Unichristus
Centro Universitário Christus

Revisão para NP2

Unidade III à Unidade VII
Herança, Classes Abstratas, Polimorfismo, Linguagem de Modelagem e Organização de Classes

Centro Universitário Christus

Professor(a): Maurício Moreira Neto	**Semestre**: 2021.1
Disciplina: Linguagens de Programação II (Orientado a Objetos)	
Curso: Sistemas de Informação	

1. Crie um programa que tenha uma classe abstrata **Animal** e duas classes concretas: **Gato** e **Cachorro**. Os métodos da classe **Animal** são todos abstratos e possuem as seguintes assinaturas:

- *String emitirSom();*
- *void mover();*
- *void descansar();*

Todos os métodos abstratos da classe **Animal** são implementados nas subclasses **Cachorro** e **Gato**

- *String emitirSom();*
  - Se a classe for Cachorro, retornar a String “Latindo...”
  - Se a classe for Gato, retornar a String “Miando...”
- *void mover();*
  - Se a classe for Cachorro, imprimir a String “Cachorro se movendo...”
  - Se a classe for Gato, imprimir a String “Gato se movendo...”
- *void descancar();*
  - Se a classe for Cachorro, imprimir a String “Cachorro descansando...”
  - Se a classe for Gato, imprimir a String “Gato descansando...”

Por fim, crie uma classe concreta **Adestrador** com os seguintes métodos:

- *Animal adestrarAnimal();*
  - Este método retorna, aleatoriamente, uma instância que pode ser um **Cachorro** ou um **Gato**
- *void brincar(Animal animal);*
  - Este método deve invocar o método *emitirSom()* da classe recebida como parâmetro e imprimir a String retorno do método *emitirSom()*

Dica: Java: utilize a classe java.util.Random

Utilize a classe Principal abaixo para testar sua solução e observe a saída do programa

```
public class Principal {
	public static void main(String[] args){
	Adestrador adestrador = new Adestrador();
	for (int i = 0; i < 10; i++) {
		Animal animal = adestrador.adestrarAnimal();
		adestrador.brincar(animal);
```

```
        }
    }
}
```

2. Crie uma interface para representar as operações bancárias: depósito e saque de uma quantia e o saldo. Crie uma classe **ContaCorrente** que implemente as operações desta interface. Para cada saque será debitada uma taxa de operação equivalente a 5% do valor sacado. Em seguida, cria uma subclasse chamada **ClasseCorrenteEspecial** que herde de **ContaCorrente**. Os clientes especiais pagam somente 1% durante a operação do saque. A Figura 1 apresenta o diagrama de classes da questão 2.

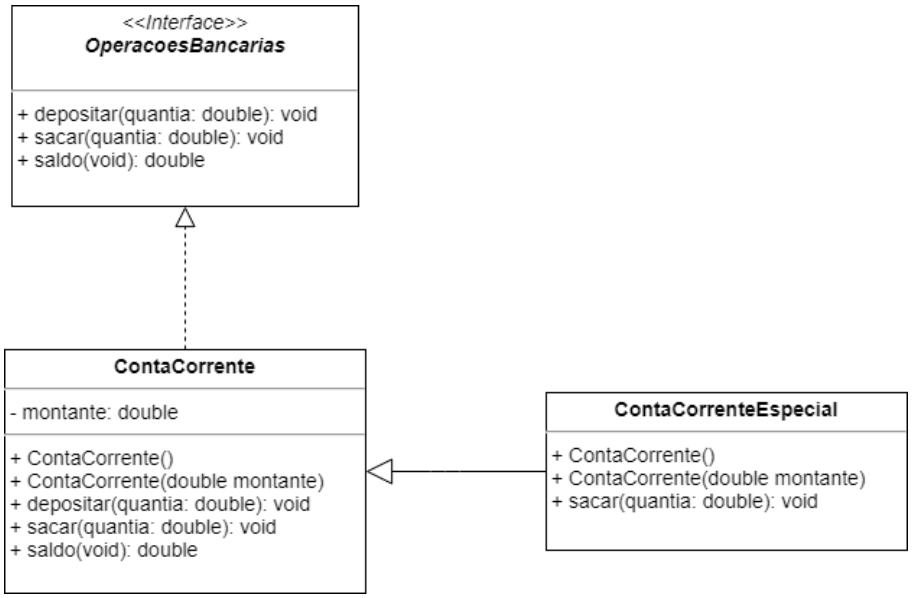

Figura. Diagrama de Classes.

3. Crie uma classe **Imovel** que possui os atributos de *endereço* e de *preço* do imóvel. O atributo *preço* deve ser privado a classe e o *endereço* deve ser visto somente por subclasses. A classe **Imovel** tem um método que retorna o valor do preço de um imóvel.

- Crie também as classes **ImovelNovo** e **ImovelVelho** que herdam da classe **Imovel**. A classe **ImovelNovo** possui um atributo privado *adicional*, um método que retorna o valor do adicional e um método que retorna o valor do imóvel novo.
- A classe **ImovelVelho** possui um atributo privado *desconto*, um método que retorna o valor do desconto e um método que retorna o valor do imóvel velho.
- Por fim, crie uma classe **ImovelNovoCorretor** que herda de **ImovelNovo** e que possui o atributo privado *porcentagem*. A classe deve sobrescrever o método que retorna o valor do imóvel novo, porém, com o valor da porcentagem do corretor.

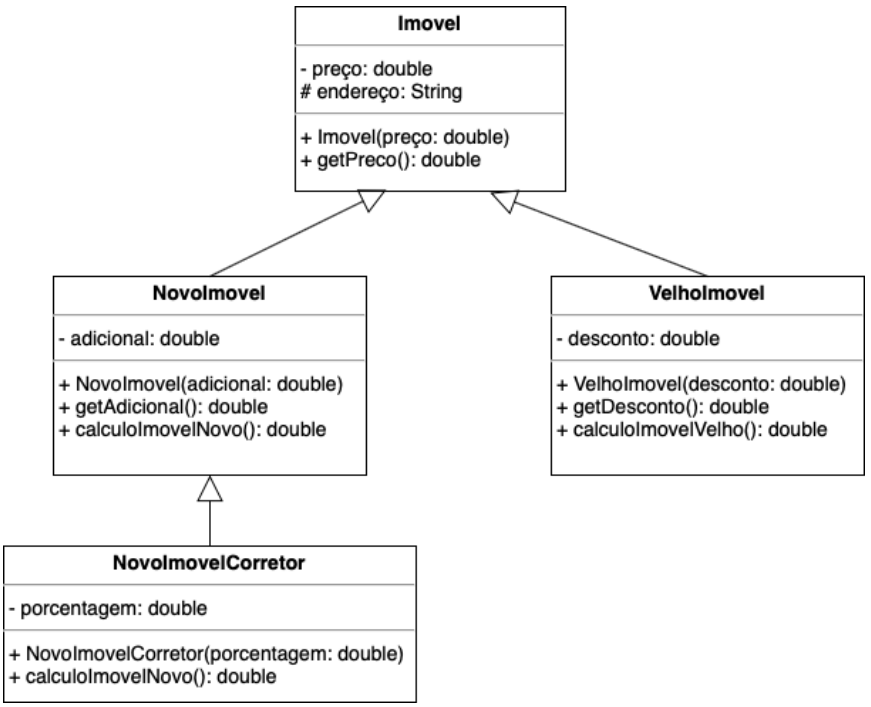

Figura. Diagrama de Classes.

4. Crie uma classe pública para representar uma pessoa, com os atributos privados nome, idade e altura. Crie os métodos públicos necessários para *sets*/*gets* e também um método público para imprimir, na saída padrão, os dados de uma pessoa.

5. Implemente uma classe Usuario que possua um identificador inteiro único (**atributo de classe**), ou seja, nunca duas instâncias de usuários possuem o mesmo identificador. O identificador deve ser gerado no momento da chamada ao construtor da classe Usuario. A geração do identificador deve ser sequencial, ou seja, primeira instância deve ter identificador 1, já a segunda instância deve ter identificador 2. Após a instanciação, objetos só podem fazer leitura do identificador. Portanto, não pode haver substituição do valor do atributo identificador após instanciação.