Primeiro Trabalho de Tópicos Avançados em Linguagens de Programação
Prof. José de Oliveira Guimarães
Veja a data de entrega no AVA

Um número complexo é da forma $z = a + bi$ no qual a e b são números reais (use tipo `Double` em Cyan). Faça um programa em Cyan que faça operações sobre números complexos. A entrada é um arquivo cujo nome é passado como parâmetro ao programa. Este arquivo deve ter uma sequência de registros, cada um deles da forma

```
5.0    -3.5
-4.5   8
+
```

A primeira linha representa o número complexo $5.0 + -3.5i$ e a segunda o número $-4.5 + 8i$. Para este registro deve ser escrito em um arquivo `"r.txt"` o resultado de $(5.0 + -3.5i) + (-4.5 + 8i)$, que é $0.5 + 4.5i$. Escreva no arquivo exatamente $0.5 + 4.5i$, deixando exatamente um espaço antes e após o + e nenhum espaço antes da parte real ou depois da parte imaginária.

As operações possíveis são +, $*$ e −. O programa deve definir um protótipo para representar um número complexo. Este protótipo deve implementar como métodos as operações de soma, subtração e multiplicação de complexos (use operadores como nomes de métodos, não letras como `mais`). Além disto, ele deve ter métodos para: converter o complexo para `Double` (lançe uma exceção `ExceptionStr` se o número tem uma parte imaginária, veja no manual como se lança uma exceção), retornar a parte real, retornar a parte imaginária, retornar o número complexo em formato de uma tupla com nomes, do tipo `Tuple<real, Double, img, Double>`, retornar o módulo do complexo. Faça também métodos

```
// suponho que o protótipo chama-se "Complexo"
func equal: Complexo other -> Boolean { ... }
func |> Function<Complexo, Complexo> f -> Double {
    return f eval: self
}
```

O primeiro deve ter comparar `self` com o parâmetro usando o método `equal:` de `Double`. O objetivo é verificar se `self` e `other` são iguais até uma certa precisão.

Os seguintes métodos herdados de `Any` devem ser redefinidos: `asString`, ==, ===, `!=`, `eq:`, `neq:`. Use `override` antes de `func` na redefinição.

O protótipo que representa um complexo deve ter pelo menos três construtores: um que constrói um número complexo a partir da parte real e imaginária, outro que constrói um complexo usando a parte real e outro, sem parâmetros, que constrói o complexo $0 + 0i$.

Faça ainda um protótipo com exatamente o nome `Teste`, diferente do protótipo principal (que deve ser `Program`), que deve possuir um método `teste` que não retorna nada. Este método deve ter linhas como

```
assert Complexo(1.0, 2.0) equal: Complexo(3.0 - 2.0, 1.0 + 0.5 + 0.5);
let zeroi = Complexo(0.0);
let oneOne = Complexo(1.0, 1.0);
assert (zeroi*zeroi + Complexo(1.0, 1.0)) equal: oneOne;
```

Através da macro `assert` devem ser testados `TODOS` os métodos implementados.