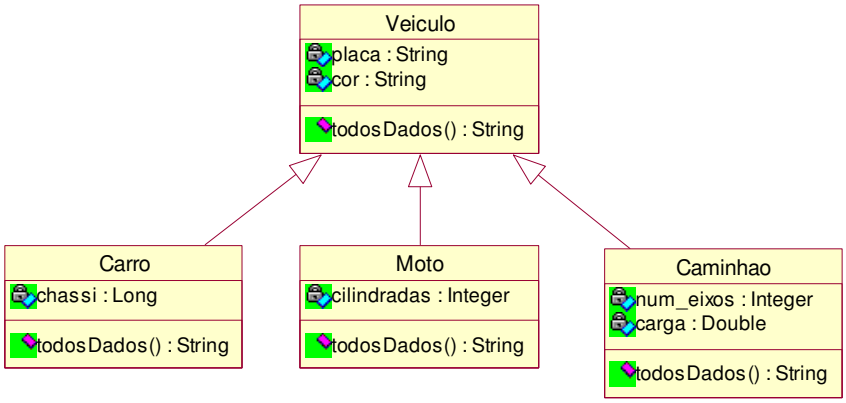
Universidade Federal de São João del-Rei – Campus Alto Paraopeba

DTECH – Departamento de Tecnologia em Engenharia, Computação e Humanidades

Exercícios de Programação Orientada a Objetos - JAVA

Prof.: Rone Ilídio / Thiago Oliveira

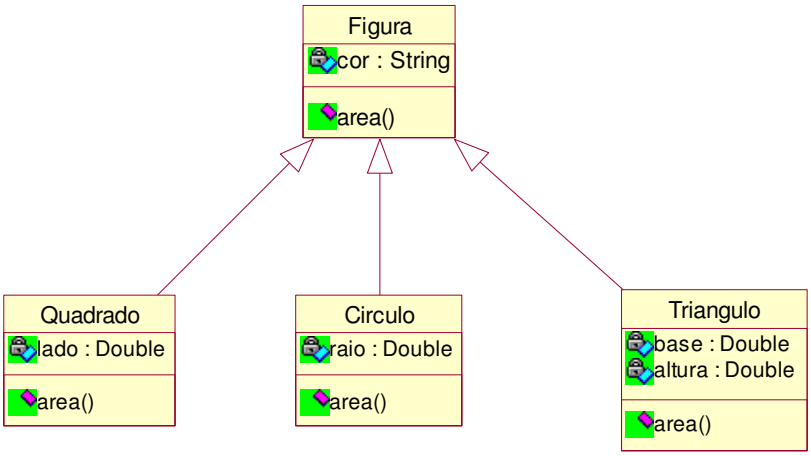
1) Crie a seguinte estrutura de classes:



O método todosDados deve retornar uma string contento todos os atributos do objeto. O construtor de cada classe deve iniciar os objetos com valores nulos.

Crie um *applet* que possua uma coleção de objetos do tipo Veículo. Para preencher a coleção, o usuário deve escolher repetidamente se quer cadastrar um carro, um caminhão ou uma moto. Deve-se criar botões na tela para cada tipo de veículo e para listar todos. Ao final, imprima todos os dados dos objetos cadastrados dentro do *appletviewer*.

2) Crie a seguinte estrutura de classes.

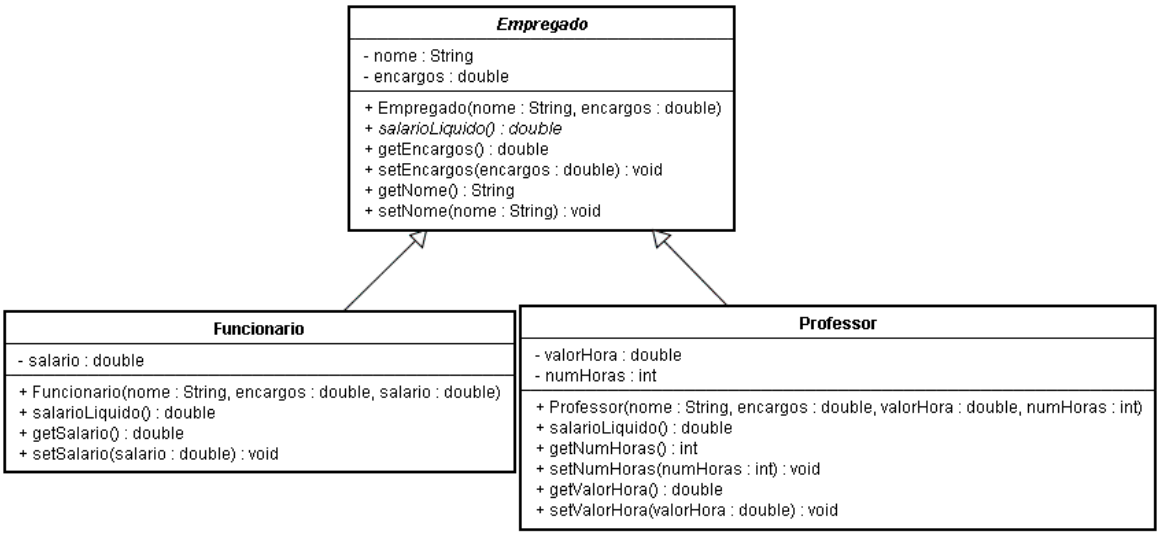


Obs.: O método área deve ser polimórfico e a classe figura abstrata.
Crie um *applet* que possua uma coleção de objetos do tipo Figura. Crie botões com as opções para adicionar elementos à coleção: cadastrar Quadrados, cadastrar Círculos e cadastrar Triângulos. Os valores de cada objeto devem ser passados pelo usuário. Assim que for

realizado o cadastro da figura, o valor da sua área deve ser exibido para o usuário. Deve existir também uma opção exiba todos os dados da figura que possuir a maior área.

3) Implemente uma classe denominada Retangulo. Seus atributos serão altura e largura. Esta classe possuirá um método denominado "area" que calculará e retornará a área do retângulo, utilizando para isso os valores contidos nos atributos altura e largura.
Crie um *applet* com campos onde o usuário insira a altura e a largura do retângulo. Ao clicar no botão Calcular, um objeto da classe retângulo deve ser criado e este *applet* deve então exibir a área do retângulo, a partir do método area do objeto.

4) Crie a estrutura de classes modelada no diagrama abaixo:



Essa estrutura de classes será utilizada para um programa do departamento pessoal de uma escola para calcular quanto cada empregado ganha. Na classe *Empregado* o atributo *nome* armazenará o nome de um funcionário e o atributo *encargos* armazenará o valor que será descontado do salário do empregado. Na classe *Funcionario*, o atributo *salario* armazenará o salário de um funcionario. Na classe *Professor,* o atributo *valorHora* armazenará o valor de uma hora de trabalho de um professor e o atributo *numHoras* armazenará quantas horas o professor trabalha por mês. O método *salarioLiquido()* é polimórfico. Na classe *Funcionario* ele retorna o *salario* menos os *encargos*. Na classe Professor ele retorna a multiplicação do número de horas trabalhadas pelo valor da hora, menos os encargos. Os construtores de todas as classes recebem valores para todos os atributos.

Crie um *applet* que possua uma coleção do tipo empregado e utilize botões para implementar na tela as seguintes opções:
a) Cadastra um objeto do tipo funcionário na coleção: o cadastro do funcionário deve estar obrigatoriamente dentro de um método denominado cadastrarFunc(), o qual não recebe parâmetros e retorna *true* ou *false,* se o objeto foi cadastrado ou não.
b) Cadastra um objeto do tipo professor: procedimento semelhante.
c) Solicita ao usuário um valor e exibe os nomes e as rendas de todos os empregados que possuam renda (método *salarioLiquido()*) igual ou superior ao valor informado. Deve ser implementada em uma função que recebe o valor informado pelo usuário e exibe na tela o resultado, ou seja, não possui retorno.

5) Crie uma classe denominada *Produto*, com os atributos nome (*String*) e preço (*double*), ambos encapsulados. Crie uma classe denominada *Venda*, a qual possui uma coleção de objetos do tipo *Produto*. Crie os seguintes métodos nessa classe, como descrito:

- Inserir: recebe um objeto do tipo *Produto* e o insere na coleção.
- Remover: recebe um nome de produto e remove tal objeto da coleção.
- Exibir: não recebe parâmetros e retorna dentro de uma *String* todos os dados de todos os produtos cadastrados e o total da venda (somatório de todos os preços).
- MaisCaro: não recebe parâmetros e retorna o objeto que possui o maior preço dentre os cadastrados.

Crie uma terceira classe, que herdará de *JFrame* e deverá possuir um objeto do tipo Venda. Implemente botões ou formas mais avançadas para chamar cada um dos métodos do objeto Venda. O programador está livre para inserir qualquer elemento novo que for necessário.

6) Crie um *applet* equivalente ao teclado virtual de bancos online. Ele deve conter botões com todos os algarismos, uma caixa de texto para o número da conta, outra para o número da agência e outra para a senha. As caixas de texto devem ser preenchidas normalmente, exceto a caixa de texto da senha que deve ser preenchida pelo teclado virtual. Crie também um botão para validar a senha. Exemplo de conta: agência: 2011-2, conta: 011220-11, senha: 123456.

7) Crie um aplicativo que quando o usuário clica na tela aparece um ponto naquela posição, ou seja, um pequeno círculo é desenhado na posição que foi clicado. Para facilitar a implementação crie uma classe Ponto com os atributos x e y. Na classe Principal, crie um vetor de pontos. Para cada clique na tela do aplicativo, crie um objeto novo no vetor e chame o método *repaint()* (método de JFrame/JApplet). Desenhe os círculos com o método *paint* do *JFrame*, utilizando para isso os pontos contidos no vetor.
Obs: o método *repaint* serve para força a execução do método *paint*.
Obs: modifique o programa utilizando o evento *mouseDragged* de *MouseMotionListener*.

8) Implemente a hierarquia de classes **ContaBancaria** (superclasse, com os atributos: senha, número e saldo), **ContaCorrente** (com um atributo para armazenar quantidade de transações realizadas e outro para armazenar o limite do cheque especial) e **ContaPoupanca** (com um atributo para armazenar a taxa de rendimento do último mês). Crie um construtor em **ContaBancaria** que recebe dados de todos os atributos. Além dos métodos get e set para cada um dos atributos, essa classe dever possuir um método abstrato denominado *Saca*(), o qual recebe como parâmetro um valor a ser retirado do saldo. Na classe ContaCorrente, tal método dever reduzir o valor do saldo até o limite do cheque especial, ou seja, os objetos podem ter saldos negativos. Na classe ContaPoupanca, tal método só aceita valores positivos.

Crie um aplicativo que simula o programa utilizado em um caixa eletrônico. Tal programa deve possuir um vetor do tipo ContaBancaria para salvar objetos do tipo ContaCorrente e ContaPoupança, os quais serão manipulados pelo programa. Crie também uma interface onde o usuário escolhe em uma tela uma das seguintes opções:

- Criar conta corrente
- Criar conta poupança
- Sacar
- Depositar
- Verificar Saldo

Crie uma tela para cada uma dessas opções. Todas as contas devem ser criadas com o saldo igual a zero. Para a opção de Sacar e Verificar Saldo, o computador deve pedir para que o usuário informe número da conta e senha. Para a opção Depositar, o usuário deve inserir somente o número da conta e, logicamente, o valor depositado. A opção Verificar Saldo deve exibir todos os dados da conta, exceto a senha.

O programador é responsável em criar qualquer outra tela ou funcionalidade que ser torne necessária para o programa.

9) Crie um *applet* que representa um jogo da velha. Ele deve conter 9 botões que quando clicados apareçam neles O ou X, dependendo de qual jogador está jogando no momento. Crie um botão para apagar todos os O e X.

10) Implementar uma calculadora (+ - * /) segundo modelo do Windows.

11) Crie a estrutura de classes exibida abaixo. Crie um *applet* com uma lista do tipo Casamento e outra do tipo Aniversário. Implemente dois botões: Cadastrar casamento e Cadastrar aniversário para manipular as listas. Outro botão Procurar evento deve pedir para o usuário um nome e exibir todos os dados de todos os eventos que possuem o referido nome (seja aniversariante, noivo ou noiva).