Exercícios e exemplos

Aqui exercícios para o curso de POO. Alguns exercícios indicam o nome do projeto entre parêntesis. Recomenda-se usar esses nomes para o projeto porque exercícios mais avançados podem usar projetos de exercícios anteriores, acrescentando novas classes ou funcionalidades ao projeto. Usar os nomes sugeridos facilita as referências.

1) Hello world
Crie seu primeiro projeto java usando o Netbeans. Siga as instruções desta página.

2) Classe Pessoa (Projeto Enterprise)
Crie um novo projeto chamado **Enterprise** e implemente nele a classe **Pessoa**. Esta classe deve ter dois atributos do tipo String para armazenar o nome e sobrenome da pessoa.

- Mantenha os atributos privados e implemente getters e setters para acesso aos atributos.
- Implemente dois construtores para a classe: um parametrizado e um Copy constructor
- Acrescente o método **getFullName** para a classe. Este método deve retornar uma string com nome completo da pessoa no formato "Sobrenome, Nome".

Resolução exercício 2

3) Classe Ponto (Projeto Cartesiano)
Crie um novo projeto chamado Cartesiano e implemente nele a classe Ponto. Esta classe representa um ponto no plano cartesiano. Tem como atributos as coordenadas X e Y do ponto (double).

- Mantenha os atributos protegidos e implemente getters e setters para acesso aos atributos.
- Implemente três construtores para a classe: o construtor padrão, um parametrizado e um Copy constructor. O construtor padrão inicia o ponto na origem (0,0).
- Implemente também os seguintes métodos públicos:
  - **void setXY(double newX, double newY):** Atribui novos valores para as coordenadas X e Y.
  - **void assign(Ponto pt):** Copia o valor dos atributos de pt para os atributos do objeto.
  - **double deltaX(double vX):** Retorna o delta X para o valor informado (vX - X).
  - **double deltaY(double vY)**: Retorna o delta Y para o valor informado (vY - Y).
  - **double distance(double posX, double posY):** Calcula e retorna a distância do ponto até a posição informada pelos parâmetros. Use os métodos deltaX e deltaY para calcular a distância.
  - **void desloc(double dX, double dY)**: Desloca o ponto adicionando os valores informados às

coordenadas do ponto.
    - **void escale(double factor):** Faz o escalonamento do ponto multiplicando as coordenadas pelo fator de escalonamento informado no parâmetro.

Resolução exercício 3

4) Classe Ponto (Projeto Cartesiano)
Para a classe Ponto do exercício anterior, implemente os seguintes métodos:

- **double distance (Ponto pt):** sobrecarga do método **distance**. Retorna a distância do ponto até o ponto informado no parâmetro.
- **double distance ( ):** sobrecarga do método **distance**. Retorna a distância do ponto até a origem do plano cartesiano.
- **String toString ( ):** retorna uma representação do ponto em forma de string no formato "(x, y)".

Resolução exercício 4

5) Classe Segmento (Projeto cartesiano)
No projeto Cartesiano, acrescente a classe **Segmento**. Esta classe representa um segmento de reta através dos dois pontos nas extremidades do dito cujo. Para representá-los, use dois objetos da classe Ponto como atributos da classe Segmento. Faça também três construtores: padrão, parametrizado e clone. O construtor padrão deve iniciar o segmento com os pontos (0,0) e (0,1). O construtor parametrizado deve receber as coordenadas x e y dos dois pontos para iniciar o segmento e finalmente o construtor clone inicia o segmento copiando as coordenadas dos pontos de outro segmento já existente.

Para esta classe faça os seguintes métodos públicos:

- **void assign(Segmento sg):** Copia o valor dos atributos de sg para os atributos do objeto.
- **void desloc(double dX, double dY)**: Desloca o segmento com os valores informados.
- **void escale(double factor):** Faz o escalonamento do segmento pelo fator de escalonamento informado no parâmetro.
- **String toString ( ):** retorna uma representação do segmento em forma de string no formato "[(x1,y1), (x2,y2)]".
- **double length( ):** retorna o comprimento do segmento.
- **public boolean isValid()**: Informa se o segmento é válido ou não. Para este método, um segmento é válido se os dois pontos estão instanciados e são diferentes entre si.
- **public Ponto midPoint()**: retorna o ponto médio do segmento.

Resolução exercício 5

6) Classe Circulo (Projeto cartesiano)

Ainda no projeto Cartesiano, acrescente a classe **Circulo**. Esta classe é descendente da classe Ponto. Para esta representação, o circulo é descrito por um ponto (centro) e o raio (double). Implemente os três construtores de praxe: padrão, parametrizado e clone. Para o construtor padrão inicie o círculo com centro em (0,0) e raio 1.

Faça também os seguintes métodos públicos:

- **void assign(Circulo cl):** Copia o valor dos atributos de cl para os atributos do objeto.
- **void escale(double factor):** Faz o escalonamento do círculo pelo fator de escalonamento informado no parâmetro.
- **String toString ( ):** retorna uma representação do círculo em forma de string no formato "(cx,cy):raio".
- **public boolean isValid()**: Informa se o círculo é válido ou não. Para este método, um círculo é válido se o raio é maior que zero.
- **double perimeter( ):** retorna o perímetro do círculo.
- **double area( ):** retorna a área do círculo.

7) Classe Poligono (Projeto cartesiano)
Para o projeto Cartesiano, acrescente a classe **Poligono**. Esta classe mantém uma lista de pontos (da classe Ponto) que representam os vértices do polígono. Para este projeto, um polígono pode ter no máximo 50 vértices. O polígono é criado sem vértices e depois eles são adicionados usando o método apropriado. Para esta classe faça os seguintes métodos:

- **public boolean ptExist(Ponto pt)**: retorna true se o ponto já existe na lista de vértices ou false se não existe.
- **public boolean addVertex(Ponto pt)**: Adiciona um novo vértice no final da lista. Retorna true se o vértice foi adicionado com sucesso ou false se houve algum erro. São condições de erro:
  - O ponto já existe na lista.
  - O polígono já tem 50 vértices.
- **public boolean isValid()**: Informa se é um polígono válido ou não. Para este método, um polígono é válido se tiver pelo menos 3 vértices.
- **public double perimeter()**: retorna o perímetro do polígono. Se não é um polígono válido, deve retornar -1.
- **public Ponto geoCenter()**: retorna o ponto que é o centro geométrico do polígono. O centro geométrico pode ser calculado fazendo a média de todos os vértices.
- **public double distance(Poligono plg)**: retorna a distância entre o centro geométrico do polígono no objeto e o centro geométrico do polígono informado no parâmetro.
- **public void desloc(double dx, double dy)**: desloca o polígono com os valores informados.
- **public void escale(double factor)**: faz o escalonamento do polígono usando o fator informado.

8) Exemplo de uso de listas e interfaces.

Cartesiano2014.zip
Details Download 25 KB

exTecelagem.doc
Details Download 27 KB

Correção detalhada do exercício

Projeto para Netbeans

exercicioLocadora.doc
Details Download 26 KB

exEstacionamento.doc
Details Download 22 KB

exEscola.doc
Details Download 24 KB

Exercicio autopecas.doc
Details Download 29 KB

Exercício Lista de alunos

public Ponto geoCenter()
renata_lima May 22, 2014

Olá Professor, não estou conseguindo fazer o método geoCenter(), devo descobrir a área e depois calcular a média?

maikaobr *May 30, 2014*
Professor, também estou com duvida, de como é feito este calculo do geoCenter(), acredito que precisamos dele no distance(Poligono plg) também ... Pode ajudar ?

nitron *May 31, 2014*
Também estou com a mesma duvida, pois encontra-se a media de cada vértice e soma ou faz as somas das vértices e depois a media?

kaiio_ *May 31, 2014*
http://dan-scientia.blogspot.com.br/2009/10/centroide-de-um-poligono.html
Pelo que eu entendi lendo aqui, é preciso achar a área, e depois disso é encontrado duas coordenadas (x,y), ou seja, o geoCenter ou centróide.
E para o método geoCenter é necessário retonar um ponto, mas no caso de achar a média dos vértices (como dito no exercício) retornaria um número, e não um ponto, pelo que eu entendi.

kaiio_ *May 31, 2014*
nitron, acredito que a média seja dividindo o valor obtido no perímetro pela quantidade de ponto no array

kaiio_ *May 31, 2014*
perímetro é o método perimeter()

fperrotti *May 31, 2014*
Oi gente, o centro geométrico é o ponto médio de todos os vértices. É só fazer a média de todos os vértices.

fperrotti *May 31, 2014*
Ah sim, vértices são pontos, portanto a média dos vértices será também um ponto.

fperrotti *May 31, 2014*
O x do centro geometrico é a média do x de todos os vértices, a mesma coisa para o y.

kaiio_ *May 31, 2014*
Acho que entendi, vou tentar fazer agora

nitron *May 31, 2014*
Só queria saber se esse Ponto que for criado para o geoCenter adiciona +1 no contador ou esse ponto é provisório.

fperrotti *May 31, 2014*
Esse ponto é criado para ser retornado, não é um vértice portanto não deve ser adicionado na lista de vértices.