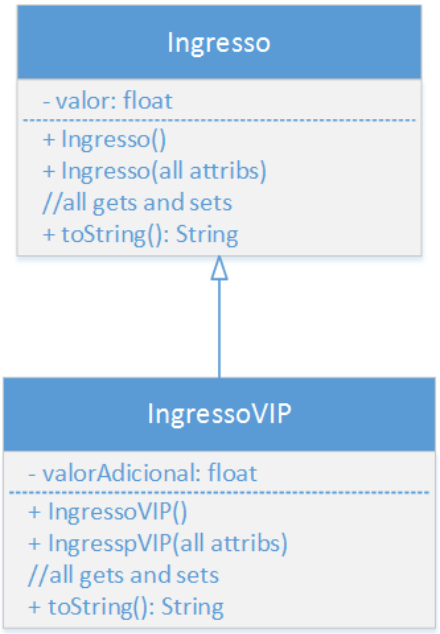
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE	**Curso Técnico em Informática**
Nome:	Data:
Disciplina de Linguagem de Programação I	Professor: Ricardo Luis dos Santos

Exercícios sobre Orientação a Objetos - Herança

Lista I

1. Crie uma classe chamada Ingresso que possua um atributo valor e um método toString que retorne à informação do valor do ingresso.
    a. Crie uma classe IngressoVIP, que herda de Ingresso e possui um atributo valor Adicional. O método toString da classe IngressoVIP deve considerar que o valor do ingresso é o valor da superclasse somado ao valor Adicional do IngressoVIP.
    b. Crie uma classe para testar os objetos das classes Ingresso e IngressoVIP.



2. Crie classes de forma a representar o diagrama a abaixo:

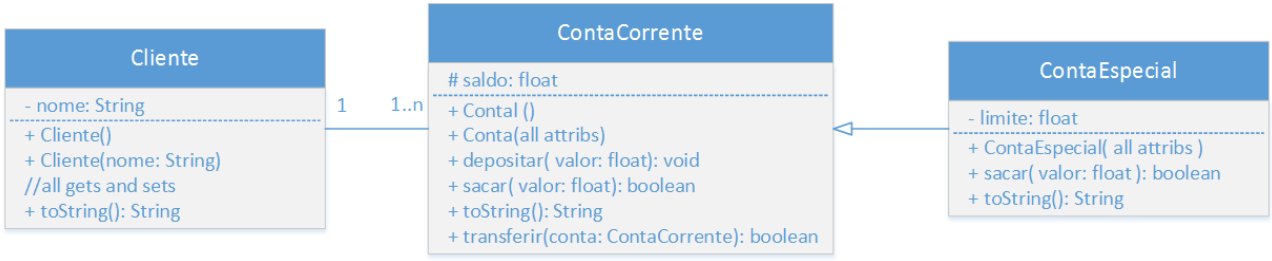
    a. A classe Empregado deve possuir dois atributos, nome e salario. Salario deve ser do tipo protected. Crie os métodos get e set para classes e o método toString.
    b. As classes Gerente deve herdar da classe Empregado. Crie os métodos get e set para a classe e o método toString. O método toString da classe Gerente deve incluir a informação do departamento, além dos dados da superclasse. O construtor da classe

deve receber por parâmetro, além as informações da superclasse, a informação do departamento.

c. A classe Vendedor deve herdar também da classe Empregado. Deve possuir ainda um método denominado calcularSalario. Esse método deve retornar um valor do tipo float, correspondente ao valor do salário acrescido do respectivo percentual de comissão. O construtor da classe deve receber por parâmetro, além as informações da superclasse, a informação do percentual de comissão do vendedor. O método toString da classe deve apresentar as informações de nome do empregado, salário sem comissão, salario com comissão e percentual de comissão.

d. Crie uma classe para testar objetos das classes implementadas.

3. Crie classes de forma a representar o diagrama a seguir:



a. A classe ContaEspecial herda da classe ContaCorrente.

b. Clientes que possuem conta especial possuem um limite de crédito. Dessa forma, podem fazer saques até esse valor limite, mesmo que não possuam saldo suficiente na conta.

c. O construtor da classe ContaEspecial deve receber como parâmetro, além dos parâmetros da superclasse, o limite que o banco disponibiliza para o cliente.

d. Sobrescreva o método sacar na classe ContaEspecial, de modo que o cliente possa ficar com saldo negativo até o valor de seu limite. Note que o atributo saldo da classe ContaCorrente deve ser do tipo protected para que possa ser modificado na subclasse.

Lista II

1. Criar o diagrama UML das classes abaixo. Colocar todas as classes criadas em um único package.

2. Cria uma Classe Pessoa, contendo os atributos encapsulados, com seus respectivos seletores (getters) e modificadores (setters), e ainda o construtor padrão e pelo menos mais duas opções de construtores conforme sua percepção. Atributos: String nome; String endereço; String telefone;

3. Considere, como subclasse da classe Pessoa (desenvolvida no exercício anterior) a classe Fornecedor. Considere que cada instância da classe Fornecedor tem, para além dos atributos que caracterizam a classe Pessoa, os atributos valorCredito (correspondente ao crédito máximo atribuído ao fornecedor) e valorDivida (montante da dívida para com o fornecedor). Implemente na classe Fornecedor, para além dos usuais métodos seletores e modificadores, um método obterSaldo() que devolve a diferença entre os valores dos atributos valorCredito e valorDivida. Depois de implementada a classe Fornecedor, crie um programa de teste

adequado que lhe permita verificar o funcionamento dos métodos implementados na classe Fornecedor e os herdados da classe Pessoa.

4. Considere, como subclasse da classe Pessoa, a classe Empregado. Considere que cada instância da classe Empregado tem, para além dos atributos que caracterizam a classe Pessoa, os atributos codigoSetor (inteiro), salarioBase (vencimento base) e imposto (porcentagem retida dos impostos). Implemente a classe Empregado com métodos seletores e modificadores e um método calcularSalario. Escreva um programa de teste adequado para a classe Empregado.

5. Implemente a classe Administrador como subclasse da classe Empregado. Um determinado administrador tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo ajudaDeCusto (ajudas referentes a viagens, estadias, ...). Note que deverá redefinir na classe Administrador o método herdado calcularSalario (o salário de um administrador é equivalente ao salário de um empregado usual acrescido da ajuda de custo). Escreva um programa de teste adequado para esta classe.

6. Implemente a classe Operario como subclasse da classe Empregado. Um determinado operário tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo valorProducao (que corresponde ao valor monetário dos artigos efetivamente produzidos pelo operário) e comissao (que corresponde à porcentagem do valorProducao que será adicionado ao vencimento base do operário). Note que deverá redefinir nesta subclasse o método herdado calcularSalario (o salário de um operário é equivalente ao salário de um empregado usual acrescido da referida comissão). Escreva um programa de teste adequado para esta classe.

7. Implemente a classe Vendedor como subclasse da classe Empregado. Um determinado vendedor tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo valorVendas (correspondente ao valor monetário dos artigos vendidos) e o atributo comissao (porcentagem do valorVendas que será adicionado ao vencimento base do Vendedor). Note que deverá redefinir nesta subclasse o método herdado calcularSalario (o salário de um vendedor é equivalente ao salário de um empregado usual acrescido da referida comissão). Escreva um programa de teste adequado para esta classe.