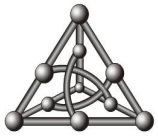
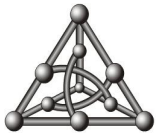


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Orientações iniciais:

- A prova é individual e sem consulta;
- A prova vale **100 pontos**, que equivalem à nota **10**;
- Distribua bem o tempo gasto em cada questão. A quantidade de pontos de cada questão corresponde ao tempo que você deveria gastar em cada uma delas;
- Apenas a resposta da questão 1 deve ser escrita na folha de papel almaço, usando caneta ou lápis.
- A prova tem duração de mínima de **40 minutos** e duração máxima de **1 hora e 40 minutos**;
- Caso você tenha alguma dúvida referente ao enunciado das questões durante a prova, pergunte em voz alta. O professor não irá atendê-lo individualmente.
- Crie uma pasta com o seu nome na sua Área de Trabalho e armazene todas as implementações da prova dentro dessa pasta. Você pode utilizar um editor de texto comum ou o Eclipse. Crie dentro dessa pasta um arquivo chamado *README.txt*, que contém um descritivo de como você organizou os exercícios (1 projeto no eclipse, vários projetos no eclipse, utilizou editor de texto comum, etc.).

BOA PROVA!



UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Prova 1

Linguagem de Programação Orientada a Objetos — 2014

Turma P02

1. **[15 pontos]** Explique sucintamente:

   (a) **[7,5 pontos]** Qual o papel do operador new? O que acontece quando você o utiliza?

   (b) **[7,5 pontos]** Vantagens e desvantagens da herança.

2. **[30 pontos]** Escreva uma classe Java chamada Pais, que representa um País (*uau!*). Um País é representado através dos atributos código ISO (cadeia de caracteres que representa universalmente o país, como por exemplo, *BRA*), nome (ex.: Brasil), população (ex.: 193.946.886) e sua dimensão em Km^2 (ex.: 8.515.767,049). Além disso, cada país mantém uma lista de outros países com os quais ele faz fronteira. Forneça os seguintes membros:

   (a) Construtor que recebe um código ISO, nome e dimensão;

   (b) Métodos de acesso (*getter/setter*) para todos os atributos;

   (c) Um método que recebe um outro País e retorna se o País recebido pertence à fronteira;

   (d) Um método que retorna a densidade populacional do País (quantidade de habitantes por Km^2);

   (e) Um método que recebe um País como parâmetro e retorna a lista de vizinhos comuns aos dois países;
   Nota: essa lista pode conter posições nulas

   (f) Atributo que armazena a quantidade de Países já criados com essa classe. Esse atributo deve ser inicializado e atualizado no momento que você achar conveniente.

   Considere que um país tem no máximo 40 outros países com os quais ele faz fronteira.

   Escreva também uma classe Java chamada TestePais, que testa de maneira satisfatória os atributos e métodos da classe Pais.

   Nota: dois Países são ditos iguais se tiverem o mesmo código ISO. Para auxiliá-lo nessa comparação, incorpore o seguinte método na classe que representa o País:

```
@Override
public boolean equals(Pais p) {
    return getCodigoIso().equals(p.getCodigoIso());
}
```

3. **[55 pontos]** Nesse exercício você deve implementar um conjunto de classes que modela os super-heróis descritos a seguir.

   Todo super-herói possui as seguintes características: nome original, descrição, data de criação, país natal, nível de resistência, estado de saúde e um conjunto de 10 habilidades. O nível de resistência de um super-herói é um valor entre 0 (morto) e 100 que indica quão forte e bem de saúde o super-herói está. Portanto, o estado de saúde de um super-herói é associado ao seu nível de resistência: nível de resistência entre 0 e 30 indica um estado de saúde ruim, nível de resistência entre 31 e 70 indica um estado de saúde bom e nível de resistência entre 71 e 100 indica um estado de saúde excelente.

   Além disso, todo super-herói tem dois comportamentos clássicos: machucar outro super-herói e ajudar outro super-herói, mas a intensidade do dano e da ajuda depende do super-herói. O homem de ferro é muito forte e egoísta, portanto, quando machuca outro super-herói, reduz em 20 unidades o nível de resistência do super-herói machucado, enquanto que, quando ajuda outro super-herói, aumenta em apenas 5 unidades o nível de resistência do super-herói ajudado. Ainda em relação ao homem de ferro, é importante armazenar a versão atual da sua armadura (número entre 1 e 10) e a quantidade de namoradas atuais.

   Já o Wolverine, apesar de ser durão, tem um coração mole. Quando machuca outro super-herói, reduz em apenas uma unidade o nível de resistência do super-herói machucado, enquanto que, quando ajuda outro super-herói, aumenta em 20 unidades o nível de resistência do super-herói ajudado. Em relação ao Wolverine, também é interessante armazenar a quantidade de adamantium no corpo (em miligramas e com duas casas decimais) e tempo médio de cicatrização (em minutos). Um pequeno detalhe é que, quando o Wolverine machuca o Homem de Ferro, ele também rouba uma de suas namoradas.

   Há também o Ciclope, que se considera um super-herói justo. Quando machuca outro super-herói, reduz em 10 unidades o nível de resistência do super-herói machucado, e quando tenta ajudar outro super-herói, aumenta em 10 unidades o nível de resistência do super-herói ajudado. Em relação ao Ciclope, sua richa pessoal é com o Wolverine: toda vez que ele o machuca, ele consegue fazer com que o tempo médio de cicatrização do Wolverine aumente em 5 minutos. Em relação ao Ciclope, é interessante armazenar qual a marca do seu óculos atual, que pode ser Oakley, HB ou Rayban.

   Por mais que não seja um super-herói, o Coringa gosta muito de machucar e, de vez em nunca, ajudar super-heróis. Quando machuca outro super-herói, ele leva o nível de resistência do super-herói machucado a 1, e quando tenta ajudar outro super-herói, aumenta em apenas 1 unidade o nível de resistência do super-herói ajudado. Em relação ao Coringa, deve ser possível armazenar em qual país ele mora atualmente.

Considere os seguintes detalhes técnicos na implementação das classes acima:

(a) Modele as suas classes antes de implementá-las. Isso facilitará muito a implementação;

(b) Encapsule corretamente e de maneira coerente os atributos de todas as classes que você criar. Lembre-se de dar acesso a um atributo apenas quando isso não romper as regras de uso da classe. Lembre-se também de sempre manter os valores dos atributos consistentes;

(c) Os construtores das classes criadas devem inicializar a maior quantidade possível de atributos;

(d) Uma data deve ser representada por uma outra classe, com atributos encapsulados *dia, mês* e *ano*, que devem ser mantidos consistentes;

(e) Há um conjunto muito específico de países, que são representados por uma das seguintes siglas: *BRA* (Brasil), *USA* (Estados Unidos) e *CAN* (Canadá). Portanto, esse conjunto deve ser representado através de um Enum com um atributo *nome completo*;

(f) O estado de saúde e a marca do óculos também devem ser representados utilizando Enums;

(g) As habilidades de um super-herói são adicionadas através de um método *adicionaHabilidade*. Uma habilidade é representada por uma cadeia de caracteres;

(h) Você deve usar herança e seus recursos;

(i) Sua implementação deve ter uma e apenas uma interface;

(j) Os métodos *machucar* e *ajudar* devem ser implementados utilizando polimorfismo e devem possuir apenas um argumento: uma instância de um super-herói a ser machucado ou ajudado.

(k) Deve ser possível imprimir qualquer objeto instanciado;

(l) Só deve ser possível instanciar um super-herói se ele for ou o Homem de Ferro, ou o Wolverine ou o Ciclope;

(m) Crie uma classe *TesteSuperHeroi*, que teste de maneira satisfatória as classes criadas neste exercício.