LP II - 2019.2 / Lista 0 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 L0, 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.

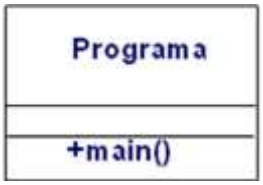

Figura 1. Diagrama de classe - Ex1

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*.

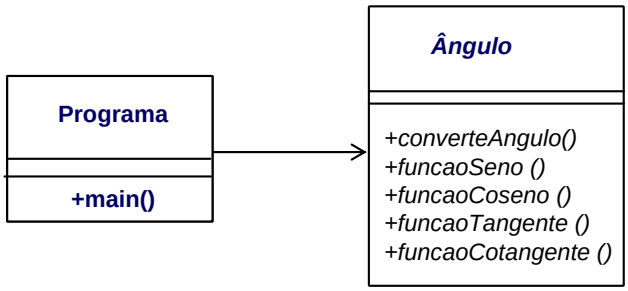

Figura 2. Diagrama de classe

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>
```

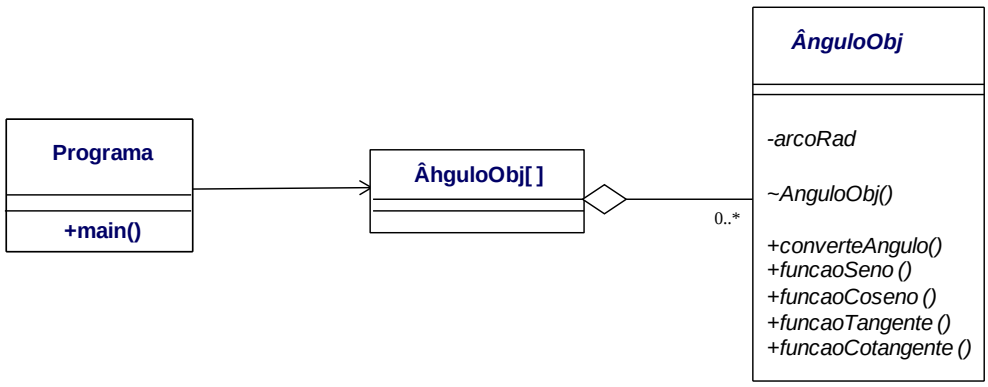

Figura3. Diagrama de classe

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
```

4º) 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*;
- um construtor que inicialize todos os campos da classe.

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 objeto da classe *ArrayList* que vai colecionar objetos 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 "principal" (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, 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.

Exemplo:

```
Digite o tamanho da matriz:
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.
```