jvo, hdaniel, jbarateiro

Lab 4: Figuras Geométricas

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 ao problema permanecerá *pending* até que seja validada pelo professor durante a aula prática. Só as submissões *final* serão consideradas para avaliação.

Todas as submissões deverão ser feitas até

23 de março 2025

A validação poderá ser feita posteriormente, se necessário, até 11 de abril de 2025

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

Neste laboratório consideram-se vários casos de Figuras Geométricas, que devem constituir uma hierarquia de classes, incluindo: polígonos, triângulos e retângulos, com as seguintes condições:

- os polígonos são definidos por uma lista ordenada de pontos (vértices), em número igual ou superior a 3;
- em todos os tipos de polígonos dois vértices consecutivos formam uma aresta;
- a última aresta do polígono é composta pela ligação entre o último e o primeiro vértice
- 3 pontos seguidos não podem ser colineares
- nenhum par de arestas se pode intersetar
- os círculos têm um raio superior a 0
- as figuras geométricas têm de estar totalmente no primeiro quadrante
- as figuras geométricas são imutáveis

Restrições complementares

Em todos os problemas deve seguir os princípios da programação orientada por objetos, estruturando o código segundo um conjunto de classes apropriadas. Para cada um dos problemas seguintes, antes de escrever qualquer código, defina um conjunto de testes unitários que reflitam o comportamento pretendido.
Comece por esboçar um diagrama de classes inicial (de análise) UML com as classes necessárias para implementar os problemas descritos. Este diagrama deve ser mostrado durante a validação, não devendo ser submetido ao Mooshak.
Adicionalmente, deve comentar o código, indicando

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)

Em todos os exercícios, o programa deve seguir a estrutura da seguinte classe Main (deve adaptar aos requisitos dos exercícios e submeter sua versão no Mooshak).

```
public class Main {

    public static String capital(String s) {
        if (s == null || s.isEmpty())
            return s;
        return s.substring(0, 1).toUpperCase() + s.substring(1).toLowerCase();
    }

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        Constructor<?> constructor;
        Class<?> cl;
        FiguraGeometrica f;
        String s;
        String[] aos;
        while (sc.hasNextLine()) {
            s = sc.nextLine();
            if (s.isEmpty())
                break;
            aos = s.split(" ", 2);
            try {
                cl = Class.forName(capital(aos[0]));
                constructor = cl.getConstructor(String.class);
                f = (FiguraGeometrica) constructor.newInstance(aos[1]);
            } catch (ClassNotFoundException cnfe) {
                System.out.println("Não foi encontrada a classe: " +
                                                        cnfe.getMessage());
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        sc.close();
    }
}
```

Consulte agora os slides das Teóricas 8 e seguintes
antes de responder ao problema seguinte.

Problema I: toString()

Pretende-se detalhar a representação em *string* de cada Figura através do método *toString()*. Assim, dependendo do tipo de Figura, o método *toString()* deve devolver as seguintes *strings*:

Poligono de N vertices: $[(x_1,y_1),\ldots(x_N,y_N)]$
Retangulo: $[(x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)]$
Triangulo: $[(x_1,y_1), (x_2,y_2), (x_3,y_3)]$
Circulo: (x,y) raio

Desenvolva um programa que recebe um conjunto de Figuras Geométricas, armazena-as numa única estrutura de dados e imprime-as usando o método *toString()* acima especificado.

Entrada
A entrada do programa corresponde a um conjunto de linhas. Cada linha tem o nome da classe que representa a figura geométrica (Poligono, Retangulo, Triangulo ou Circulo). Se for um polígono genérico, a linha tem o número natural que indica o número de vértices, seguida das coordenadas inteiras x e y de cada vértice, sempre separadas por espaços. Nos retângulos e triângulos, a linha tem as coordenadas inteiras de cada vértice. Finalmente, nos círculos a linha tem as coordenadas inteiras do centro, seguidas de um *double* que representa o raio do círculo.

Saída
Uma linha com uma das opções seguintes:

i) Ponto:vi: indicação de que os pontos não pertencem ao primeiro quadrante;
ii) "Segmento:vi"; indicação de que dois pontos não formam um segmento;
iii) "Poligono:vi"; indicação de que os *n* pontos não formam um polígono;
iv) "Triangulo:vi"; indicação de que os pontos não formam um triângulo;
v) "Retangulo:vi"; indicação de que os pontos não formam um retângulo;
vi) "Circulo:vi"; indicação de que o círculo não está todo no primeiro quadrante ou o raio não é maior do que zero;
vii) Impressão do conjunto de polígonos, um por linha, conforme a definição indicada na especificação do problema.

Exemplo de Entrada 1
Poligono 4 5 5 8 6 8 7 5 7
Triangulo 7 1 9 1 9 3
Circulo 2 2 1

Exemplo de Saída 1
Poligono de 4 vertices: [(5,5), (8,6), (8,7), (5,7)]
Triangulo: [(7,1), (9,1), (9,3)]
Circulo: (2,2) 1

Exemplo de Entrada 2
Retangulo 3 0 4 2 3 4 2 2

Exemplo de Saída 2
Retangulo:vi

Exemplo de Entrada 3
Poligono 5 5 5 6 6 7 7 5 7 2 1

Exemplo de Saída 3
Poligono:vi

Consulte agora os slides das Teóricas 8 e seguintes
antes de responder ao problema seguinte.

Problema J: Translação de Figuras Geométricas

Pretende-se aplicar movimentos de translação a cada Figura Geométrica. Para esse efeito, desenvolva um método que tem como parâmetros, o deslocamento em x d_x e o deslocamento em y d_y a aplicar à figura, significando que a figura se deve movimentar em x e em y pelas quantidades definidas em d_x e em d_y.

Entrada

Tal como no exercício anterior, a entrada começa com uma linha que tem o nome da classe que representa a figura geométrica (Poligono, Retangulo, Triangulo ou Circulo). Se for um polígono genérico, a linha tem o número natural que indica o número de vértices, seguida das coordenadas inteiras x e y de cada vértice, sempre separadas por espaços. Nos retângulos e triângulos, a linha tem as coordenadas inteiras de cada vértice. Finalmente, nos círculos a linha tem as coordenadas inteiras do centro, seguidas de um *double* que representa o raio do círculo.
A linha seguinte tem dois valores inteiros que correspondem aos deslocamentos dx e dy.

Saída

Uma linha com uma das opções seguintes:

i) Ponto:vi: indicação de que os pontos não pertencem ao primeiro quadrante;
ii) "Segmento:vi"; indicação de que dois pontos não formam uma segmento;
iii) "Poligono:vi"; indicação de que os *n* pontos não formam um polígono;
iv) "Triangulo:vi"; indicação de que os pontos não formam um triângulo;
v) "Retangulo:vi"; indicação de que os pontos não formam um retângulo;
vi) "Circulo:vi"; indicação de que o círculo não está todo no primeiro quadrante ou o raio não é maior do que zero;
i) Impressão da figura geométrica, após aplicada a translação definida.

Exemplo de Entrada 1

```
Poligono 4 1 2 5 6 8 7 12 14
-1 3
```

Exemplo de Saída 1

```
Poligono de 4 vertices: [(0,5), (4,9), (7,10), (11,17)]
```

Exemplo de Entrada 2

```
Poligono 4 1 2 5 6 8 7 12 14
-5 1
```

Exemplo de Saída 2

```
Ponto:vi
```

Consulte agora os slides das Teóricas 8 e seguintes
antes de responder ao problema seguinte.

Problema K: Colisões entre Figuras Geométricas

O objetivo deste exercício é desenvolver uma funcionalidade para a deteção de colisões entre quaisquer Figuras Geométricas. Para esse efeito, deve implementar um programa que recebe múltiplas Figuras Geométricas e as armazena numa única estrutura de dados ordenada.
O programa lê e armazena as Figuras Geométricas, sequencialmente, pela ordem da entrada. Sempre que uma nova Figura é adicionada, verifica se colide com alguma das Figuras já armazenadas. Se houver uma colisão, o programa exibe a posição da primeira Figura com a qual colidiu (começando em 0) e termina imediatamente. Se não houver colisões, o programa exibe a mensagem "Sem colisoes".
Por exemplo, considere um caso em que a entrada define 4 figuras, pela ordem F1, F2, F3, F4. Não havendo quaisquer colisões entre as Figuras F1, F2 e F3, caso a Figura F4 colida com a F1, o programa apresenta: “Colisao na posição 0” e termina, i.e., não verifica se há mais colisões. Se, em vez de colidir com F1, F4 colidisse apenas com a F3, o programa apresentaria: “Colisao na posição 2”. Caso não haja colisões, a mensagem apresentada é: “Sem colisoes”.

Entrada
A entrada do programa corresponde a um conjunto de linhas. Cada linha tem o nome da classe que representa a figura geométrica (Poligono, Retangulo, Triangulo ou Circulo). Se for um polígono genérico, a linha tem o número natural que indica o número de vértices, seguida das coordenadas inteiras x e y de cada vértice, sempre separadas por espaços. Nos retângulos e triângulos, a linha tem as coordenadas inteiras de cada vértice. Finalmente, nos círculos a linha tem as coordenadas inteiras do centro, seguidas de um *double* que representa o raio do círculo.

Saída
Uma linha com uma das opções seguintes:

i) Ponto:vi: indicação de que os pontos não pertencem ao primeiro quadrante;
ii) “Segmento:vi”; indicação de que dois pontos não formam uma segmento;
iii) “Poligono:vi”; indicação de que os *n* pontos não formam um polígono;
iv) “Triangulo:vi”; indicação de que os pontos não formam um triângulo;
v) “Retangulo:vi”; indicação de que os pontos não formam um retângulo;
vi) “Circulo:vi”; indicação de que o círculo não está todo no primeiro quadrante ou o raio não é maior do que zero;
vii) Impressão da informação sobre a colisão conforme indicado anteriormente.

Exemplo de Entrada 1
```
Poligono 4 5 5 8 6 8 7 5 7
Triangulo 7 1 9 1 9 3
Circulo 4 4 2
```

Exemplo de Saída 1
```
Colisao na posicao 0
```

Exemplo de Entrada 2

Poligono 4 5 5 8 6 8 7 5 7
Triangulo 7 1 9 1 9 3
Circulo 2 2 1

Exemplo de Saída 2

Sem colisoes

Exemplo de Entrada 3

Poligono 4 10 10 10 50 50 50 50 10
Circulo 30 30 10

Exemplo de Saída 3

Colisao na posicao 0