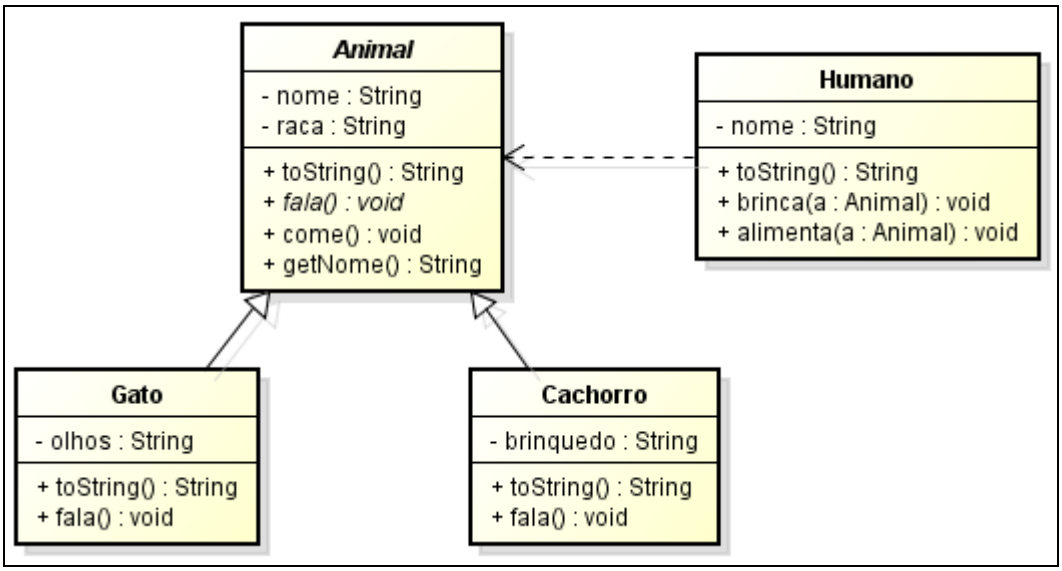
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRIÂNGULO MINEIRO
CAMPUS UBERLÂNDIA CENTRO

TECNOLOGIA EM SISTEMAS PARA INTERNET
Turma: **3º PERÍODO**
Unidade Curricular: **PROGRAMAÇÃO ORIENTADA A OBJETOS**
Professor: **WILL ROGER PEREIRA**

LISTA 3-3

Obs: Caso haja alguma divergência em relação a visibilidade, abstração e/ou parâmetros, consulte o diagrama do exercício correspondente. Bom estudo.

1º Questão



Classe Animal:
- nome : String → Nome do Animal.
- raca : String → Raça do Animal.

+ toString() : String → Retorne as informações do Animal: Seu nome e raça.
+ *fala() : void* → Todo Animal deve ser capaz de falar.
+ come() : void → Um Animal come ração. Mostre isso na tela, juntamente com seu nome.
+ getNome() : String → Retorne o nome do Animal.

Classe Gato:
- olhos : String → Cor dos olhos do Gato.

+ toString() : String → Retorne as informações da Gato: Suas informações como Animal, além de olhos.
+ fala() : void → Um Gato fala "Miau". Mostre isso na tela, juntamente com seu nome.

Classe Cachorro:
- brinquedo : String → Brinquedo do Cachorro.

+ toString() : String → Retorne as informações do Cachorro: Suas informações como Animal, além do seu brinquedo.
+ fala() : void → Um Cachorro fala "Auau!". Mostre isso na tela, juntamente com seu nome.

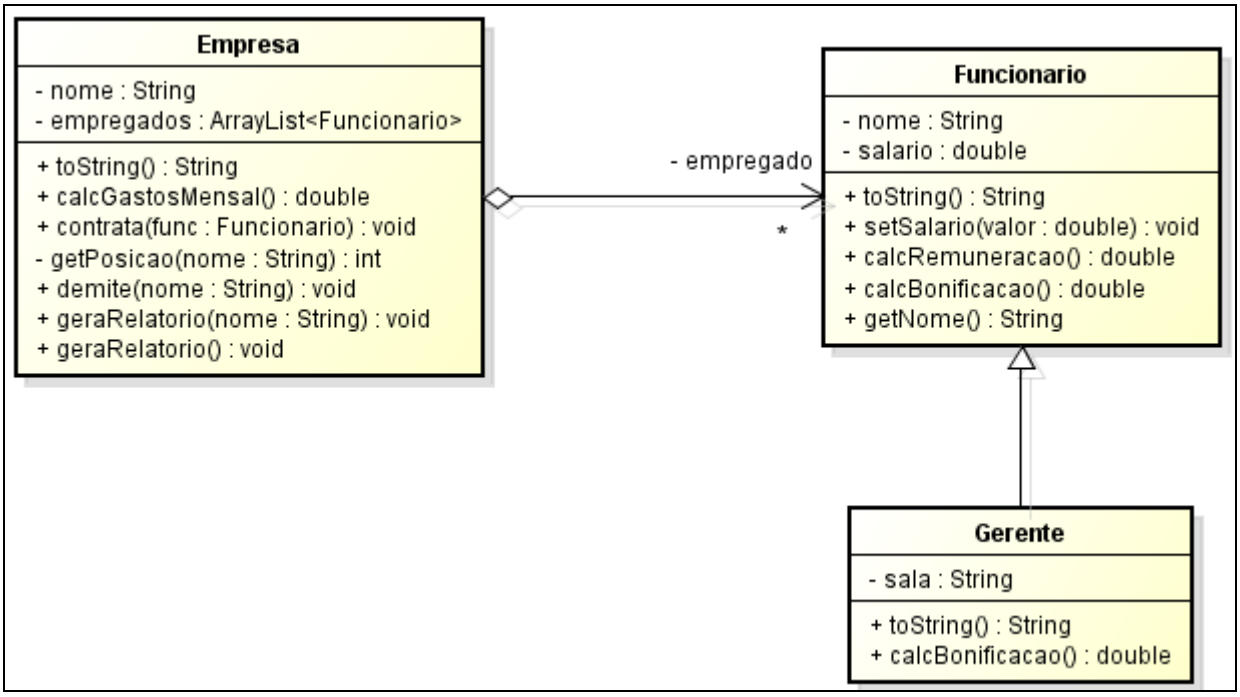
Classe Humano:
- nome : int → Nome do humano.

+ toString() : String → Retorne as informações do Humano: Seu nome.
+ brinca(a : Animal) : void → Brinca com um Animal. Identifique se o Animal é um Cachorro ou um Gato. Mostre isso na tela, juntamente com o nome dos dois envolvidos. Quando um Humano brinca com um Animal, este último fala.

+ alimenta(a : Animal) : void → Alimenta um Animal. Identifique se o Animal é um Cachorro ou um Gato. Mostre isso na tela, juntamente com o nome dos dois envolvidos. Quando um Humano brinca com um Animal, este último come.

Exercício:
Crie objetos das classes concretas, mostre suas informações na tela e invoque os métodos delas. Brinque e alimente todos os animais.

2º Questão



Classe Funcionario:
- nome : String → Nome do Funcionario.
- salario : String → Salário do Funcionario.

+ toString() : String → Retorne as informações do Funcionario: Seu nome e salário.
+ setSalario(valor : double) : void → Modifica o salário do Funcionario. O salário não pode diminuir.
+ calcRemuneracao() : double → Calcula os ganhos mensais do funcionário. Ganhos incluem salário e bonificação.
+ calcBonificacao() : double → Retorna a bonificação do funcionário. Todo funcionário tem bonificação de 10% do seu salário.

Classe Gerente:
- sala : String → Sala em que o Gerente trabalha.

+ toString() : String → Retorna as informações do Gerente: As informações dele como Funcionário, além de sua sala.
+ calcBonificacao() : double → Retorna a bonificação do Gerente, que é o dobro da de um Funcionário.

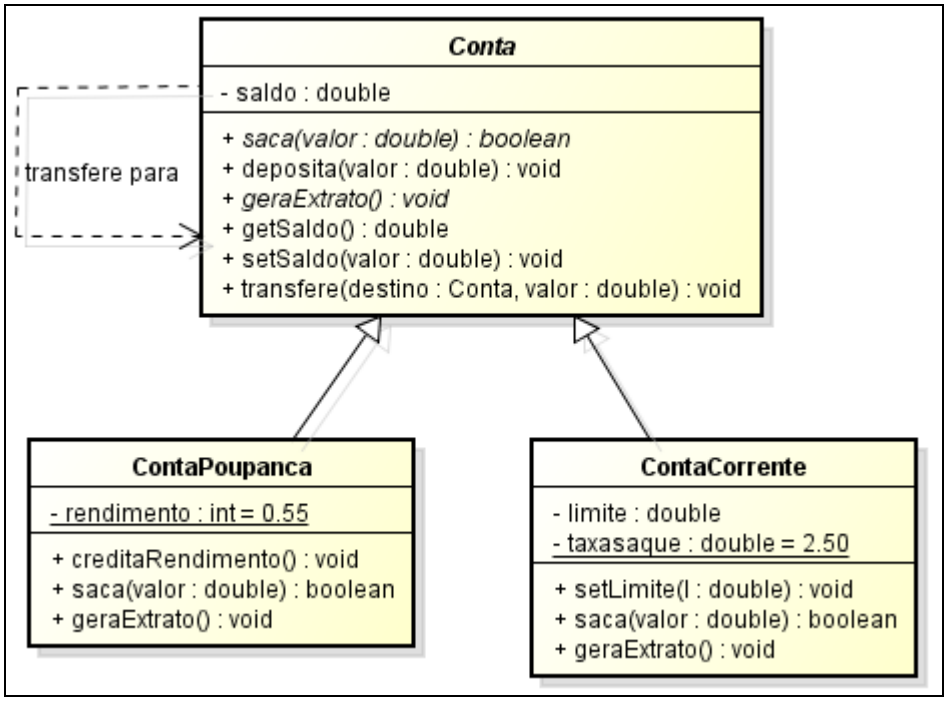
Classe Empresa:
- nome : String → Nome da empresa.
- empregados : ArrayList<Funcionario> → Lista de funcionários.

+ toString() : String → Retorna as informações da empresa: nome, a quantidade de Funcionários e o gasto mensal da empresa com seus funcionários e gerentes.
+ calcGastosMensal() : double → Calcula os gastos mensais da empresa com todos os seus Funcionários e Gerentes. Gastos incluem salário, bonificação, ou seja, a remuneração do Funcionário ou Gerente.
+ contrata(func : Funcionario) : void → Contrata um funcionário. Verifique se um Funcionário já existe antes de contratá-lo. Mostre na tela se um Funcionario ou um Gerente que foi contratado.
- getPosicao(nome : String) : int → Retorna o índice do primeiro funcionário que possuir o nome igual ao argumento do método. Caso nenhum funcionário tenha este nome, retorne o valor **-1**.
+ demite(nome : String) : void → Demite um Funcionario com o nome igual ao argumento. Pegue sua posição e o retire da lista. Mostre na tela se um Funcionario ou um Gerente que foi demitido.
+ geraRelatorio(nome : String) : void → Gera um relatório de um Funcionario com o nome igual ao argumento do método, mostrando sua informação. Mostre na tela se um Funcionario ou um Gerente que teve seu relatório gerado.
+ geraRelatorio() : void → Gera um relatório da empresa, com as informações dela e de todos os funcionários.

Exercício:
Crie objetos das classes concretas, mostre suas informações na tela e invoque os métodos delas. Contrate, demita e mostre relatórios de Funcionários e Gerentes.

3º Questão

.



Classe Conta:
- saldo : double → Saldo da Conta.

+ *saca(valor : double) : boolean* → Toda Conta deve ser capaz de realizar retirada de dinheiro.
+ deposita(valor : double) : void → Deposita uma determinada quantia na conta, contida no argumento. O argumento deve ser positivo.
+ *geraExtrato() : double* → Toda Conta deve ser capaz de gerar extrato.
+ getSaldo() : double → Retorna o saldo da conta.
+ setSaldo(valor : double) : void → Modifica o saldo da Conta, baseado no argumento.
+ transfere(destino : Conta, valor : double) : void → Transfere valor de uma Conta para a Conta destino. Utilize os métodos saca e deposita para realizar tal operação. Mostre de e para qual tipo de Conta, seja ContaCorrente ou ContaPoupanca, o valor foi transferido.

Classe ContaPoupanca:
- rendimentos : double = 0.55 → Rendimentos mensais, em porcentagem(%).

+ creditaRendimento() : void → Deposita o rendimento na ContaPoupanca.
+ saca(valor : double) : boolean → Saca uma determinada quantia da ContaCorrente. O saldo deve ser sempre um número natural.
+ geraExtrato() : double → Gera informações sobre a ContaCorrente, contendo seu saldo e rendimentos.

Classe ContaCorrente:
- limite : double → Limite da ContaCorrente, ou seja, o quão negativo o saldo pode ficar.
- tarifa : double = 2.50 → Tarifa cobrada a cada operação de saque, em reais.

+ setLimite(l : double) : void → Modifica o limite da conta, baseado no argumento.
+ saca(double) : boolean → Saca uma determinada quantia da conta. O saldo deve ser sempre maior do que –limite, ou seja, o negativo do limite. Lembre-se de sacar a tarifa também, verificando se pode ser sacado.
+ geraExtrato() : double → Gera informações sobre a ContaPoupança, contendo seu saldo, limite e tarifa.

Exercício:
Crie objetos das classes concretas, gere extratos e invoque os métodos delas. Saque, deposite e transfira dinheiro.