- Faça login no computador seguindo as instruções do docente.
- No directório *Desktop* vai encontrar um conjunto de ficheiros úteis para o exame.
- Utilize o executável `./run-jar` como complemento na especificação do programa.
  Exemplo: `./run-jar p1.txt`
- Desenvolva a linguagem por forma a que os programas de exemplo da linguagem (`p?.txt`) sejam aceites.
- Pode consultar a documentação das classes `Java` usando o comando `view-javadoc`.
  Exemplo: `view-javadoc ParseTreeProperty`
- Tem à sua disposição os comandos de apoio à programação em `ANTLR4`:
  `antlr4`, `antlr4-build` (ou `build`), `antlr4-run` (ou `run`), `antlr4-clean` (ou `clean`), `antlr4-main` (ou `main`), `antlr4-test` (ou `a4-test`)
- Utilize o enunciado como rescunho, e no final **entregue-o com o cabeçalho preenchido.**
- Caso pretenda desistir deve indicar essa decisão no enunciado e executar o comando: `desisto`

Problema: Pretende-se implementar um interpretador para cálculo com números complexos. Como exemplo inicial, considere o seguinte programa:

```
*COM* p1.txt
display 2+3i;    *COM* escreve na consola o número 2+3i
display 4;       *COM* escreve na consola o número 4
display i;       *COM* escreve na consola o número i
c <= 4.3-i;      *COM* guarda o número 4.3-i na variável c
display c;       *COM* escreve na consola o número armazenado na variável c
```

Nota 1: partindo das instruções exemplificadas, tente tornar a linguagem o mais genérica possível.

Nota 2: os identificadores para variáveis contêm apenas letras e dígitos, não podendo começar com um dígito.

Nota 3: considere que as partes reais e imaginárias dos números complexos literais são números inteiros ou reais de vírgula fixa.

Nota 4: não se esqueça das verificações sintácticas e semânticas. Existem ficheiros `err?.txt` para o ajudar nesse fim.

a) Implemente em `ANTLR4`, uma gramática `LangComplex` para esta linguagem. [4 valores]

b) Implemente um interpretador que faça a verificação semântica e execute as instruções desta linguagem. [4 valores]

c) Altere a gramática e o interpretador por forma a permitir a realização das seguintes operações sobre complexos (ver programa `p2.txt`): [6 valores]

- soma/subtracção/multiplicação/divisão de números complexos (operadores $+ - * :$), com as precedências naturais.[1].
- operadores prefixos unários ($+ -$): aplicáveis a qualquer expressão complexa. Este operador deve ser prioritário relativamente a todos os anteriores.
- parêntesis: este operador serve para impor prioridades na realização de operações sobre complexos.

[1] $(a+bi) \pm (c+di) = (a \pm c) + (b \pm d)i \quad (a+bi) * (c+di) = (ac-bd) + (ad+bc)i \quad \frac{a+bi}{c+di} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$

"COM" p2.txt

```
display i-4+5i+8;
r1  <=  2;
i1  <=  i;
display r1*r1*i1;
display r1+i1:i1-r1;
display -r1;
display -i1;
c1<= 1-i;
c2<=-(c1+r1  -  i1*i);
display c2;
```

d) Acrescente ainda as seguintes operações (ver programa `p3.txt`): [6 valores]

   (a) Extracção de parte real (`re number`) e parte imaginária (`im number`) de um complexo.

   (b) Conjugado[2] de números complexos (`number*`).

   (c) Módulo[3] de números complexos (`|number|`).

Considere as seguinte prioridades entre operadores (por ordem decrescente): operadores unários prefixos, multiplicação/divisão, soma/subtracção, conjugado e, por fim, extracção das partes real/imaginária.

11 de Julho de 2019