# 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

1 de março 2024

A validação poderá ser feita posteriormente, se necessário, até 15 de março de 2024

**NB:** Nos problemas seguintes, sempre que necessário, considera-se que dois double *d* e *g* são iguais se $|d-g|<10^{-9}$.

**NB2:** Caso utilize algum LLM ou copiloto salve o(s) prompt(s) utilizado(s) e comente os resultados obtidos. Adicione-os como comentários em seu código.

## Problema D: Polígonos

Uma lista de pontos forma um polígono simples se o número de pontos é superior a dois, não existem três pontos consecutivos colineares e nenhum par de arestas se cruza.

**NB:** A ordem acima de verificação só é importante por causa do Mooshak.

§A tarefa é desenvolver a classe `Poligono` com um único construtor que recebe uma lista ordenada de pontos do primeiro quadrante e testa a condição invariante de instância.

A classe `Poligono` não tem construtor por omissão e os seus objetos são imutáveis, i.e., uma vez criados não são modificáveis.

Dado um objeto da classe `Poligono` é necessário calcular o seu perímetro.

### Entrada

A entrada começa com um número natural, *n*, indicando o número potenciais de vértices do polígono.
Cada uma das *n* linhas seguintes tem um par de inteiros, separados por um espaço, pretendendo representar um vértice do primeiro quadrante de um polígono simples. Para formar o polígono, o

último vértice liga ao primeiro, formando a última aresta.

**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) “Reta:vi”; indicação de que dois pontos não formam uma reta;
iv) “Poligono:vi”; indicação de que os *n* pontos não formam um polígono;
v) A parte inteira do perímetro do polígono, se existir.

**Exemplo de Entrada 1**
```
4
-1 0
0 0
0 0
0 0
```

**Exemplo de Saída 1**
```
Ponto:vi
```

**Exemplo de Entrada 2**
```
3
1 1
1 1
0 0
```

**Exemplo de Saída 2**
```
Reta:vi
```

**Exemplo de Entrada 3**
```
4
1 1
3 1
2 3
5 2
```

**Exemplo de Saída 3**
```
Poligono:vi
```

**Exemplo de Entrada 4**
```
4
1 1
3 1
5 2
2 3
```

**Exemplo de Saída 4**
9

**Exemplo de Entrada 5**
```
4
1 0
2 1
3 0
2 0
```

**Exemplo de Saída 5**
Poligono:vi

**Exemplo de Entrada 6**
```
4
1 0
2 0
3 1
4 0
```

**Exemplo de Saída 6**
Poligono:vi

**Restrições complementares**
O programa deverá estar estruturado em *cinco* classes, i.e., `Ponto`, `Poligono`, `Reta` e `SegmentoReta` que implementam as funcionalidades indicadas acima e a classe `Cliente` que lê os dados de entrada, processa-os utilizando a funcionalidade das classes e escreve o resultado na consola. Cada classe deve estar implementada no seu próprio ficheiro.

Nesta fase da aprendizagem é proibida a utilização do Java 2D API.

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)

## Problema E: Polígonos e trajetórias

Uma trajetória é constituída por uma sequência de pontos do primeiro quadrante.

Escreva um programa que dada uma trajetória e um conjunto de obstáculos (representados por polígonos simples como no problema anterior) indique se a trajetória interseta ou não os obstáculos; ou indique se existiu alguma violação da condição invariante de instância em alguma das classes.

**Entrada**
A entrada começa com um número natural, *n*, indicando o número de pontos da trajetória.
Seguem-se *n* linhas cada uma das quais com um ponto da trajetória.
Segue-se outro natural, *m,* indicando o número de obstáculos (polígonos)
Seguem-se *m* conjuntos de linhas com o seguinte significado: A primeira linha de cada conjunto é um natural, digamos *k*, indicado o número de vértices de um polígono cujos vértices são dados pelas *k* linhas seguintes.

**Saída**
Uma linha com uma das opções seguintes:

i) "Ponto:vi"; mensagem indicando que um ponto não é do primeiro quadrante;
ii) "Trajetoria:vi"; mensagem indicando que não se trata de uma trajetória;
iii) "Segmento:vi"; indicação de que dois pontos não formam um segmento de reta;
iv) "Reta:vi"; indicação de que dois pontos não formam uma reta;
v) "Poligono:vi"; indicação de que os *n* pontos não formam um polígono;
vi) 1, se houver alguma interceção; 0 caso contrário.

**Exemplo de Entrada 1**

```
4
2 2
4 6
7 6
10 4
2
4
5 5
8 5
8 7
5 7
4
7 1
9 1
9 3
7 3
```

**Exemplo de Saída 1**

```
1
```

**Exemplo de Entrada 2**

```
4
2 2
4 6
7 6
10 4
1
4
7 1
9 1
9 3
7 3
```

**Exemplo de Saída 2**

```
0
```

**Exemplo de Entrada 3**

```
4
2 2
4 6
7 6
10 4
1
4
7 6
9 6
9 8
7 8
```

**Exemplo de Saída 3**

```
0
```

**Restrições complementares**

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

Nesta fase da aprendizagem é proibida a utilização da java 2D API.

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)