Lab 1: Primeiras classes e objetos

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

16 de fevereiro 2024

A validação poderá ser feita posteriormente, se necessário, até 1 de março de 2024

Objetivos

1. Inscrição no Mooshak
2. Atualização do JDK e Eclipse, se necessário
3. Desenvolvimento e validação via Mooshak de programas simples usando classes e objetos
4. Obtenção do relatório de erros do Mooshak

Inscrição no Mooshak

1. Aceda ao mooshak através do link da tutória ou http://deei-mooshak.ualg.pt/~jvo/

1.1. Para obter User/Password clique em Register [for On-line Contest]

1.1.1. Em Contest, selecione POO 2023/24

1.1.2. Em Group, selecione o seu turno prático

1.1.3. Use o seu login da ualg (sem @ualg.pt). Exemplo: o estudante como o número 123456 em Name escreve a123456 e em Email: a123456@ualg.pt

1.2. Depois de se autenticar com o seu login pessoal, pode escolher, entre outras coisas, o seguinte:

1.2.1. Problema

1.2.2. Visualizar (View) – lista o enunciado do problema selecionado

1.2.3. Perguntar (Ask) – coloque aqui as suas questões sobre o problema selecionado

1.2.4. Escolher ficheiro … Submeter (Submit) – permite fazer o upload do ficheiro com o código java ou em formato zip; mais detalhes serão dados abaixo.

1.2.5. Depois de devidamente testado, faça o upload do seu ficheiro .java, .gz, ou .zip submetendo-o.

1.2.6. Aguarde pela resposta do Mooshak e veja no Help o seu significado, se necessário.

Submissão ao Mooshak
Para submissão ao Mooshak temos duas hipóteses: ou submetemos um único ficheiro .java ou um ficheiro compactado .zip ou .gz. Estas duas hipóteses são descritas abaixo.

Hipótese ficheiro .java: o programa completo é *integralmente* incluído num único ficheiro de extensão **.java**; é esse ficheiro que é submetido ao mooshak.

O conteúdo possível desse ficheiro é:

0. opcionalmente, uma diretiva **package**;
1. zero ou mais diretivas **import**;
2. uma ou mais definições de classes, sendo que uma e só uma será pública e terá o mesmo nome do ficheiro.

Hipótese ficheiro compactado: Compacte o diretório raiz do código fonte, e todos os sub-diretorios, num arquivo .zip ou .gz; é esse ficheiro que é submetido ao mooshak.

Por defeito, no IDE Eclipse, o diretório raiz está no Workspace, num subdirectório com o nome do projeto:

…/<Workspace>/<Nome projeto>/src

Nota 1: Tem de ser compactado o diretório *src*; não apenas os ficheiros que estão lá guardados.

src.zip
ou
src.tar.gz

Nota 2: Só pode haver um ficheiro com uma função main(), mesmo que outra esteja comentada.

Avisos do Compilador
Alguns compiladores geram avisos (*warnings*) quando encontram caracteres que não conseguem identificar, ***mesmo nos comentários***. Isto sucede com caracteres acentuados como os portugueses (por vezes pode ser apenas uma questão de *codepage* diferente entre o sistema onde foi escrito o código e o sistema onde é compilado).

O Mooshak pode estar configurado para abortar a compilação não só quando o compilador encontra erros no código mas também quando o compilador gera avisos, sendo indicado **"Compilation error"**.

Por isso recomenda-se que se evite a utilização no código fonte de caracteres acentuados mesmo nos comentários (no caso geral caracteres cujo código ASCII é superior a 127).

Atualização do jdk e Eclipse, se necessário

1. Verifique a versão do jdk abrindo uma consola
   1.1. **Windows:** A partir do menu **Start -> Run**, escreva **cmd** e pressione **<enter>** de modo a abrir a linha de comandos.
   Linux: Numa consola em modo de texto ou abrindo uma consola virtual ou *xterm* no ambiente gráfico.
   MacOS: Abra um Terminal

   1.2. Na linha de comandos escreva:

```
javac -version
```

Caso necessário atualize (ou instale) o JDK. O ficheiro de instalação pode ser obtido no site da Oracle, em:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Seleccione Java Platform, Standard Edition e faça o download correspondente ao seu sistema operativo. Em Windows, após efetuado o download execute o ficheiro de instalação. Em Linux instale na linha de comandos com o gestor de pacote:

```
sudo dpkg –i jdk-21_linux-x64_bin.deb
```

Em sistemas Linux poderá também estar disponível através do gestor de pacotes de instalação, o instalador do JDK, embora possa ser uma versão não tão atualizada como a obtida no site da Oracle. Em sistemas baseados no Debian (Debian, ubuntu, etc.) a instalação do JDK livre pode ser efetuada na linha de comandos com:

```
apt-get install openjdk-21-jdk
```

A instalação do JDK da Oracle:

```
sudo add-apt-repository ppa:linuxuprising/java
sudo apt-get update
sudo apt-get install software-properties-common
sudo apt-get install oracle-java21-installer
```

Verificando a versão do JDK instalada, deve agora reportar a 21:

```
javac -version
```

Se continuar a reportar uma versão anterior, escolher a versão 19 com:

```
sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk-21/bin/java" 1
sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/lib/jvm/jdk-21/bin/javac" 1
sudo update-alternatives --config java
```

2. Para instalar o Eclipse no seu computador pode usar o gestor de pacotes do sistema operativo, ou visite:

http://www.eclipse.org/downloads/

descarregue o instalador, descompacte-o e execute-o.

3. Para atualizar o Eclipse, execute-o. Depois, vá a Help->Check for updates

Desenvolvimento e validação via Mooshak

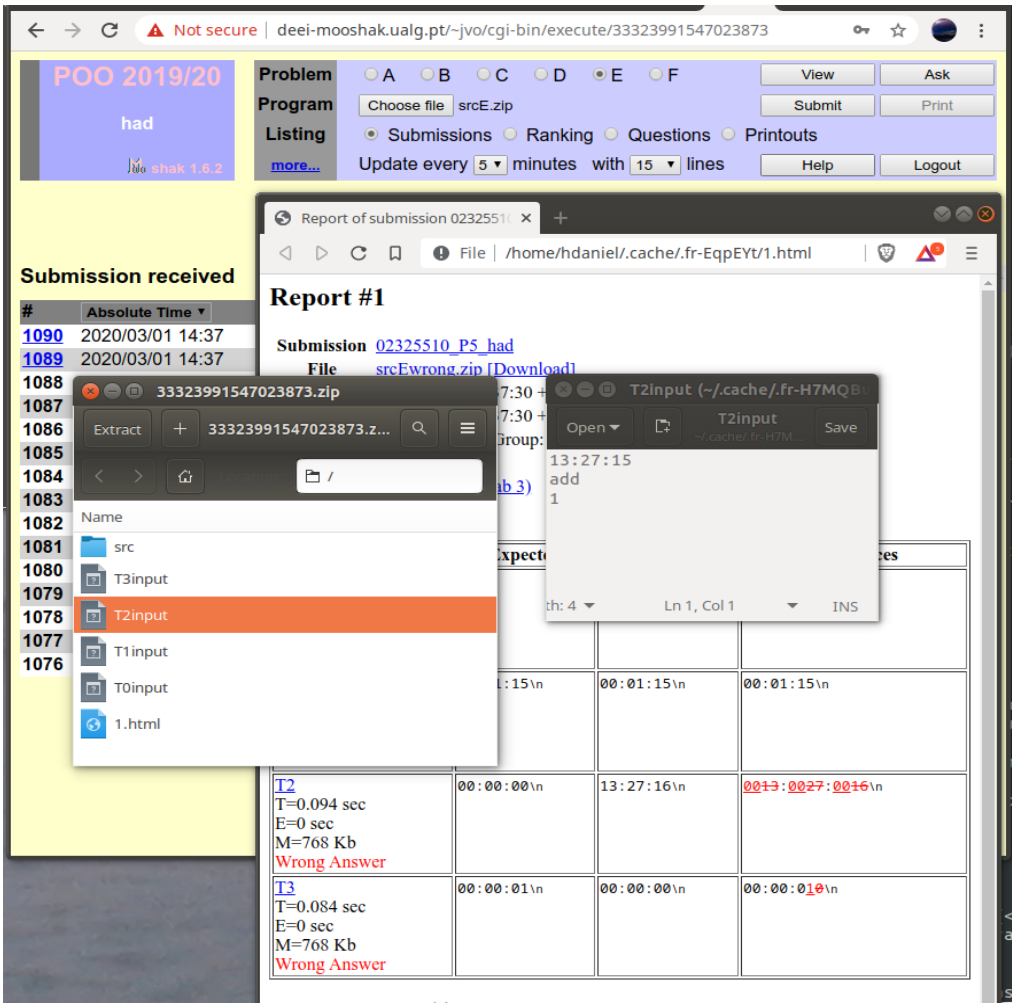
Depois de desenvolvido um programa, compacte a pasta com os ficheiros onde estão definidas as classes no código fonte com zip e submeta o zip ao Mooshak.

Veja o relatório de erros do Mooshak como descrito abaixo.

Quando tiver *Accepted* e o seu código estiver refinado, valide-o com o professor numa aula prática.

Visualização do relatório de erros do Mooshak

1. Após submissão, clique no número da submissão (o 1090, 1089,... a azul na figura abaixo) ou com o botão direito escolha **gravar como**.
2. É descarregado um ficheiro **zip**, com a submetida pasta **src**, o código, e um ficheiro **1.html**
3. Abra o ficheiro html que contém o relatório da submissão, como se pode ver na figura abaixo.
4. São também incluídos alguns ficheiros com nome **XXinput**. Cada um deles refere-se aos dados de input para cada caso de teste. No exemplo o ficheiro T2Input tem os dados de entrada para o caso de teste T2 apresentado no relatório 1.html.



Desenvolvimento de programas com as primeiras classes e objetos

Os programas resultantes das respostas aos exercícios seguintes devem ser submetidos ao Mooshak, no problema respetivo.

Problem A: Distância inteira entre 2 pontos de R^2

Desenvolva um programa que receba dois pontos e calcule a distância euclidiana entre eles. Os pontos têm coordenadas cartesianas e pertencem ao plano, i.e., as suas coordenadas podem tomar qualquer valor.

Se os pontos *A* e *B* tiverem de coordenadas (x_A, y_A) e (x_B, y_B) respetivamente, a distância euclidiana *d* entre eles é dada por:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Entrada
A entrada é composta por duas linhas.
A primeira linha contem dois números reais x_A, y_A, representando as coordenadas do ponto *A*
A segunda linha é em tudo idêntica à primeira, representando as coordenadas do ponto *B*.

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

Exemplo de Entrada
0.0 0.0
-2.0 2.0

Exemplo de Saída
2

Restrições complementares
O programa deverá estar estruturado em duas classes. A classe `Ponto` que implementa a funcionalidade básica indicada acima e a classe `Cliente` que lê os dados de entrada, processa-os utilizando a funcionalidade da classe Ponto e escreve o resultado na consola. Cada classe deve estar implementada no seu próprio ficheiro.

Dado que como a coordenadas de um ponto do plano podem tomar qualquer valor, a classe `Ponto` pode ser simplesmente dada por:

```
// File:  Ponto.java
class Ponto {
      double x, y;
      double dist (Ponto p) {
        double dx = x - p.x;
        double dy = y - p.y;
        return Math.sqrt(dx*dx + dy*dy);
      }
 }
```

Problem B: Distância inteira entre dois pontos do primeiro quadrante

Desenvolva agora um programa que, dados dois pontos pertencentes ao primeiro quadrante, calcule **a parte inteira** da distância entre eles. Uma vez que os pontos pertencem ao primeiro quadrante, as suas coordenadas têm que ser não negativas.

Restrições complementares

Tal como antes, o programa deverá estar estruturado nas classes `Ponto` e `Cliente`.

A classe Ponto deverá verificar, de cada vez que um novo objeto é criado ou sempre que seja modificado, se as suas coordenadas dos seus objetos são não negativas. Caso esta condição seja violada, o programa termina com uma mensagem de erro, conforme descrito abaixo.

Entrada

A entrada é composta por duas linhas.
A primeira linha contem dois números reais x_A, y_A, representando as coordenadas do ponto *A*
A segunda linha é em tudo idêntica à primeira, representando as coordenadas do ponto *B*.

Saída

A saída tem uma única linha, que contém **a parte inteira** da distância entre os dois pontos dados, se as suas coordenadas forem não negativas; caso contrário termina com mensagem "iv" que significa *invariant violated.*

Exemplo de Entrada 1

0.0 0.0
2.0 2.0

Exemplo de Saída 1

2

Exemplo de Entrada 2

0.0 0.0
-2.0 2.0

Exemplo de Saída 2

iv

Problem C: Path length

Task

Develop a program that takes *n* R^2 points of a path and computes the path length.
The first point received is the beginning of the path and the last the end of the path.
Path length in R^2 can be computed as sum of the Euclidean distance between points in the path, from the starting point to the second point, plus from the second point to the third point, ... until the last point. Points have Cartesian coordinates and belong to the R^2 plane, i.e., coordinates can have any real number.

Input

Input have *n* + 1 lines.
The first line is an integer that specifies the number of points in the path: *n*.
The *n* following lines contain each 2 real numbers, respectively for the X and Y coordinates of each point in the path.

Output

Output have only one line, which contains the computed path length, with 2 decimal places of precision.

To print a double, say **2.3678**, with decimal precision, say 2 decimal places, the following can be used:

```
System.out.println(String.format("%.2f", length));  //prints: 2.37
```

Additional requirements

Use the principles of object-oriented programming. In particular, use the appropriated classes to implement what is asked. Should you use any LLM or copilot save the prompt(s) used and comment on the obtained results. Add these as comments in your code.

Input sample 0

```
2
0.0 0.0
-2.0 2.0
```

Output sample 0

```
2.83
```

Input sample 1

```
3
0.0 0.0
-2.0 2.0
3.0 1.2
```

Output sample 1
7.89

Alguns aspetos práticos sobre o Mooshak
(Baseado em http://ctp.di.fct.unl.pt/~amd/cpn/2007tiup/etapa5/praticos.html)

Dados de entrada
Os dados de entrada são lidos da entrada padrão. Consistem em texto cuidadosamente formatado para ser simples de processar:

- Normalmente, nas primeiras linhas dos dados de entrada surgem alguns números inteiros que anunciam o tamanho das diversas partes do texto que se segue. Isso evita a necessidade de testar a condição de "fim-de-ficheiro", durante a leitura dos dados.
- A última linha do ficheiro está sempre devidamente terminada por uma mudança de linha.
- Espaços em branco, quando usados, são sempre considerados como separadores. Os espaços em branco nunca ocorrem em grupos. Uma linha nunca começa com um espaço em branco. Uma linha nunca acaba com um espaço em branco.

  Note que as linhas com números inteiros que ocorrem no início dos dados de entrada devem ser lidas até ao fim para evitar desalinhamentos na leitura dos dados subsequentes.

  Eis como se lê um inteiro em Java:

```
java.util.Scanner sc = new java.util.Scanner(System.in);
/* Aviso: nunca crie mais do que um Scanner sobre o input. */
int n = sc.nextInt();
```

- Supondo que os dados se iniciam por uma linha contendo dois inteiros separados por um espaço em branco:

```
java.util.Scanner sc = new java.util.Scanner(System.in);
int n = sc.nextInt();
int m = sc.nextInt();
```

NB: Em alternativa a

```
java.util.Scanner sc = new java.util.Scanner(System.in);
```

escreva apenas

```
Scanner sc = new Scanner(System.in);
```

Mas para isso terá que escrever no inicio do ficheiro de código:

```
import java.util.Scanner;
```

Dados de saída
Os dados de saída são escritos na saída padrão.

Todas as linhas de saída terminam com “\n”. Exemplo: Use

```
System.out.println("iv");
```

Para escrever na saída a String literal “iv”.

É necessário respeitar rigorosamente o formato exigido no enunciado.

Qualquer desacerto, mesmo ligeiro, é suficiente para que um programa seja classificado como "Presentation error".

Note que não é possível detetar visualmente certas anomalias nos dados de saída. Por exemplo: um espaço em branco no final duma linha, uma linha em branco no final dos dados, a omissão da mudança de linha na última linha dos dados. Todas estas situações são inaceitáveis e provocam um "Presentation error".