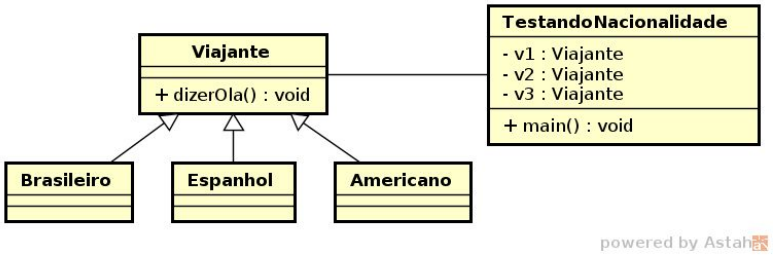
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - UFRN
Instituto Metrópole Digital
IMD0040 - Linguagem de Programação 2
Atividade: Polimorfismo, Interface e Classe Abstrata
Professor: Emerson Alencar

POLIMORFISMO

Atividade 01:

Considere a modelagem UML:



Implemente o modelo acima aplicando conceitos de Polimorfismo. Na classe *TestaNacionalidade,* faça com que cada viajante **diga Olá** em um idioma diferente.

Ex:
v3.dizerOla();
saída: *Hello*

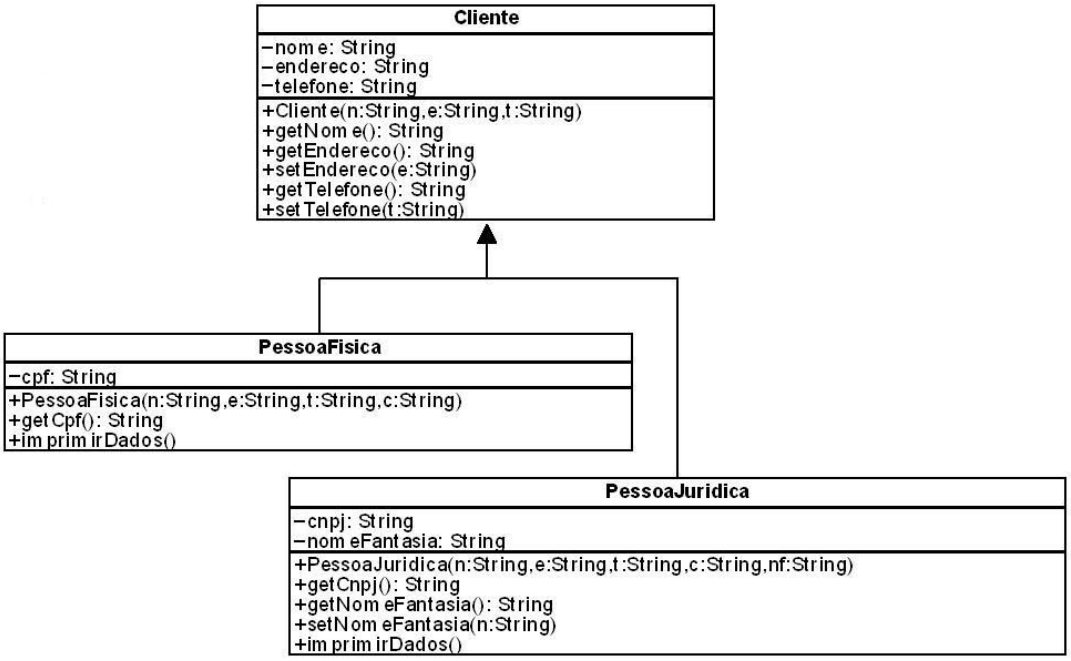
Atividade 02:

Você é o responsável por desenvolver um sistema de cadastro de clientes para uma empresa. Essa empresa deseja guardar os nomes, endereços e telefones de cada cliente. Mas essa empresa possui dois tipos de clientes: pessoas físicas e pessoas jurídicas.

O cadastro dos clientes deve ser diferente para cada tipo de cliente. Se o cliente for uma pessoa física, deverá ser armazenado seu CPF juntamente com o restante dos dados. Já se o cliente for uma pessoa jurídica, deverá ser armazenado seu CNPJ e o nome fantasia da empresa do cliente.

A empresa deseja imprimir os dados dos clientes e saber a quantidade de clientes que ela possui cadastrado no sistema, tanto a quantidade geral como a quantidade de cada tipo de cliente.

Altere a modelagem a seguir e implemente as classes necessárias utilizando os conceitos de polimorfismo.



INTERFACE E CLASSE ABSTRATA

Atividade 04:

Crie uma interface AreaCalculavel com um método calcularArea() e crie classes de figuras geométricas que implementam este método (como quadrado, circunferência e retângulo). Depois crie uma classe com um método main() para exercitar as chamadas aos métodos que calculam a área.

Atividade 05:

Crie uma classe ContaBancaria, que representa uma conta bancária genérica e não pode ser instanciada. Esta classe deve ter um atributo saldo (visível apenas para ela e para as suas subclasses) e os métodos depositar(double), sacar(double) e transferir(double, ContaBancaria). Estes métodos devem depositar um valor na conta, sacar um valor da conta e transferir um valor da conta de origem para uma conta de destino, respectivamente.

Além destes, ContaBancaria deve ter um método calcularSaldo(). Este método possui a regra do cálculo do saldo final (que pode ser diferente do saldo armazenado no atributo saldo) e deve ser obrigatoriamente implementado pelas subclasses de ContaBancaria, pois cada classe possui suas próprias regras de cálculo.

Crie duas subclasses de ContaBancaria: ContaCorrente e ContaInvestimento. Cada uma deverá implementar suas regras para calcular o saldo (método calcularSaldo()). No caso de ContaCorrente, o saldo final é o saldo atual subtraído de 10%, referente a impostos que devem ser pagos. Já para a ContaInvestimento, o saldo final é o saldo atual acrescido de 5%, referente aos rendimentos do dinheiro investido.

Crie uma aplicação que instancia uma conta corrente e uma conta investimento e executa as operações de depósito, saque, transferência e cálculo de saldo. Verifique se os resultados obtidos são consistentes com a proposta do modelo e com as regras de cálculo estabelecidas.