LP II - 2019.1 / 1ª Lista de Exercícios

Em todos os exercícios o aluno deve atender aos requisitos enunciados. Métodos e variáveis auxiliares podem ser criados e usados, 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

A lista deve ser entregue, deixando os arquivos com código fonte e os executáveis (.class) na máquina virtual. O professor vai avaliar o código fonte, o cumprimento dos requisitos e o programa em execução, na própria máquina.

O aluno deve criar um diretório L1, dentro do seu diretório home. Dentro de L1, criar um diretório para cada exercício com os seguintes nomes: E1, E2, E3, etc. O nome da classe com o método *main()* de cada exercício deve ser Ex1, Ex2, Ex3. Mesmo que no enunciado esteja sendo solicitado outro nome (mude o nome solicitado por estes). A correção vai ser, o máximo possível, automatizada.

Todos os exercícios devem pertencer ao pacote *default*. O aluno não deve colocar as classes desenvolvidas dentro de pacotes. Isso só deve ser feito quando explicitamente solicitado no exercício.

Para não termos problemas, não use caracteres acentuados ou cecedilha, nem no código nem nos comentários.

Para todos os exercícios devem ser tratadas as exceções numéricas, de conversão, número de argumentos, de input/output, etc., além das exceções discutidas no enunciado. Se você fizer rotins modulares, pode reusar.

"Tratamento" de exceções no estilo abaixo não serão consideradas muito boas, muito menos o *throws* sem razão (ou melhor, para se livrar das exceções):

```
try{
    // codigo sem tratamento
}catch (Exception e)
{
    System.out.println("Erro");
}
```

Exercício 1:

Fazer um programa que calcule a área de 3 tipos de figuras: círculos, retângulos e triângulos dependendo da entrada recebida, e imprima seus valores de acordo.

- Caso a figura seja um círculo é passado o raio do circulo;
- Caso a figura seja um retângulo é passado base e altura;
- Caso a figura seja um triângulo, o comprimento dos lados do Triângulo será passado.

Os valores devem ser passados como argumentos de linha de comando, valores com tipo *real.*

Você vai identificar o tipo de figura com base no número de argumentos passados.

O cálculo da área deve ser feito em um método de classe, chamado *calcula* cujas assinaturas são dadas a seguir.

```
private static float calcula(float r)
private static float calcula(float b, float a)
private static float calcula(float l1, floatl2, floatl3)
```

Chame estes métodos para fazer os cálculos (tornando o programa mais modular).

No caso de ser um triângulo, classifique também se ele é: equilátero, isósceles ou escaleno. Para isso crie outro método de classe e use o mesmo.

Utilize técnicas de criticas de dados e/ou tratamento de exceções para capturar problemas de entrada: número insuficiente de argumentos, número de excessivo de argumentos, argumentos que não sejam convertíveis para float, argumento(s) inválido(s), argumentos que não formam um triângulo.

Exemplos de execução (arredonde o número de casas decimais):

```
>java Ex1 1.3
A area do circulo e': XXX unidades de area.
>java Ex1 3.0 4.0 5.0
A area do triangulo e': XXX unidades de area.
O triangulo e' isosceles.
>java Ex1
Número de argumentos insuficiente
>java Ex1 0 1 3.7 9.5
Numero de argumentos excessivo
>java Ex1 a 0 0xD
1o argumento, "a", nao eh numero
3o argumento, "0xD", nao eh numero
```

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

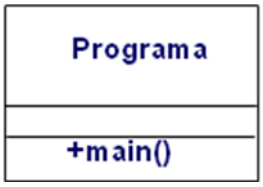

Figura 1. Diagrama de classe - Ex1

Exercício 2:

Crie uma classe *Angulo* que deverá ter os seguintes **métodos de classe**:

- `converteAngulo` que recebe como parâmetro um valor do tipo *double* que é a medida em graus de um ângulo e retorna um valor do tipo *double* que é a medida deste ângulo em radianos.
- `funcaoSeno` que recebe como parâmetro um ângulo e retorna um valor do tipo *double* que é o seno deste ângulo.
- `funcaoCoseno` que recebe como parâmetro um ângulo e retorna um valor do tipo *double* que é o coseno deste ângulo.
- `funcaoTangente` que recebe como parâmetro um ângulo e retorna um valor do tipo *double* que é a tangente deste ângulo.
- `funcaoCotangente` que recebe como parâmetro um ângulo e retorna um valor do tipo *double* que é a cotangente deste ângulo.

Para implementar os métodos anteriores, os métodos da classe *Math* podem ser embrulhados (*wrapped*). Observem que todos eles recebem um parâmetro *double* (tipo primitivo) e retornam como resultado um valor *double*.

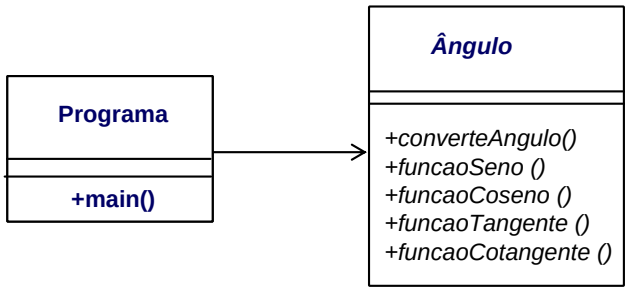

Figura 2. Diagrama de classe

Crie uma classe para o programa principal, com o método *main()* que:

- aceite como argumento da linha de comando a medida em graus de um ângulo, e utiliza os métodos da classe *Angulo* para convertê-lo para radianos e calcular o valor de suas funções trigonométricas, imprimindo estes valores. Arredonde a saída.
- havendo ou não um parâmetro de entrada na linha de comando, ao se iniciar a execução do programa (ou seja, depois de executar a rotina com o valor passado na linha de comando), também leia, através de um *fluxo de entrada*, a medida em graus de um ângulo, e utilize os métodos da classe *Angulo* para convertê-lo para radianos e calcular o valor de suas funções trigonométricas, imprimindo estes valores.
- logo após o usuário pode entrar com um novo ângulo. A entrada de uma *String* vazia encerra a leitura de valores e a aplicação.
- Trate as exceções de entrada (exceções de E/S, de conversão e passagem de argumentos inválidos).

Exemplo de saída (veja como o não arredondamento pode ficar “desnecessário”):

```
>java Ex1 60
Seno: 0.87
Cosseno: 0.50
Tangente: 1.73
Cotangente: 0.58

Digite uma medida em graus do angulo:
90
Seno: 1.0
Cosseno: 6.123233995736766E-17 [arredondar]!
Tangente: 1.633123935319537E16
Cotangente: 6.123233995736766E-17

Digite a medida em graus do angulo:
>
```

Exercício 3:

Crie a classe *AnguloObj*, que tem papel semelhante a da classe *Angulo* do exercício anterior com as seguintes modificações (o objetivo é comparar os dois estilos de arquitetura):

- A classe possui o campo privativo (encapsulado) *arcoRad* que é a medida em radianos de um ângulo.
- A classe deverá ter um construtor que recebe um valor do tipo *double*, que é a medida de um ângulo em graus, e o converte para radianos, e armazena no campo *arcoRad*.
- Seus métodos (os mesmos listados para a classe *Angulo*) agora devem ser **métodos de instância,** e não recebem parâmetros (obs: não recebem parâmetros, neste exercício – “não receber parâmetros” não caracteriza métodos de instância), mas devolvem um *double*.
- A classe *anguloObj* também implementa o método *toString()* que retorna uma instância da classe *String* na seguinte forma:

```
Arco: <medida em radianos do ângulo> rad
Cosseno: <valor>
Tangente: <valor>
Cotangente: <valor>
```

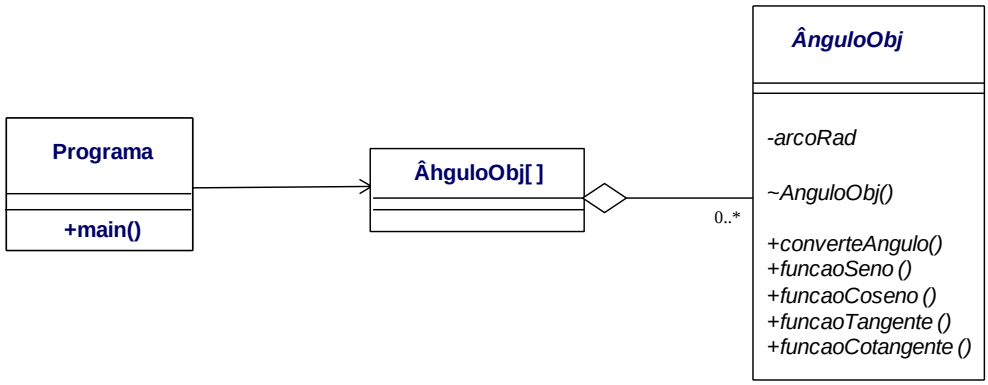

Figura3. Diagrama de classe

O programa principal deve perguntar quantos ângulos o usuário quer digitar por *stream*, e criar um *array* de objetos da classe *AnguloObj*, onde serrão armazenados objetos criados. Criar instâncias da classe *AnguloObj*, (número de objetos foi passado pelo usuário) lendo do teclado via *stream* a medida dos ângulos, e armazenar cada um no *array*. Você vai ter que desenvolver uma rotina de entrada para isso.

Em seguida (depois de ler todos os ângulos), iterar pelo *array*, e calcular suas funções trigonométricas e exibir o resultado (se você for "esperto" isso fica bem simples).

As exceções de entrada devem ser tratadas convenientemente.

Exemplo (veja como não arredondar a saída exibida fica ruim):

```
java Ex3
Digite o numero de angulos:
2

Digite a medida em graus do primeiro angulo:
90
Digite a medida em graus do segundo angulo:
60

Resultado ====================
Arco: 1.5707963267948966
Seno: 1.0
Cosseno: 6.123233995736766E-17 [arredondar]!
Tangente: 1.633123935319537E16
Cotangente: 6.123233995736766E-17

Arco: 1.0471975511965976
Seno: 0.8660254037844386
Cosseno: 0.5000000000000001
Tangente: 1.7320508075688767
Cotangente: 0.577350269189626
```

Exercício 4:

Uma empresa contratou novos funcionários 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 *Funcionario* com campos *String* para o nome e o código do empregado e campos do tipo *float* para o salário e o salário-líquido e os métodos:

- Construtor que recebe como parâmetros dois objetos *String* e um valor *float*, e inicializa os campos referentes a nome, código do empregado, salário e salário-líquido (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 *FuncionarioContratado* que estende a classe *Funcionario* e que possua, além dos campos herdados, um campo do tipo *int* para o número de dependentes, um campo do tipo *float* 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 *float* 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.

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 ou *geters*.

c) Crie uma classe *Ex4* para o programa principal que crie instâncias da classe *FuncionarioContratado* (recebendo os dados via fluxo de entrada). Estes devem ser armazenados num objeto que implemente a interface *Collection* (ArrayList, Vector, etc.).

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

O método *main* deve fazer o seguinte:

1) pedir ao usuário o número de objetos da classe *EmpregadoContratado* para os quais o salário vai ser calculado;
2) criar uma instância da classe *Ex4* (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 *Ex4* deve receber como argumento o número lido anteriormente;
3) invocar o método *calculaSalarios* no objeto *Ex4.*

O método *calculaSalarios*, na realidade faz 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 *FuncionarioContratado*, criar a instância e armazená-lo no *objeto com interface Collection*. Após ler os dados do número adequado de objetos, a aplicação faz os cálculos e exibe o resultado. Como alternativa, pode usar o <ENTER> "vazio".

Exemplo:

```
--- Cadastro de Funcionarios
Nome do Empregado: Rubens Andrade
Codigo: Emp01
Salario:500
Numero de Dependentes: 0

Nome do Empregado: Paulo Ricardo
Codigo: Emp02
Salario: 500
Numero de Dependentes: 2


--- Folha Salarial ---
Nome: Rubens Andrade
Codigo: Emp01
Salario-Base: 500.0
Salario-Liquido: 425.0
----
Nome: Paulo Ricardo
Codigo: Emp02
Salario-Base: 519.16
Salario-Liquido: 441.28599999999994
----
```

Montar o diagrama UML.

Exercício 5: (não será pontuado – mas você deve fazer)

O IMC de uma pessoa pode ser calculado através de uma fórmula que consiste em dividir o peso (em kg) pelo quadrado da altura (em m²). O número resultante é então verificado em uma escala que varia de acordo com o seu gênero e então se chega a um resultado. Sua tarefa, neste exercício, é realizar o cálculo do IMC usando dados fornecidos pelo(a) usuário(a) e analisar a escala a fim de mostrar se ele(a) está acima, na média ou abaixo do peso ideal

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 - ele é **abstrato**)
- O método *toString()* desta classe deve retornar uma string da seguinte forma (um bom lugar para você exercer o reuso de código por herança):

```
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 classificar o calculo do IMC (o cálculo efetivo é realizado pelo método *calculaIMC*). O método *toString()* retorna um objeto da classe *String* com o resultado acordo com o valor obtido e todos as demais informações da pessoa.

A classificação do IMC, indicando se a pessoa está abaixo, acima ou com o peso ideal é dada pela tabela abaixo:

Para Homem:	Para Mulher:
IMC < 20.7 : Abaixo do peso ideal	IMC < 19 : Abaixo do peso ideal
20.7 ≤ IMC ≤ 26.4: Peso ideal	19 ≤ IMC ≤ 25.8: Peso ideal
IMC > 26.4 : Acima do 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 da classe *Vector (*ou algum outro objeto do *framework Collection – ArrayList, por exemplo)*. O programa deve perguntar ao usuário o número de pessoas, 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 elemento do *Vector*, imprimindo os dados do objeto ali contido, com todos os dados.

Não usar GENERICS “<tipo>”

Erros de entrada de dados devem ser criticados e tratados quando possível (por exemplo, solicitando nova entrada).

Exemplo:

```
java Ex4
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: 19.99  Abaixo do peso
--------
--------
Nome: Mariazinha
Data de Nascimento: 02/02/02/1902
Peso: 64.8
Altura: 1.8
IMC: 19.99  Peso ideal
--------
```

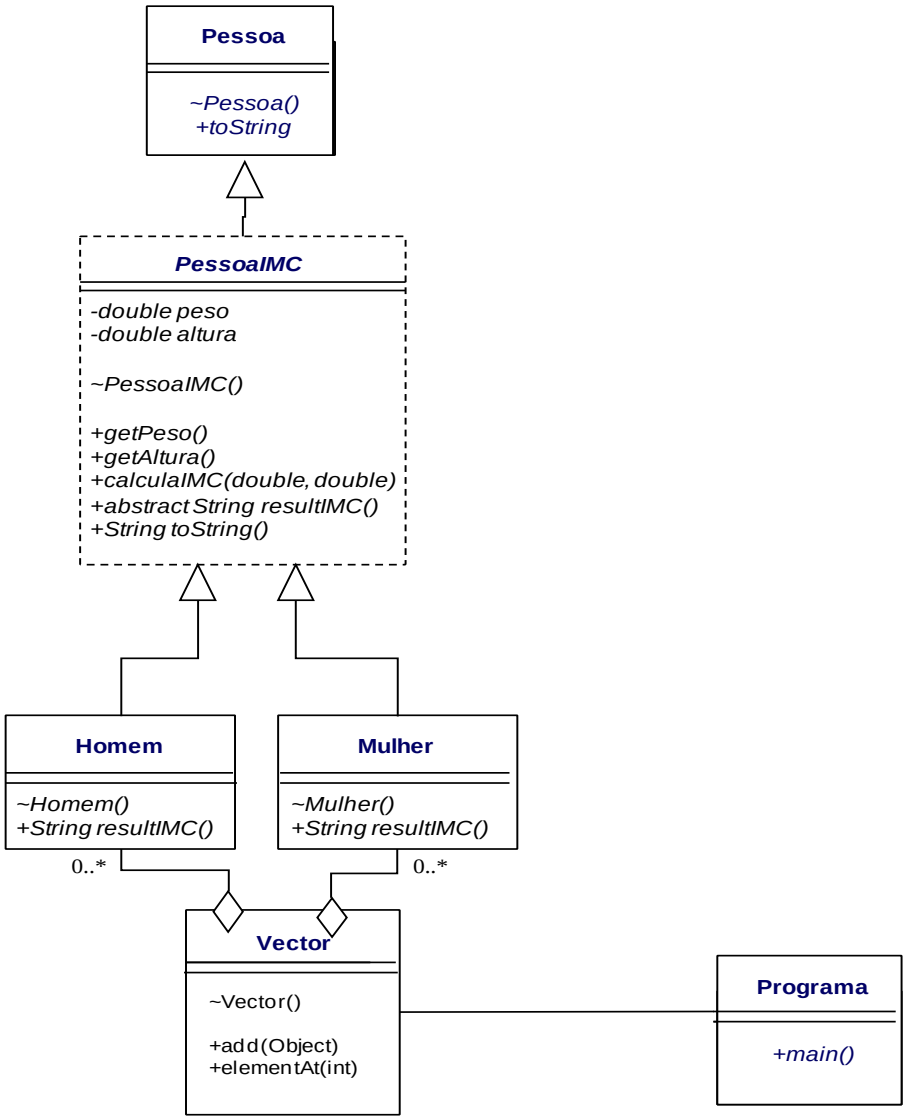

Figura4. Diagrama de classe - Ex5