



Compartilhar o seu link com: [luciorocha @ professores.utfpr.edu.br](mailto:luciorocha@professores.utfpr.edu.br)

Arnald: [☰ Aula 16 - 04/10](#)

Matheus H. A. Pereira: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Plinio Koiana: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Mariana Naves: [☰ Aula 16 - POO - Exercícios propostos](#)

Michael Pariz Pereira: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Fernando Rafael: [☰ POCO4A - 04/10/2022 - Exercícios propostos](#)

Gustavo Naoki Jodai Kurozawa: [☰ Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Iago Macarini: [☰ POO - Aula 16](#)

Rafael Zaupa Watanabe: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Gustavo Nunes : [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Alexandre Calisto: [☰ Aula 16](#)

Lucas R. P. Maroja: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Raphael Uematsu: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Gabriel Takeshi Abe: [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Julio farias : [☰ Cópia de Aula 16 - POCO4A - 04/10/2022 - Exercícios propostos](#)

Parte 1

Exercícios propostos:

- 1) (Online) Acesse o link da atividade online e realize as tarefas propostas:

<https://codeboard.io/projects/347925>

```
/**
 * 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;
    }
}

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

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

public void iniciar(){

    //TODO 1
    MembroUniversitario bolsista = new Bolsista(1228.9f);

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

    //TODO 3
    MembroUniversitario aluno = new Estudante();

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

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

```

```
/**
```

```

* 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;
        }
    }

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

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

    public void iniciar(){

        //TODO 1
        MembroUniversitario bolsista = new Bolsista(1228.9f);

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

        //TODO 3
        bolsista = new Estudante();
    }
}

```

```

//TODO 4
System.out.println(bolsista.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/347927>

```

public class Principal {

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

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

    public class Estudante implements MembroUniversitario {
        protected float salario;
        public Estudante(){
            this.salario = 0;
        }
        public float getSalario(){
            return this.salario;
        }
        public String toString(){
            return this.getClass().getSimpleName() + " - Salario: " + this.salario;
        }
    }

    public void iniciar(){

```

```

//TODO 1
//feito

//TODO 2
MembroUniversitario obj = new Bolsista(100000);
System.out.println(obj);

//TODO 3
obj = new Estudante();

//TODO 4
System.out.println(obj);
}

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/347928>

```

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(){

    //TODO 1
    List<MembroUniversitario> lista = new ArrayList<MembroUniversitario>();

    //TODO 2
    MembroUniversitario b = new Bolsista(155);

    //TODO 3
    lista.add(b);

    //TODO 4
    b = new Estudante();

    //TODO 5
    lista.add(b);

    //TODO 6
    for(MembroUniversitario p : lista)
        System.out.println(p.toString());

}

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/347930>

```

/**
 * 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 {

    //TODO 2
    List<MembroUniversitario> lista;

    //TODO 3
    public Principal(){
        lista = new ArrayList<MembroUniversitario>();
    }

    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();
    }
}

//TODO 1

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

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

public void iniciar(){

    //TODO 4
    MembroUniversitario aluno = new Bolsista(33.4f);

    //TODO 5
    adicionar(aluno);

    //TODO 6
    aluno = new Estudante();

    //TODO 7
    adicionar(aluno);

    //TODO 8
    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.io/projects/347933>

/**


```

/**
 * 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. mudar o tipo dele
 * 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(){
            this.salario=500f;
        }
        public float getSalario(){
            return this.salario;
        }
        public String toString(){
            return this.getClass().getSimpleName() + "\t\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() + "\t\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();

    //
    adicionar( joao );

    //TODO 3
    imprimir();
}

```

```

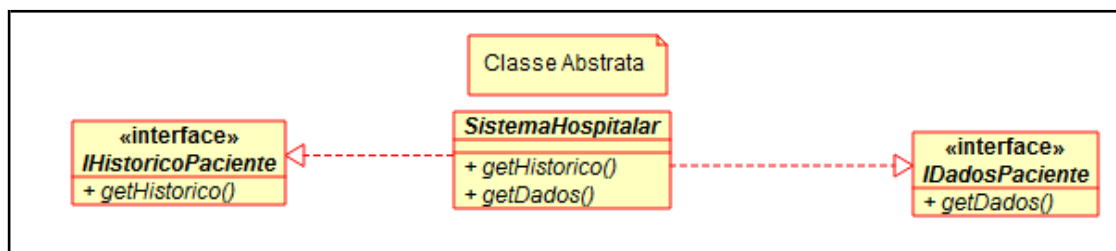
    }

    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 class Principal {

    public interface IHistoricoPaciente {

        public float getHistorico();

    }

    public interface IDadosPaciente {

        public float getDados();

    }

    public abstract class SistemaHospitalar implements IHistoricoPaciente,
IDadosPaciente{

        public abstract float getHistorico();

        public abstract float getDados();

    }

}

```

```

    }

    public Principal() {

    }

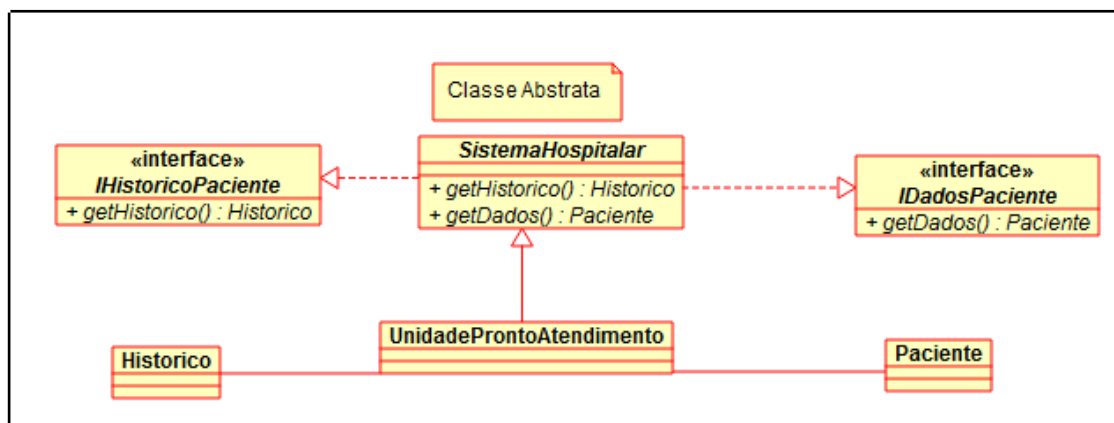
    public static void main(String[] args) {

    }

}

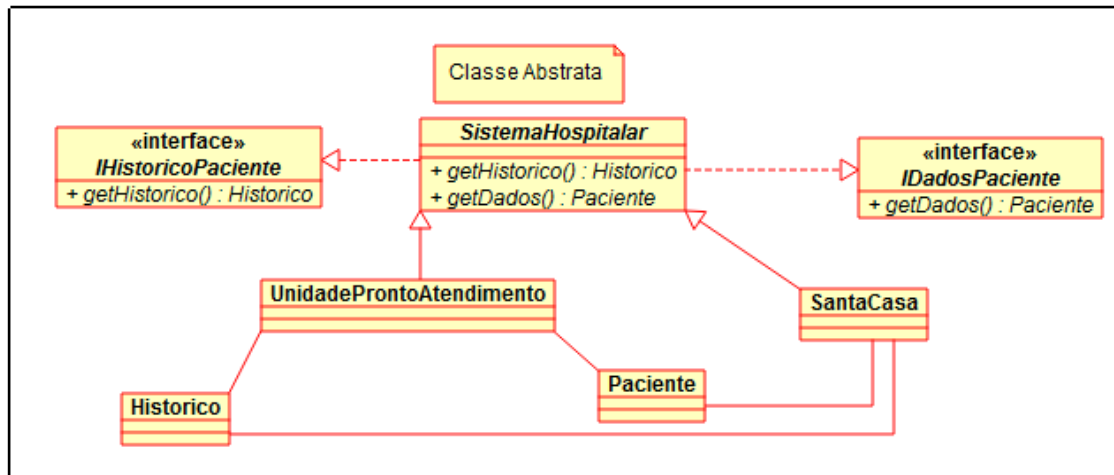
```

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.

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.

- 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?