Lab 2: Invariantes

Este lab deve ser feito individualmente.

Submeta o seu código ao mooshak http://deei-mooshak.ualg.pt/~jvo/ usando o seu login da ualg (sem @ualg.pt). Ex: a123456

Uma submissão permanecerá *pending* até que seja solicitada a sua validação ao professor durante a aula prática. Só as submissões *final* serão consideradas para avaliação.

A submissão deverá ser feita até

21 de fevereiro 2025

A validação poderá ser feita posteriormente, se necessário, até 7 de março de 2025

NB: Nos problemas seguintes, sempre que necessário, considera-se que dois double *d* e *g* são iguais se $|d-g|<1e-9$

Consulte agora os slides da Teóricas 2 e 3 antes de responder aos problemas seguintes.

Problema D: Círculo

Um círculo define-se por um Ponto e um raio. A tarefa deste problema é desenvolver a classe Circulo, que recebe um ponto do primeiro quadrante e um raio *double* maior do que zero. Todo o círculo deve estar no primeiro quadrante, sendo necessário testar a condição invariante de instância.

Dado um objeto da classe Circulo, é necessário calcular o seu perímetro.

Entrada
A entrada é composta por duas linhas.
A primeira linha contém dois números inteiros X_C, Y_C, que representam as coordenadas cartesianas no centro do círculo.
A segunda linha é um *double* que representa o raio do círculo

Saída
Uma linha com uma das opções seguintes:

i) "Ponto:vi"; mensagem indicando que um ponto não é do primeiro quadrante;
ii) "Circulo:vi"; indicação de que o círculo não está todo no primeiro quadrante;
iii) A parte inteira do perímetro do círculo, se existir.

A saída tem uma única linha, que contém **a parte inteira** da distância entre os dois pontos dados.

Exemplo de Entrada 1
2 2
1.0
Exemplo de Saída 1
6

Exemplo de Entrada 2
-1 0
1.0
Exemplo de Saída 2
Ponto:vi

Exemplo de Entrada 3
1 1
2.0
Exemplo de Saída 3
Circulo:vi

Restrições complementares
O programa deve completar a classe Ponto definida nos problemas anteriores, acrescentando um novo construtor que recebe dois inteiros correspondentes às coordenadas cartesianas (x,y). Naturalmente que a classe Ponto deve assegurar a consistência entre coordenadas polares e Cartesianas.

Nos comentários indique
- i) Uma linha com a descrição da responsabilidade da classe ou do que o método faz
- ii) @author (apenas para classes)
- iii) @version (apenas para classes; inclua uma data)
- iv) @inv (apenas para classes; inclua uma descrição da invariante usada)
- v) @param (apenas para métodos e construtores)
- vi) @return (apenas para métodos)
- vii) @see (qualquer referência bibliográfica ou sítio web consultado para o desenvolvimento do código respetivo)

Problema E: Interseção entre um Círculo e um Segmento de reta

Desenvolva um programa que dado um círculo, conforme definido no problema anterior e um segmento de reta definido por dois pontos em coordenadas cartesianas, determine se o segmento interseta ou não o círculo; ou indique se existiu alguma violação da condição invariante de instância em alguma das classes.

Entrada
A entrada é composta por duas linhas.
A primeira linha contém dois números inteiros X_C, Y_C, que representam as coordenadas cartesianas no centro do círculo e um *double* que representa o raio do círculo.
A segunda linha contém quatro números inteiros que representam, respetivamente, as coordenadas (x,y) do primeiro e do segundo ponto do segmento: X_1 Y_1 X_2, Y_2

Saída
Uma linha com uma das opções seguintes:

i) “Ponto:vi”; mensagem indicando que um ponto não é do primeiro quadrante;
ii) “Segmento:vi”; indicação de que dois pontos não formam um segmento de reta;
iii) “Circulo:vi”; indicação de que o círculo não está todo no primeiro quadrante;
iv) 1, se houver alguma interceção; 0 caso contário.

Exemplo de Entrada 1
2 2 1.0
1 3 3 1
Exemplo de Saída 1
1

Exemplo de Entrada 2
2 2 1.0
1 4 3 3
Exemplo de Saída 2
0

Exemplo de Entrada 3
-1 0 1.0
1 3 3 1
Exemplo de Saída 3
Ponto:vi

Exemplo de Entrada 4
1 1 2.0
1 3 3 1
Exemplo de Saída 4
Circulo:vi

Restrições complementares
Seguindo os princípios da programação orientada por objetos, o programa deverá estar estruturado segundo um conjunto de classes apropriadas.

Nos comentários indique

i) Uma linha com a descrição da responsabilidade da classe ou do que o método faz
ii) @author (apenas para classes)
iii) @version (apenas para classes; inclua uma data)
iv) @inv (apenas para classes; inclua uma descrição da invariante usada)
v) @param (apenas para métodos e construtores)
vi) @return (apenas para métodos)
vii) @see (qualquer referência bibliográfica ou sitio web consultado para o desenvolvimento do código respetivo)