



Disciplina: Programação Orientada a Objetos

Turma: POCO4A – 2022/2

Professor: Lucio Agostinho Rocha

Lista de Exercícios 2 (DUPLA)

1. (1,0 ponto) Observe a Figura 1 a seguir:

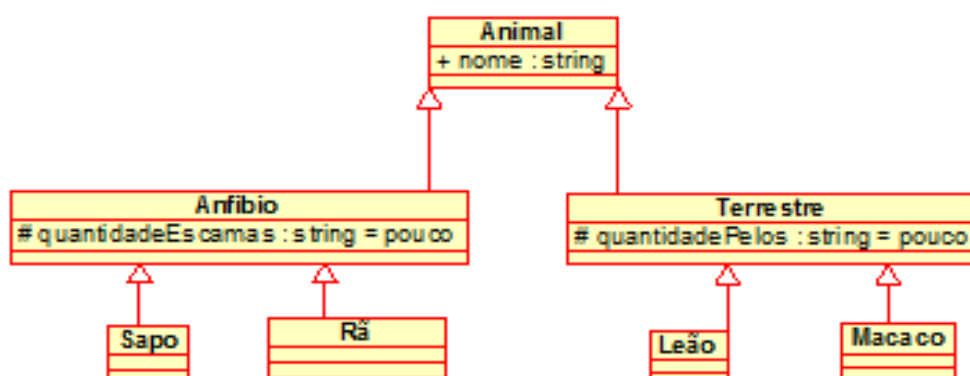


Figura 1 - Diagrama de Classes: Tipos de Animais.

Utilize Polimorfismo para implementar uma lista dinâmica de objetos das subclasses folha da Figura 1. A classe Animal possui 5 (cinco) métodos polimórficos. A classe Principal deve exibir o estado dos objetos das Classes folha com a saída dos métodos sobrecarregados da classe Animal. Utilize o trecho a seguir.

//Classe Principal.java

```
ArrayList <Animal> lista = new ArrayList<>();
lista.add(sapo);
lista.add(ra);
lista.add(leao);
lista.add(macaco);
```

2. (1,0 ponto) Observe a Figura 2 a seguir:

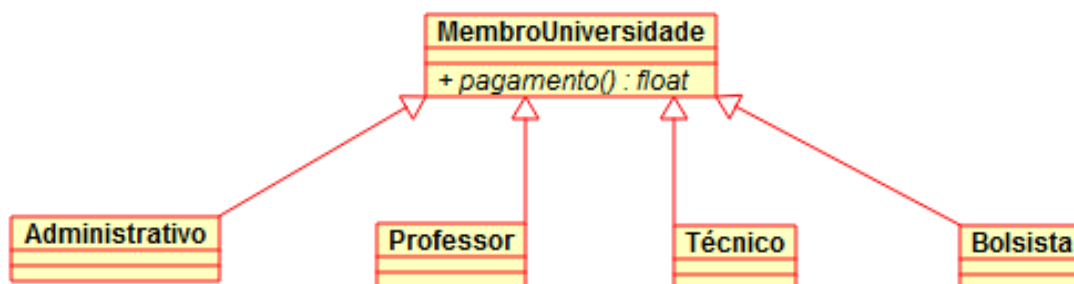


Figura 2 - Diagrama de Classes: Membros da Universidade.

Utilize Polimorfismo para implementar uma lista dinâmica de objetos das subclasses folha da Figura 2. A classe *MembroUniversidade* é uma classe abstrata, e possui 5 (cinco) métodos abstratos. A classe *Principal* deve exibir o estado dos objetos das Classes folha com a saída dos métodos implementados.

3. (1,0 ponto) Implemente com Programação Orientada a Objetos:

- Crie 3 (três) classes não relacionadas por herança: *BitCoin*, *Euro* e *Real*.
 - Dê a cada Classe atributos e comportamentos únicos que não estão presentes nas outras classes.
 - Crie a Interface '*IConversorMoeda*' com um método *getConversaoDolar()* que retorne o valor da conversão em decimal com precisão simples.
 - Cada Classe deve implementar a Interface '*IConversorMoeda*'.
 - Crie um *ArrayList<IConversorMoeda>* com objetos de cada uma das Classes *BitCoin*, *Euro* e *Real*. Itere polimorficamente pelo *ArrayList* invocando o método *getConversaoDolar* de cada objeto.
-

4. (1,0 ponto) Observe o diagrama da Figura 3 a seguir:

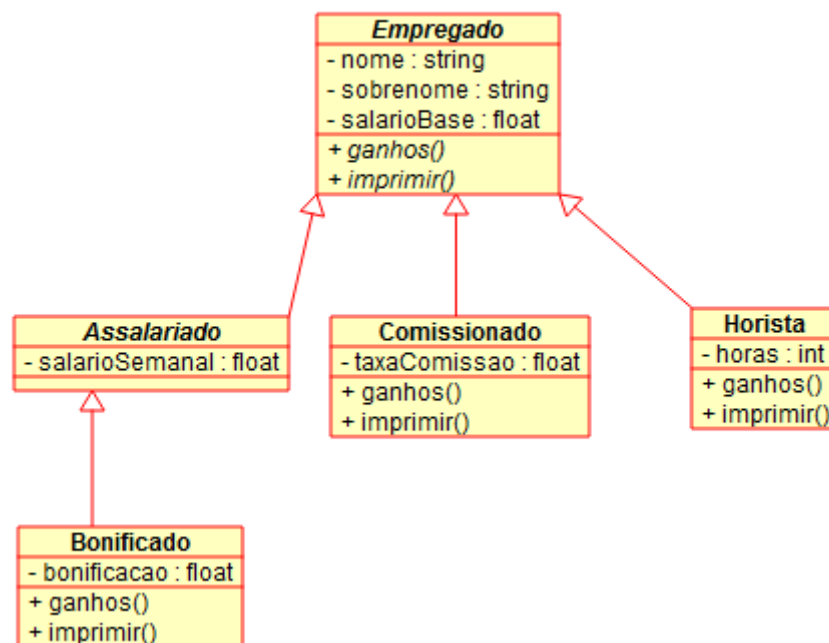


Figura 3: Diagrama de Classes: Folha de Pagamento.

a) As Classes *Empregado* e a Classe *Assalariado* são classes abstratas que possuem apenas métodos abstratos. Crie uma Classe que instancie objetos das classes derivadas da Classe raiz da hierarquia apresentada. Utilize o trecho de código a seguir:

```
Bonificado b1 = new Bonificado("Joao","Silva", salarioBase, bonificacao, salarioSemanal);  
Comissionado c1 = new Comissionado("Maria","Soares", salarioBase, taxaComissao);  
Horista h1 = new Horista("Jomar","Silva Soares", salarioBase, horas);
```

b) Utilize polimorfismo para exibir a saída dos métodos públicos das classes derivadas que podem ser instanciadas. Utilize o trecho de código a seguir:

```
for ( Empregado emp : lista ) {  
    empregado.imprimir();  
    empregado.ganhos();  
}
```

5. (1,0 ponto) Um aluno de Engenharia de Computação deseja implementar uma calculadora simples com polimorfismo de Interface. Para isso, cada operação é implementada por uma Classe: Soma, Subtração, Divisão e Multiplicação. A Interface 'IOperacoes' é implementada por essas Classes e possui os seguintes métodos abstratos:

```
void setOperando1(float operando1); //Define o valor do primeiro operando  
void setOperando2(float operando2); //Define o valor do segundo operando  
float getResultado(); //Retorna o resultado da operação  
String getNome(); //Retorna o nome da operação  
int getQuantidade(); //Retorna a quantidade de instâncias da classe
```

Utilize polimorfismo para exibir a saída dos métodos acessores dos objetos instanciados na Classe Principal.

6. (1,0 ponto) Utilize herança para criar uma superclasse de exceção (chamada ExceptionA) e subclasses de exceção ExceptionB e ExceptionC, em que ExceptionB herda de ExceptionA e ExceptionC herda de ExceptionB. Escreva um programa para demonstrar que o bloco catch para tipo ExceptionA captura exceções de tipos ExceptionB e ExceptionC.

7. (1,0 ponto) Escreva um programa que demonstra como várias exceções são capturadas com catch (Exception exception). Desta vez, defina as classes ExceptionA (que herda da classe Exception) e ExceptionB (que herda da classe ExceptionA). Em seu programa, crie blocos try que lançam exceções de tipos ExceptionA, ExceptionB, NullPointerException e IOException. Todas as exceções devem ser capturadas com blocos catch para especificar o tipo Exception.

8. (1,0 ponto) Elabore um programa orientado a objetos que instancie um objeto do tipo da interface e um objeto do tipo da classe abstrata. Acesse dois métodos de cada objeto na classe que possui o método main.

9. (1,0 ponto) Modifique o exercício anterior para incluir todo o código em uma única classe com classes e interfaces internas.

10. (1,0 ponto) Na Plataforma de Ensino, Acesse o Link do Exercício 10:

a) (0,1 ponto) Responda à postagem anterior da seguinte forma: informe, no início da postagem e antes do código-fonte, em um comentário de bloco, o nome completo dos membros da sua equipe.

b) (0,2 ponto) Informe a seguir, antes do código, em um comentário de bloco, se o exemplo de entrada e a saída informados na postagem anterior pela outra equipe está correta. Caso não esteja, informe o motivo.

c) (0,5 ponto) Informe a seguir, antes do código, em um comentário de bloco, a nova funcionalidade acrescida pela sua equipe. Acrescente a nova funcionalidade ao programa da postagem anterior.

d) (0,2 ponto) Informe a seguir, antes do código-fonte, em um comentário de bloco, um exemplo de entrada e a saída do seu programa.