Lab 1: Classes e objetos

Sobre os trabalhos:

1) Os trabalhos são individuais.

2) Devem ser submetidos ao Mooshak no concurso de POO2021:

http://deei-mooshak.ualg.pt/~hdaniel

É necessário efetuar registo, no mesmo endereço.

O nome da conta deve ter o formato: aXXXXX

onde XXXXX é o número de estudante, por exemplo: a12345

O email indicado na conta deve ser o da UAlg para poderem receber a password.

3) Após submetido tem de ser validado numa aula prática, como indicado nos slides da 1ª aula teórica.

A submissão permanecerá ***pending*** até que seja solicitada a sua validação.
Só as submissões ***final*** serão consideradas para avaliação.

4) A validação poderá ser feita posteriormente à data limite da submissão, mas nesse caso não poderá voltar a ser resubmetida se for necessário alterar alguma coisa para que seja validada.
Por isso recomenda-se que a validação seja feita antes do prazo limite de submissão.

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 tutoria ou: **http://deei-mooshak.ualg.pt/~hdaniel**

1.1. Para obter User/Password clique em Register [for On-line Contest]
- 1.1.1. Em **Contest**, selecione POO 2020/21
- 1.1.2. Em **Name**, coloque o seu número de estudnate precedido de a, ex: **a12345**
- 1.1.3. Em **Email**, coloque o seu **e-mail da ualg.**
- 1.1.4. Em **Group**, selecione: **students**

1.2. Depois de autenticar-se 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 ou .zip, submetendo-o.
   1.2.6. Aguarde pela resposta do Mooshak e veja no Help o seu significado, se necessário.

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 uma mudança de linha: "\n". Exemplo:

```
System.out.println("uma string");
```

Para escrever na saída a String literal "uma string\n".
A função **println()** adiciona automaticamente o "\n" no fim da String.
A função **print()** não o faz.

É necessário respeitar rigorosamente o formato exigido no enunciado, para que o Mooshak possa faaer o teste.

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

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".

Submissão ao Mooshak
Para submissão ao Mooshak temos duas hipóteses: ou submetemos um único ficheiro .java ou um ficheiro compactado .zip ou .gz. Estes dois casos são descritos abaixo.

Caso um único 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 será pública e terá o mesmo nome do ficheiro.

Caso pasta com vários ficheiros compactada com .zip ou .gz:
Coloque todos os ficheiros do código fonte numa única pasta.
Compacte essa pasta num arquivo .zip ou .gz; é esse ficheiro que é submetido ao Mooshak.

Ao criar classes no Eclipse, todas as classes devem ser criadas no Default package, para que fiquem todos os ficheiros na pasta src. Se não for assim o Mooshak não descobre o ficheiro.

Por defeito, no IDE Eclipse, o código fonte é colocado numa pasta com o nome do projeto, no Workspace e na subpasta **src**:

…/<Workspace>/<Nome projeto>/**src**

Verificar que todos os ficheiro estão na pasta **src** e que não há subpastas com ficheiros antes de compactar.
Tem de ser compactado o diretório **src**, não apenas os ficheiros que estão lá guardados:

src.zip

ou

src.tar.gz

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 (muitas vezes é apenas uma questão de ***codepage*** diferente entre o sistema onde foi escrito o código e o sistema onde é compilado, sendo o carácter interpretado no servidor do Mooshak como outro qualquer).

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 NÃO podem ser utilizados no código fonte de caracteres acentuados, mesmo nos comentários (no caso geral caracteres cujo código ASCII é superior a 127).
Também não podem ser dados nomes aos ficheiros com caracteres portugueses.

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

Selecione 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-15.0.2_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-15-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-java15-installer
```

Verificando a versão do JDK instalada, deve agora reportar a 15:

```
javac -version
javac 15.0.2

java -version
java version "15.0.2" 2021-01-19
```

Se continuar a reportar uma versão anterior, escolher a versão 15 com:

```
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. O executável do instalador deverá ter o nome: **eclipse-inst**.
Abre-se uma janela onde deverá ser escolhida a opção: **Eclipse IDE for Java Developers**. Escolha a versão 13 do JDK previamente instalada e o diretório de instalação.

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

Desenvolvimento e validação via Mooshak de programas simples usando classes e objetos

Desenvolva o exemplo seguinte, compacte a pasta com os ficheiros onde estão definidas as classes no código fonte com zip, submeta o zip ao Mooshak.

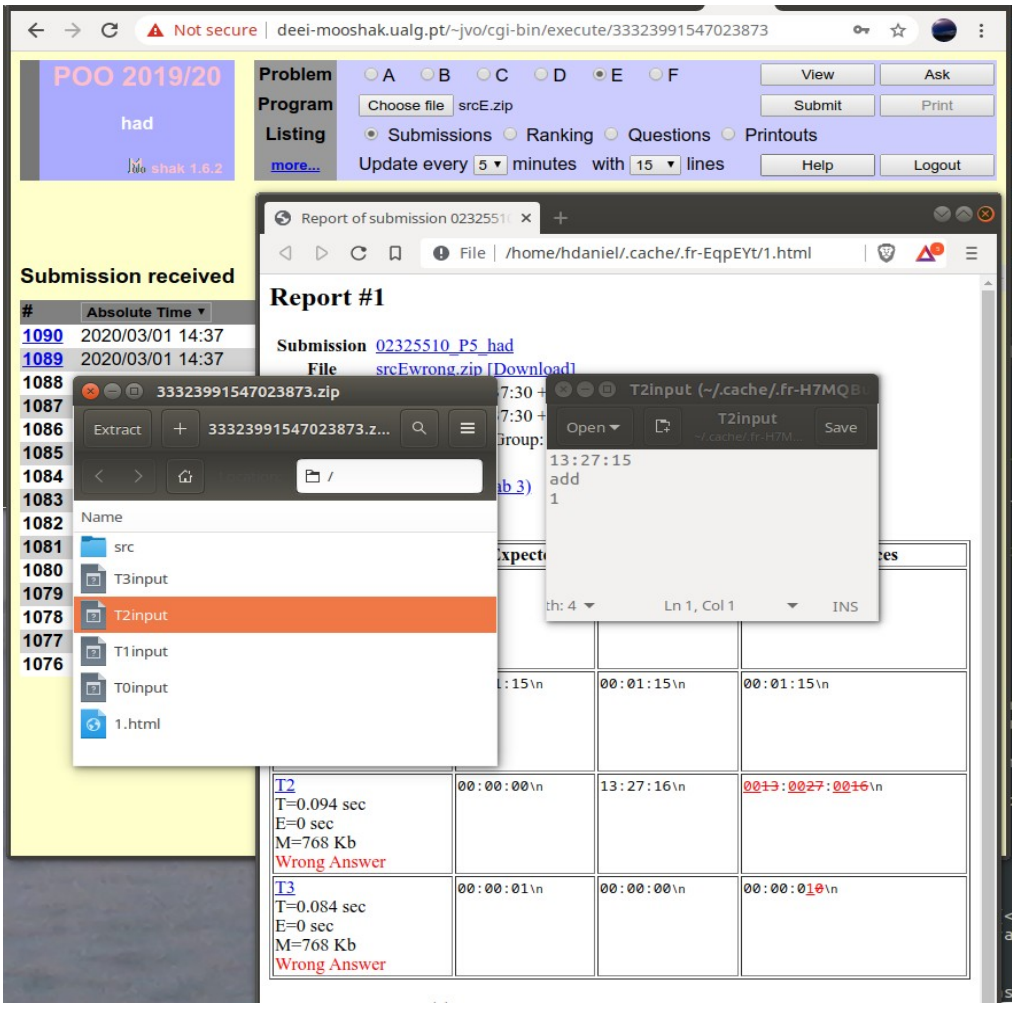
Veja o relatório de erros do Mooshak com descrito abaixo.

Verifique se foi **Accepted**. Quando for valide com o professor numa aula prática.

Visualização do relatório de erros do Mooshak

Para que funcione o código **(C ou Java)** tem de ser submetido **numa pasta compressa com zip.**

Exemplo:
1) Se essa pasta for chamada **src**, e após compressa **src.zip**
2) Após submissão, clicando no número da submissão (o 1090, 1089,... a azul na figura abaixo) ou com o botão direito escolhendo **gravar como**.
3) É descarregado um ficheiro **zip**, com a submetida pasta **src**, o código, e um ficheiro **1.html**
4) Abrindo este ficheiro está o relatório da submissão, como pode-se ver na figura.
5) 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.



Problem A: Path length

Submit your source code to Mooshak at:

http://deei-mooshak.ualg.pt/~hdaniel

up to Mar 15, 2020

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 past 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.
If point A and B have coordinates (x_A, y_A) and (x_B, y_B) respectively, the Euclidian distance *d* between them is given by:

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

Constraints:
The program must be structured in two classes: Class **Point** implements basic operations on points as defined above. Class **Client** reads input data processes them and using **Point** class operations write the result in the console.
Each class must be defined in a different file.

Since point coordinates can have any real value, **Point** class implementation can be as simple as:

```
class Point {
    private double x, y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double distance(Point p) {
        double dx = x - p.x;
        double dy = y - p.y;

        return Math.sqrt(dx*dx + dy*dy);
    }

}
```

To print a double, say **length=2.3678**, with decimal precision, say 2 decimal places, can be used the **String** class **format()** method:

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

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.

Note that Mooshak assumes that each input and output lines end always with a newline character.

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