LP II - 2013.2 / 1ª Lista de Exercícios

Em todos os exercícios a aluno deve atender aos requisitos enunciados. Métodos e variáveis auxiliares podem ser criadas e usadas, desde que pertinentes. O aluno deve necessariamente empregar e explorar as características de orientação a objetos do Java:

- Encapsulamento (incluindo modificadores de acesso),
- Herança (de classe e interface) e polimorfismo; Classes e métodos abstratos.
- Sobrecarga de métodos;
- Tratamento e geração de Exceções;
- Uso das classes básicas (Object, por exemplo);
- Classes / pacotes

1º) Exercício

Crie uma aplicação que receba argumentos da linha de comando e, de acordo com os argumentos, calcula a área de um retângulo ou trapézio e imprime uma mensagem adequada.

Se receber dois ou três argumentos, calcula e exibe o valor da área do retângulo ou do trapézio, respectivamente. Caso contrário exibe uma mensagem informando ao usuário o número de argumentos inserido e que o mesmo não é válido.

Se ocorrer alguma exceção de conversão de números (ou seja, tem que tratar as exceções), a aplicação deve exibir a mensagem "`Você não inseriu números validos. Tente novamente`".

Trate todos os outros problemas possíveis de se antecipar. Pode usar uma técnica de deteção de erro ou capturar exceões confirmadas e não confirmadas.

Para esta 1ª experiência, o programa pode ser codificado todo no método *main* (sabendo que isso não é incentivado), como na Figura 1.

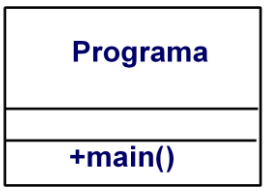

Figura 1. Diagrama de classe - Ex1

Obs.: Para calcular a área do trapézio considere a entrada na seguinte ordem: <*base maior*> <*base menor*> <*altura*>.

2º) Exercício

Implemente a classe *Frase*, com os seguintes métodos de classe:

- *contaVogais*: recebe como parâmetro um objeto da classe *String* e retorna o número de vogais presentes nela;
- *contaConsoantes*: recebe como parâmetro uma referência a um objeto da *String* e retorna o número de consoantes presentes nela;
- *contaEspacos*: recebe como parâmetro uma "string" e retorna o número de espaços em branco presentes nela.
- *Ordena*: recebe como parâmetro uma "string" e retorna um objeto *String* com as palavras dessa frase em ordem alfabética. Caso o usuário não insira uma frase, o programa deve exibir uma mensagem.

PS.: Observe que, propositalmente, usamos formas diferentes para descrever o argumento de entrada... além disso, como "número de xxx" podemos inferir como sendo *int* o tipo do retorno.

Implemente a classe *LeFrase*, que recebe "strings" via *stream* de entrada (*System.in*) e conta seu número de vogais, consoantes, espaços em brancos e ordena as palavras. A aplicação deve ler "strings" até que o usuário tecle `<ENTER>` para encerrar o programa.

Exemplo:

```
exc02> java LeFrase
Digite uma frase ou tecle <Enter> para encerrar: Sem osmose e som, somes.
A frase possui:
8 vogais
10 consoantes
4 espacos em branco
Digite uma frase ou tecle <Enter> para encerrar:
```

PS.: também vamos admitir que o programa "principal" vai estar todo contido no método *main*, que será codificado na classe *LeFrase*. Mas para realizar as operações, objetos da classe *Frase* devem ser instanciados. Veja a Figura 2.

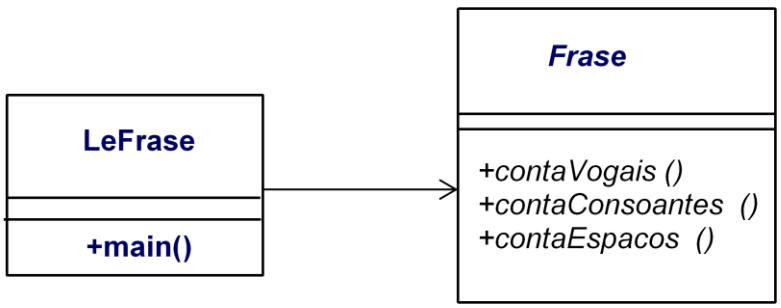

Figura 2. Diagrama de classe - Ex2

3º) Exercício

A experiência mostra que os sólidos se dilatam ao sofrerem um aquecimento, e se contraem, ao serem resfriados. Vamos criar uma aplicação que calcule a dilatação sofrida por um sólido.

a) Implemente a classe *Solido* com campos para a temperatura inicial, a medida inicial e o coeficiente de dilatação do sólido (que são valores do tipo *real*) e o seguintes métodos:

- *toString* (que sobrepõe o método *toString* da classe *Object*) para que imprima todos os dados referentes ao objeto *Solido*;
- **dois** construtores que inicializem todos os campos da classe: um que recebe os valores no tipo básico que represente o tipo real e outro que recebe os valores como *String*.

b) Crie uma classe chamada *Dilatacao* com campos do tipo *real* para o tamanho final e a temperatura final do sólido e também para a variação de temperatura e de tamanho sofridas por ele. Essa classe também deve conter um campo para um objeto *Solido*. O construtor da classe *Dilatacao* recebe como argumentos um objeto do tipo *Solido* e a temperatura final atingida por esse objeto, calcula a variação de temperatura ocorrida e inicializa os respectivos campos. Além do construtor, a classe *Dilatacao* deve possuir os seguintes métodos:

- *calculaDilatacao* : Realiza o cálculo da variação de tamanho sofrida pelo objeto;
- *toString* : imprime todos os dados do objeto, a variação de temperatura, dilatação sofrida e o tamanho final atingido pelo objeto.

As classes acima descritas devem implementar também um método *getXXX* para cada campo da classe (onde "XXX" é o nome do campo). Estes métodos são geralmente chamados métodos de acesso.

c) Crie uma classe para o programa principal, chamada *Dilatacoes* (não consegui pensar em um nome melhor ;-) em que deve haver um vetor (um *array*) do tipo *Dilatacao*.

Atenção: está classe conterá o método *main* e TAMBÉM um construtor e um método *calculaDilatacao.*

O método *main* deve fazer o seguinte:

1) pedir ao usuário o número de objetos da classe *Dilatacao* para os quais a dilatação será calculada);
2) criar uma instância da classe *Dilatacoes* (sim, isso mesmo!). Para isso você também vai ter que resolver onde vai ficar a referência a esta instância: global à classe ou local ao método *main*. O construtor da classe *Dilatacoes* deve receber como argumento o número lido anteriormente;
3) invocar o método *calculaDilatacao* no objeto *Dilatacoes.*

O construtor da classe *Dilatacoes* recebe o número de objetos da classe *Dilatacao* para os quais a dilatação será calculada e com este número instancia o *array* já descrito.

O método da *calculaDilatacao* faz, na realidade o trabalho "importante" (também não é o ideal, mas é melhor do que deixar tudo dentro do método *main*). Não recebe argumentos e retorna *void*. Ao ser executado, este método deve ler os dados necessário para cada objeto

Solido e *Dilatação*, criar a instância e armazená-lo no *array*. Após ler os dados do número adequado de objetos a aplicação faz os cálculos e exibe o resultado.

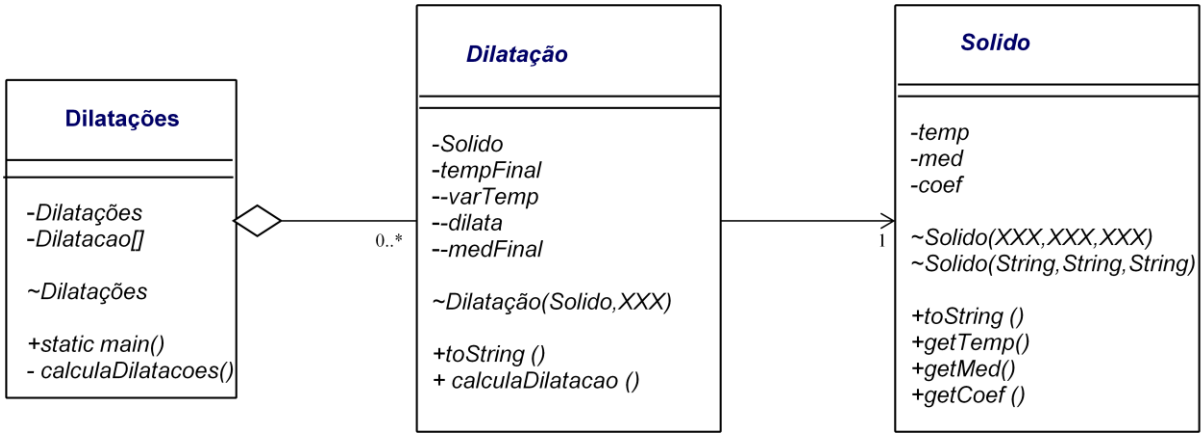

Figura 3. Diagrama de classe - Ex3

Na criação de cada objeto *Solido*, o programa deve alternar o uso dos dois construtores desta classe. O objetivo é pedagógico. Bole a sua forma para impor a alternância...

Exemplo de execução

```
Digite o tamanho do vetor:
2
 -> Solido #1
Medida Inicial (m): 6
Temperatura Inicial (¦C): 10
Temperatura Final (¦C): 50
Coeficiente de Dilatacao (1/¦C): 0.000017

 -> Solido #2
Medida Inicial (m): 6
Temperatura Inicial (¦C): 50
Temperatura Final (¦C): 10
Coeficiente de Dilatacao (1/¦C): 0.000017

Resultado
 -----> Solido #1
Medida Inicial: 6.0 u.m.
Temperatura Inicial: 10.0 graus C
Coeficiente de Dilatacao: 1.7E-5 1/C
Variacao de Temperatura: 40.0 graus
Dilatacao Sofrida: 0.00408
Medida Final: 6.00408 u.m.

 -----> Solido #2
Medida Inicial: 6.0 u.m.
Temperatura Inicial: 50.0 graus C
Coeficiente de Dilatacao: 1.7E-5 1/C
Variacao de Temperatura: -40.0 graus
Dilatacao Sofrida: -0.00408
Medida Final: 5.99592 u.m.
```

4º) Exercício

Uma empresa contratou novos empregados e deseja calcular o valor de seus salários após a aplicação do desconto do Imposto de Renda. No caso de funcionários com dependentes, o salário-base é a soma do salário com o salário-família.

a) Crie a classe *Empregado* com campos *String* para o nome e o código do empregado e campos do tipo real para o salário e o salário-líquido e os métodos:

- Um construtor que recebe como parâmetros dois objetos *String* e um valor *real*, e inicializa os campos referentes a nome, código do empregado, salário e salário-base (que devem ser inicializados com o mesmo valor). *Atenção: não deve haver nessa classe nenhum outro construtor*.
- `double calculaSalario(double desconto)` : retorna o valor do salário-líquido a ser recebido pelo empregado, que é o valor do salário-base reduzido do percentual de desconto passado ao método;
- `String toString()` : retorna o nome, código e o salário-base do empregado.

b) Implemente a classe *EmpregadoContratado* que estende a classe *Empregado* e que possua, além dos campos herdados, um campo do tipo *int* para o número de dependentes, um campo do tipo *real* para o salário-família e dois campos de valor constante: *valorPorDep* (que vale R$ 9.58, para o cálculo do salário-família) e *aliquotaIR* (valendo 15%, para o cálculo do desconto do IR). Essa classe deve implementar os seguintes métodos:

- Um construtor que recebe como parâmetros duas "strings", um número real e um inteiro, e inicializa os campos nome, código, salário e número de dependentes. Atenção: *é obrigatória uma referência explícita ao construtor da classe Empregado, que foi definido acima.*
- `void calculaSalario()` : Calcula o salário-líquido do empregado, invocando o método *calculaSalario* da superclasse com o valor da *aliquotaIR* como parâmetro.
- `void calculaSalario(int numeroDependentes)` : Para empregados que possuam dependentes. Calcula o valor no salário-base do empregado, acrescentado ao salário o valor do salário-família (*número de dependentes * valPorDep*), e invoca o método *calculaSalario*() para calcular seu salário-líquido.
- `String toString()` : retorna um objeto *String* contendo o nome, código, salário-base e salário-líquido do Empregado.

c) Crie uma classe para o programa principal que crie instâncias da classe *EmpregadoContratado* (recebendo os dados via fluxo de entrada) e imprima seus dados calculados. A aplicação deve executar até que o usuário opte por seu encerramento. Empregue a mesma estrutura de "classe principal" adotada no 4º Exercício.

Exemplo:

```
--- Folha Salarial ---
Nome do Empregado: Rubens Andrade
Codigo: Emp01
Salario:500
Numero de Dependentes: 0
---
Nome: Rubens Andrade
Codigo: Emp01
Salario-Base: 500.0
Salario-Liquido: 425.0
```

```
Nome do Empregado: Paulo Ricardo
Codigo: Emp02
Salario: 500
Numero de Dependentes: 2
---
Nome: Paulo Ricardo
Codigo: Emp02
Salario-Base: 519.16
Salario-Liquido: 441.28599999999994
```

Ex5. Crie a classe *Pessoa* com os campos protegidos (encapsulados), *nome* e *dataNascimento*, objetos da classe *String*, que vão representar o nome e data de nascimento. A classe *Pessoa* deve conter:

- Um construtor que recebe como parâmetros duas *strings* e inicializa os campos *nome* e *dataNascimento*.
- O método *toString*, que não recebe parâmetros e retorna um objeto da classe String na seguinte forma:

```
Nome: <nome da pessoa>
Data de Nascimento: <data de nascimento da pessoa>
```

Crie a classe abstrata *PessoaIMC* que herde da classe *Pessoa* e contenha tenha os campos protegidos *peso* e *altura*, ambos do tipo *double*. O construtor desta classe deve receber como parâmetros duas *strings* e dois valores do tipo *double* e inicializar os campos *nome*, *dataNascimento*, *peso* e *altura*. A classe *PessoaIMC* deve conter os seguintes métodos:

- *public double getPeso()* que retorna o peso;
- *public double getAltura()* que retorna a altura;
- *calculaIMC()* que recebe como parâmetros dois valores do tipo *double* que são a altura e o peso e retorna um valor do tipo *double* correspondente ao IMC (Índice de Massa Corporal = peso / altura ao quadrado) calculado.
- o método abstrato *resultIMC()* que não recebe parâmetros e retorna uma instância da classe String. (o método não é implementado nesta classe - **abstrato**)
- O método *toString()* desta classe deve retornar uma string da seguinte forma:

```
Nome: <nome da pessoa>
Data de Nascimento: <sua data de nascimento>
Peso: <seu peso>
Altura: <sua altura>
```

Crie as classes *Homem* e *Mulher*, herdeiras de *PessoaIMC*. Cada uma deve implementar o método abstrato *resultIMC()* para realiza o calculo do IMC e exibe uma mensagem de resultado acordo com o valor obtido.

Para Homem:	Para Mulher:
IMC < 20.7 : Abaixo do peso ideal 20.7 < IMC < 26.4: Peso ideal IMC > 26.4 : Acima do peso ideal	IMC < 19 : Abaixo do peso ideal 19 < IMC < 25.8: Peso ideal IMC > 25.8 : Acima do peso ideal

Crie uma classe para o programa principal, com o método *main()*, que crie instâncias das classes *Homem* e *Mulher* e armazene essas instâncias em um objeto array do tipo *PessoaIMC*. O programa deve perguntar ao usuário o tamanho do ***array***, que tipo de objeto (Homem ou Mulher) deseja criar e os dados referentes a cada objeto. A leitura de dados deve ser feita através de fluxo de entrada. Após o armazenamento de todos os objetos, o programa deve ler cada posição do *array*, imprimindo os dados do objeto ali contido e calculando seu IMC.

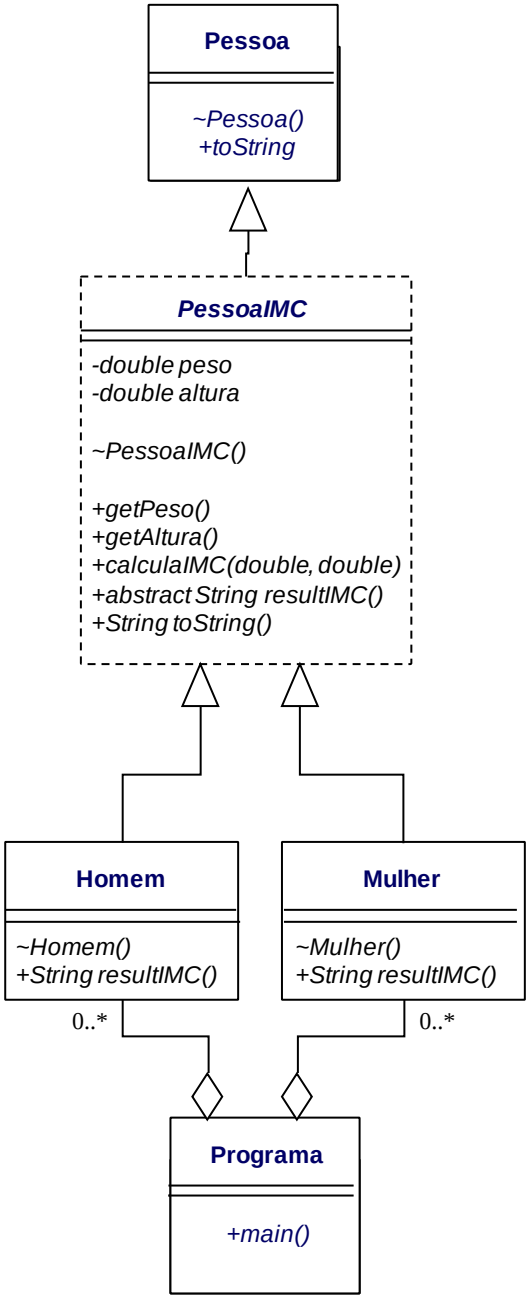

Figura4. Diagrama de classe - Ex5

Exemplo:

```
java PesoM
Digite o numero de pessoas:
2
Inserir homem (h) ou mulher(m)?
j
--- Opcao Invalida!!!
Inserir homem (h) ou mulher(m)?
h
Digite o nome:
Zezinho
Digite a data de nascimento:
01/01/1901
```

```
Digite o peso:
64.8
Digite a altura (em metros):
um m
--- A altura deve ser um numero real!!!
Digite a altura (em metros):
1.80
Inserir homem (h) ou mulher(m)?
m
Digite o nome:
Mariazinha
Digite a data de nascimento:
02/02/02/1902
Digite o peso:
64.8
Digite a altura (em metros):
1.8
--------
Nome: Zezinho
Data de Nascimento: 01/01/1901
Peso: 64.8
Altura: 1.8
IMC: 19.999999999999996  Abaixo do peso
--------
--------
Nome: Mariazinha
Data de Nascimento: 02/02/02/1902
Peso: 64.8
Altura: 1.8
IMC: 19.999999999999996  Peso ideal
--------
```