LP II - 2015.1 / 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

A lista deve ser entregue, deixando os arquivos com código fonte e os executáveis (class) na máquina virtual. O professor avaliar o código fonte, 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. Exemplo: 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 um 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, nem no código nem nos comentários.

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 for um círculo é passado o raio do circulo
- Caso a figura for um retângulo é passado base e altura
- Caso for um triângulo o comprimento dos lados do Triângulo serão passadas

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.

```
private static real calcula(real r)
private static real calcula(real b, double a)
private static real calcula(real l1, real l2, real l3)
```

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

No caso de ser um triângulo, classifique se ele é: equilátero, isósceles ou escaleno. Para isso crie um outro método estático 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 real [argumento(s) inválido(s)], argumentos que não formam um triângulo.

Exemplos de execução:

```
>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 área.
O triangulo e' isosceles.

>java Ex1
Número de argumentos insuficiente

>java Ex1 0 1 3.7 9.5
Número de argumentos excessivo

>java Ex1 a 0 0xD
1o argumento, "a", nao é numero
3o argumento, "0xD", nao é 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.

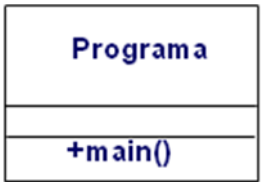

Figura 1. Diagrama de classe - Ex1

Exercício 2:

Crie uma classe *Angulo* que deverá ter o 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*.

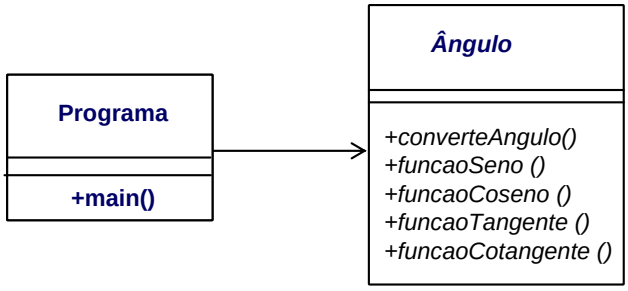

Figura 2. Diagrama de classe

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.
- havendo ou não um parâmetro de entrada na linha de comando, ao se iniciar a execução do programa, também leia através de um *fluxo de entrada* 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.
- 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).

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

Digite uma medida em graus do angulo:
90
Seno: 1.0
Coseno: 6.123233995736766E-17
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 protegido (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).
- 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
Coseno: <valor>
Tangente: <valor>
Cotangente: <valor>
```

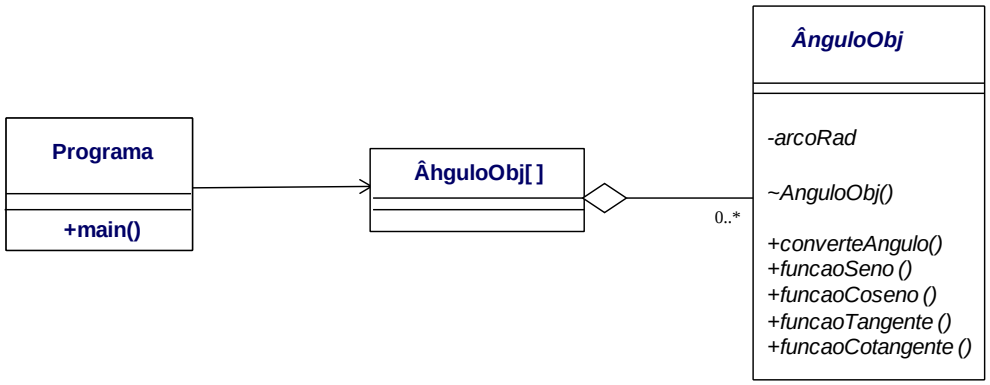

Figura3. Diagrama de classe

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*.

Em seguida, 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:

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

```
Digite a medida em graus do primeiro Ângulo:
90
Digite a medida em graus do segundo Ângulo:
60

Resultado ====================
Arco: 1.5707963267948966
Seno: 1.0
Coseno: 6.123233995736766E-17
Tangente: 1.633123935319537E16
Cotangente: 6.123233995736766E-17

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

Exercício 4 (Relógio)

Hora

Crie uma classe que contenha 3 atributos: hr, min e seg, para representar respectivamente a hora, minuto e segundo todas protegidas.

Essa classe deve definir 3 métodos:

- somaHora(int hora) que soma o valor horas à quantidade de horas
- somaMinuto(int minuto) que soma o valor horas à quantidade de horas
- somaSegundo(int segundo) que soma o valor horas à quantidade de horas

Sua classe deve estar preparada para acertar os minutos ou horas com a soma de segundos, e também para mudar as horas com a soma dos minutos. Todos os métodos devem retornar quantos dias se passaram (ou seja, a assinatura não está completa no enunciado).

Esta classe vai ter dois construtores (ver diagrama).

Data

Crie uma classe que contenha 3 campos: dia, mes, ano, todas protegidas, representando respectivamente a quantidade de dias, meses e anos. Crie também 3 métodos:

- somaDia(int dia) que soma o valor *dia* à quantidade de dias
- somaMes(int mes) que soma o valor *mes* à quantidade de meses
- somaAno(int ano) que soma o valor **ano** à quantidade de anos

Essa classe também deve estar preparada para mudar o valor dos outros campos de acordo com a necessidade, lembre-se que cada mês tem uma quantidade diferente de dias.

Esta classe vai ter dois construtores (ver diagrama).

Tempo

Crie uma classe que contenha como atributos um objeto da classe *Hora* e um objeto da classe *Data*, encapsule os objetos e disponha de um método *SomaGeral* (dia, mes, ano, hora, minuto , segundo) que usa os métodos em seus respectivos objetos, e um método que sobrescreve o *toString*() da classe pai (*Object*).

A classe Tempo contém 3 construtores, dois com parâmetros e um sem, que vai inicializar os respectivos atributos *Hora* e *Data*:

- um dos construtores recebe dois objetos, das classes *Hora* e *Data*,
- outro recebe como parâmetro os valores para construir Hora e Data no próprio construtor,
- o ultimo construtor, o vazio, inicializa os objetos, obtendo a data e hora do sistema

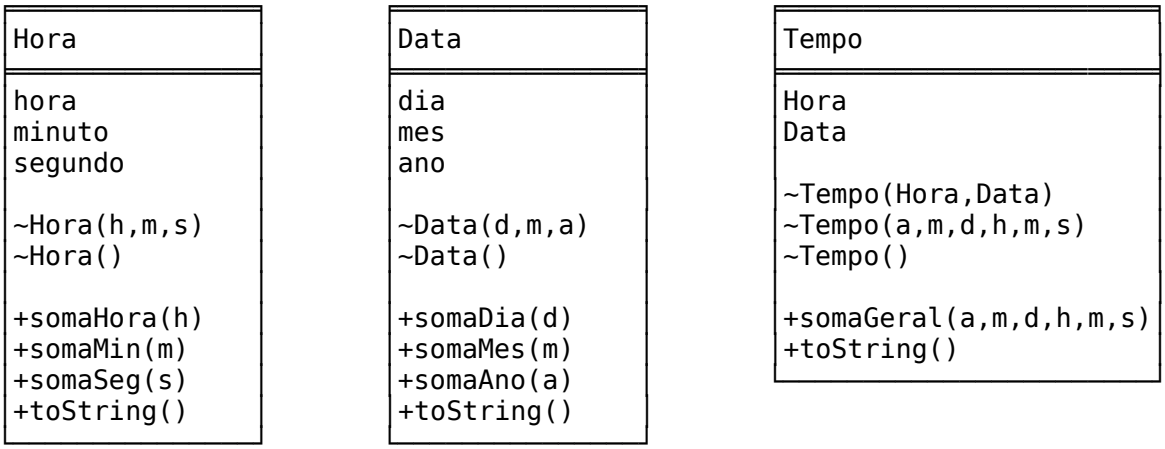

Figura4. Diagrama de classe

A classe que implementa o "programa principal" deve criar um *array* de objetos da classe *Object* de *n* elementos e salvar em cada uma dessas posições uma data, hora ou tempo. O usuário informa o valor de *n*, quantidade de elementos, e informa se quer inserir uma data, hora ou tempo.

A rotina de entrada de dados deve ser desenvolvida em um método separado.

Atenção. Deve-se alternar o uso dos construtores das classes, ex: primeiro objeto com construtor vazio passando os valores através dos métodos, segundo com construtor com parâmetros. A ordem não importa, o importante é usar todos os construtores de todas as classes.

Após receber todos os valores pelo teclado, deve-se imprimir os objetos dentro desse *array*.

Geração e Tratamento de Exceções. Cada um dos métodos, das 3 classes, construtores ou métodos de instância, que receba parâmetros referente a seg, min, hora, dia, mês, ano. Deve criticar o valor das entradas e gerar uma **exceção confirmada.** Para os parâmetros de hora, *HoraAtribEx* e para os parâmetros de data, *DataAtribEx*.

A rotina de entrada de dados vai ter que considerar estas exceções ("se livrar" delas está errado). Vai ser necessário adaptar a assinatura dos métodos que devem gerar as exceções e, obviamente quem chama um método que gera uma exceção confirmada deve tratar a mesma.

Exercício 5:

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 realizar o calculo do IMC. O método *toString()* retorna um objeto da classe String com o resultado acordo com o valor obtido e todos as demais informações da pessoa.

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*. 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.

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
--------
```

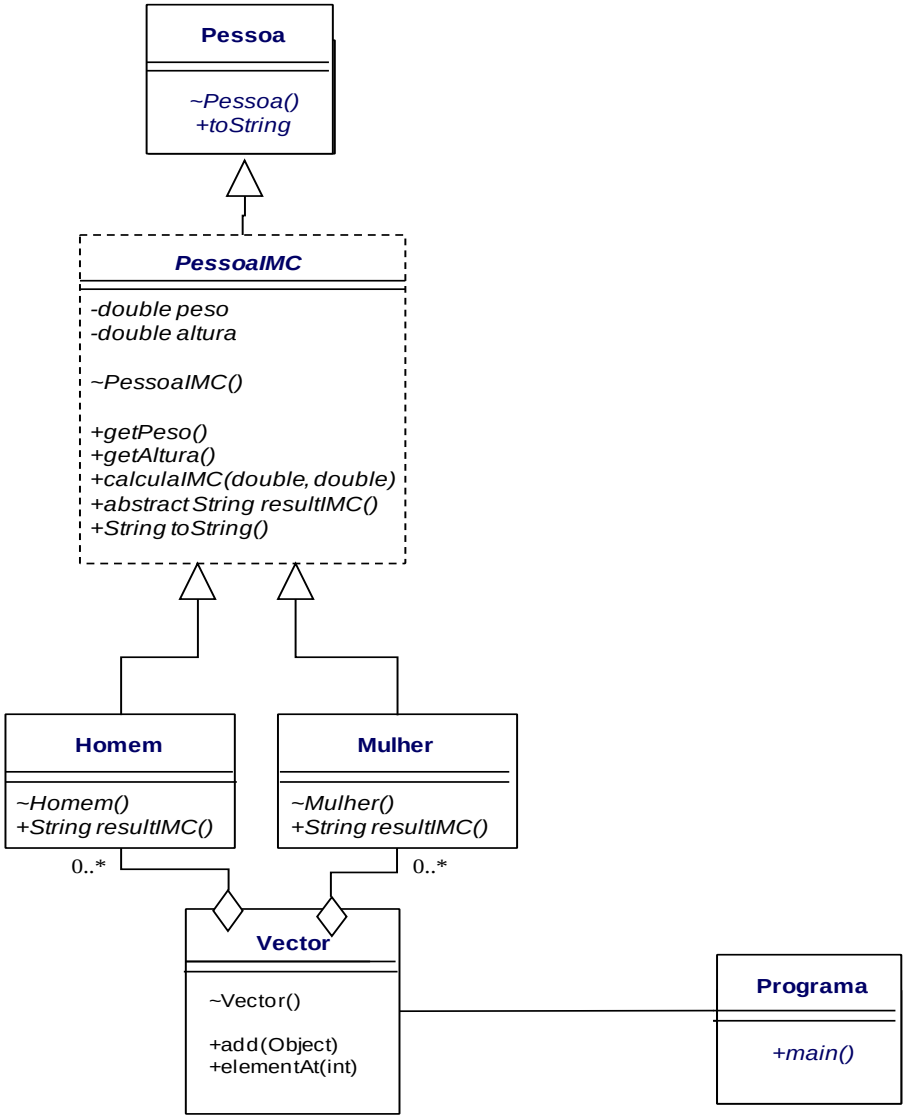

Figura4. Diagrama de classe - Ex5

Exercício 6: (bônus)

a) Crie a interface *JogoInterface*

```
public interface JogoInterface {
   public void resetar ();
   public Placar jogar ();
   public void abortar ();
}
```

b) Crie uma classe chamada *Placar*. Esta classe será usada representar o final do jogo, o resultado ou alguma outra informação que você precise passar para a estrutura de gerenciamento de jogos.

- o construtor deve receber um objeto *String*, a ser atribuído para a um campo de instância.
- um campo com que terá atribuído uma constante que indica se o jogo foi ganho ou não, com métodos *set* e *get*
- um método *toStrig()* exibindo o placar
- outros campos e métodos que você achar necessários

c) Crie uma classe chamada *MenorMaior*, que implemente *JogoInterface*, que sorteie um número aleatório **>= 1** e **<= 100**. O usuário deve tentar adivinhar qual foi o número sorteado.

O construtor deve sortear o número aleatório e guardar numa variável de instância.
O método *resetar()* pode gerar novo número aleatório
Pense no método *abortar()*
O método *jogar()* deve entrar na rotina do jogo propriamente:

Pergunta um número para o usuário, espera a sua entrada. A entrada deve ser criticada.
Ao receber o número o programa informa o número entrado tela e em seguida:

- se o usuário não acertou o número, informa se o mesmo é MAIOR ou MENOR que o número sorteado e pede uma nova entrada/tentativa de número;
- se o usuário acertar o número sorteado, o método termina e um objeto Placar deve ser criado com a informação do resultado preenchida nos seus campos. Por exemplo, uma mensagem dizendo que o usuário venceu, o numero sorteado e o número de tentativas.
- o usuário pode desistir do jogo digitando o valor 0 (zero), então o mesmo procedimento do fim de jogo deve ser usado, criando-se um objeto Placar , com uma mensagem dizendo que o usuário perdeu, o número sorteado, e o numero de tentativas, etc. e esse objeto é o retorno do método *jogar()*.

Reforçando, o método *jogar()* retorna um objeto da classe Placar, com o resultado do jogo.

Exemplo:

```
O jogo Maior Menos comecou...

Digite um número: 50
```

```
O numero fornecido eh MAIOR que o sorteado.
Digite um número: 35
O numero fornecido eh MAIOR que o sorteado.
Digite um número: 25
O numero fornecido eh MAIOR que o sorteado.
Digite um número: 15
O numero fornecido eh MAIOR que o sorteado.
Digite um número: 5
O numero fornecido eh MENOR que o sorteado.
Digite um número: 9
O numero fornecido eh MENOR que o sorteado.
Digite um número: 12
O numero fornecido eh MAIOR que o sorteado.
Digite um número: 11
```

d) Crie uma classe chamada *CaraCoroa* que implemente a interface *JogoInterface*.

O jogo você conhece. Use os requisitos da classe *MaiorMenor* como base e desenvolva classe. Adapte o que for necessário. Mas, o uso da *JogoInterface* amarra propositalmente algumas coisas. Isso serve para fazer com que todas as classes de "jogos" possam ser usadas da mesma forma por uma classe "gerenciadora de jogos".

e) Crie uma classe *Ex5 (*antes o nome era *Jogo),* classe principal do programa. A idéia é que o usuário vai executar esta classe e vai poder escolher o jogo, jogar ou abortar, jogar novamente o mesmo jogo, jogar outro jogo e terminar o programa.

Também, queremos mostrar que o uso de modularidade, interface e herança pode tornar a produção do programa mais ágil. Imagine por exemplo a criação de uma fábrica de jogos.

O método *main()* deve:
- criar uma instância de objeto da classe *Ex5* (sim!, não vamos mais usar a execução de tudo no método *main(),* estático – vamos criar um objeto mesmo)
- chamar o método *iniciar()*
- chamar o método *finalizar()*

O método *iniciar()* deve:
- perguntar ao usuário qual jogo ele deseja jogar.
- criar o objeto da classe do jogo correspondente
- chamar o método *jogar(),* passando o objeto do jogo
- fazer os laços necessários para o usuário jogar novamente
- aqui, se você for "esperto" vai preparar a estrutura de "menus" para que novos jogos sejam incluídos.

O método *jogar()* do objeto da classe *Ex5* deve:
- receber como parâmetro o objeto do jogo (ou seja, (*JogoInterface* j))
- chamar o método *jogar()* do objeto do jogo passado como parâmetro.
- com o retorno do método anterior, deve chamar o método *exibirPlacar(),* que exibe o resultado do jogo

O método *exibirPlacar(Placar p)* recebe o objeto Placar e exibe o resultado dessa "rodada"

O método *finalizar()* fecha a aplicação. Neste método, seria possível fazer uma contabilidade de quanto tempo o usuário usou o jogo, poderia totalizar alguns resultados e até pedir uma contribuição se ele gostou de usar a aplicação.

Adicione campos e métodos que você achar necessários.