Ciências ULisboa | Informática

# Introdução à Programação

Licenciatura em Engenharia Informática

## Série de Exercícios 3

2021/2022

## Abstração procedimental

**1.** Na série de exercícios anterior, vários exercícios envolviam escrever um pedaço de código que executasse determinada tarefa. Resolva esses exercícios de novo, desta vez recorrendo à definição de funções e procedimentos, sempre que isso for apropriado. Escolha criteriosamente os parâmetros das funções e procedimentos que definir, tendo em vista que o objetivo é maximizar o seu potencial de reutilização.

Em Java, funções e procedimentos são definidos através de métodos declarados como *static*. Para cada método definido deve escrever um cabeçalho *javadoc* incluindo uma descrição sucinta e geral do método e, sempre que apropriado, `@param`, `@requires`, `@return` e `@ensures`.

**2.** Considere que todos os métodos definidos no exercício anterior foram colocados numa classe `Util`. Junte a essa classe adicionalmente implementações das seguintes funções:

a. `boolean isPerfect(int n)` — determina se o número inteiro `n` é perfeito (um número é perfeito se e só se for igual à soma dos seus divisores próprios)

b. `int howManyPerfectNumbers(int n)` — calcula o número de números perfeitos menores que `n`

c. `int gcd(int m, int n)` — calcula o máximo divisor comum entre `n` e `m` recorrendo ao algoritmo de Euclides

d. `int mmc(int m, int n)` — calcula o mínimo múltiplo comum entre `n` e `m`

**3.** Programe uma classe `QuadraticEquationSolver` que resolva equações de segundo grau, ou seja, encontra as raízes de polinómios da forma

$$a.x^2+b.x+c$$

O programa deve considerar que os três coeficientes do polinómio são fornecidos no comando de execução programa como se mostra no exemplo abaixo:

```
$ java QuadraticEquationSolver 1.0 -3.0 2.0
solutions: 1.0  2.0
```

O programa deve imprimir as soluções, caso existam, ou um aviso no caso contrário. O programa deve utilizar

- o método `parseDouble` da classe `java.lang.Double`
- o método `sqrt` da classe `java.lang.Math`

A documentação destes dois métodos, mostrada abaixo, encontra-se disponível online em

https://docs.oracle.com/javase/8/docs/api/

Por exemplo a descrição de `sqrt` da classe `java.lang.Math` está em

https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#sqrt-double-

**sqrt**

```
public static double sqrt(double a)
```

Returns the correctly rounded positive square root of a `double` value. Special cases:

- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the `double` value closest to the true mathematical square root of the argument value.

**Parameters:**

`a - a value.`

**Returns:**

`the positive square root of a. If the argument is NaN or less than zero, the result is NaN.`

**parseDouble**

```
public static double parseDouble(String s)
                          throws NumberFormatException
```

Returns a new `double` initialized to the value represented by the specified `String`, as performed by the `valueOf` method of class `Double`.

**Parameters:**

`s - the string to be parsed.`

**Returns:**

`the double value represented by the string argument.`

**Throws:**

`NullPointerException - if the string is null`

`NumberFormatException - if the string does not contain a parsable double.`

**4.** O ISBN é um número com 10 dígitos que identifica um livro de forma única. O dígito mais à direita é um *checksum digit* que pode ser univocamente determinado pelos restantes 9 dígitos, utilizando a seguinte condição:

$$10*d_{10}+9*d_9+...+2*d_2+1*d_1 \quad \text{múltiplo de } 11$$

onde $d_i$ representa o *i*-ésimo dígito do número, a contar da direita. Na condição acima, $d_1$ representa o *checksum digit*. O *checksum digit* pode ser qualquer valor entre 0 e 10. A convenção do ISBN é usar o caracter 'X' para representar o 10.

Programe uma classe `IsbnGenerator` que, como se mostra no exemplo abaixo, recebe um inteiro com 9 dígitos como argumento na linha de comandos e imprime o ISBN válido com *checksum digit*:

```
$ java IsbnGenerator 020131452
ISBN: 0201314525
```

Recorra à definição de funções e/ou procedimentos para que o seu código fique mais fácil de entender, modificar e de reutilizar.

**5.** Considere o *método da bisseção* para encontrar zeros de uma função real contínua *f*:

> Dados $a < b$ com $f(a)*f(b) < 0$, calcula-se o ponto médio $c = (a+b)/2$ e o valor de $f(c)$; se f(a)*f(c) < 0, repete-se o processo com $b = c$; caso contrário, repete-se o processo com $a = c$.
>
> O processo termina quando a distância entre *a* e *b* for menor que um valor de *erro* pré-estabelecido, sendo o valor de *c* uma aproximação para o zero de *f*.

Implemente este método numa classe *Bissection*. Os valores iniciais de *a, b*, e do *erro*, devem ser variáveis do método *main*. A função *f* é definida também na classe. Por exemplo, para $f(x) = x^2$, a sua classe deve incluir o seguinte método:

```
static double f(double x){ return x*x; }
```