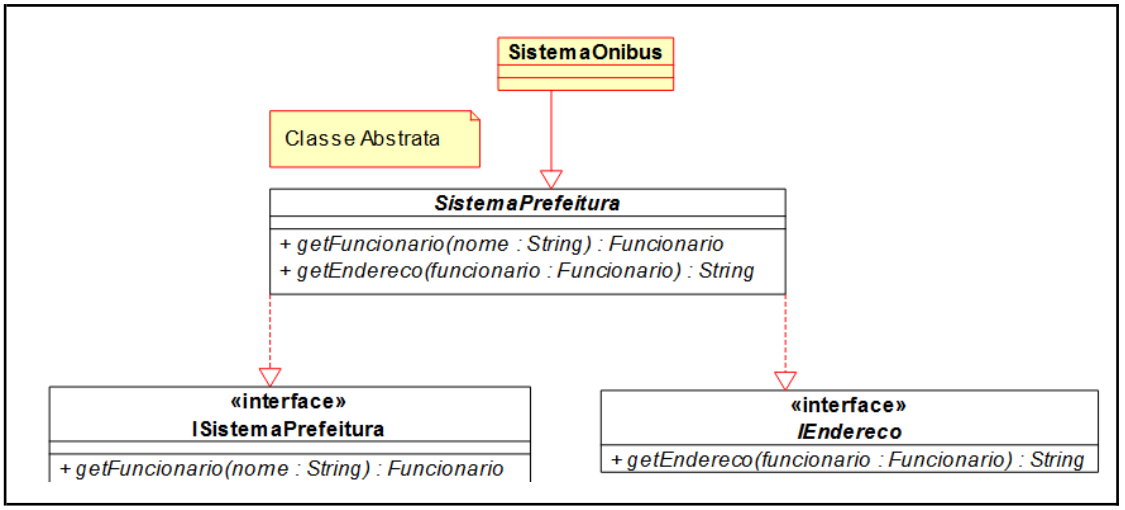
1) Observe o diagrama UML a seguir:



a) Implemente o código-fonte do diagrama. Nota: neste exemplo, a classe abstrata só possui métodos abstratos.

```
public abstract class SistemaPrefeitura
        implements ISistemaPrefeitura, IEndereco {

      public abstract Funcionario getFuncionario( String nome );
      public abstract String getEndereco( Funcionario funcionario );

}

public interface ISistemaPrefeitura {
     public abstract Funcionario getFuncionario( String nome );
}

public interface IEndereco {
     public abstract String getEndereco( Funcionario funcionario );
}
```

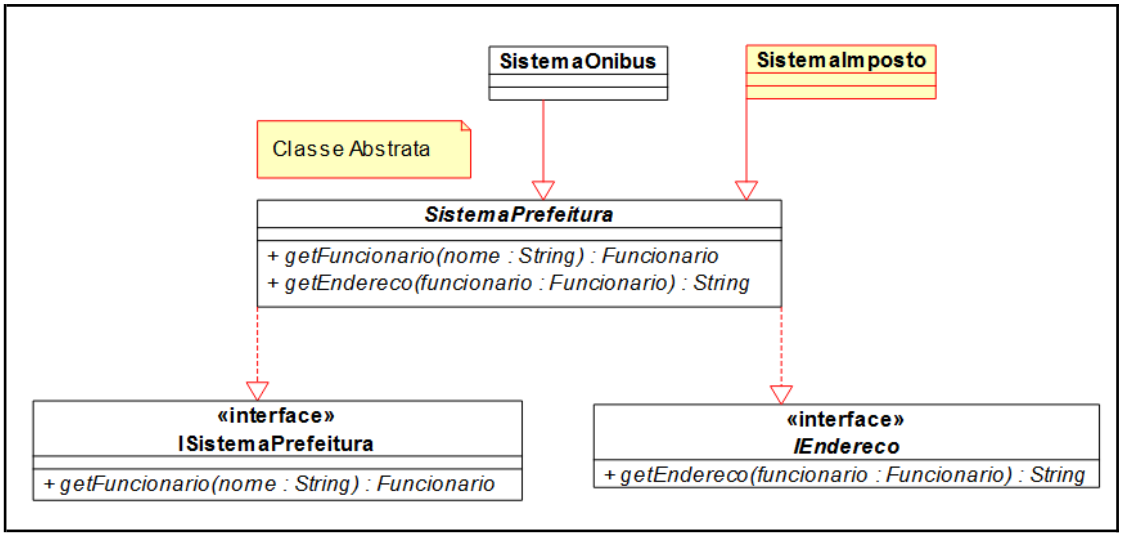
2) Observe o diagrama UML a seguir:



a) Implemente o código-fonte do diagrama. Nota: neste exemplo, as classes derivadas da classe abstrata só possuem métodos concretos.

```
public class SistemaOnibus extends SistemaPrefeitura {

        public Funcionario getFuncionario(String nome) {
            //TODO
        }
        public String getEndereco( Funcionario funcionario ) {
            //TODO
        }

}
```

3) Observe o diagrama UML a seguir:



a) Implemente o código-fonte do diagrama. Nota: neste exemplo, as classes derivadas da classe abstrata só possuem métodos concretos.

```
public class SistemaImposto extends SistemaPrefeitura {

        public Funcionario getFuncionario(String nome) {
                //TODO
            }
            public String getEndereco( Funcionario funcionario ) {
                //TODO
            }

}
```

4) Ilustre um exemplo funcional de polimorfismo com uma lista de alocação dinâmica a partir da implementação do diagrama do item anterior.

```
<Insira o seu código-fonte aqui>
```

5) Ilustre um exemplo funcional com polimorfismo que utilize classe interna anônima para definir uma nova classe derivada da classe abstrata.

```
<Insira o seu código-fonte aqui>
```

6) Explique: como as classes abstratas e a sobrescrita de métodos contribuem para o polimorfismo?