

# POO (LEI/LCC)

2014/2015

---

## Ficha Prática #01

---

António Nestor Ribeiro, José Creissac Campos  
{anr,jose.campos}@di.uminho.pt

## Conteúdo

|     |   |    |
|-----|---|----|
| 1   | Objectivos  | 3  |
| 2   | Introdução ao Java                                  | 3  |
| 2.1 | A plataforma Java . . . . .                         | 3  |
| 2.2 | Compilação e execução de um programa Java . . . . . | 4  |
| 2.3 | Mais sobre a linguagem . . . . .                    | 7  |
| 2.4 | A Java SE API . . . . .                             | 9  |
| 2.5 | Input/output . . . . .                              | 11 |
| 3   | Exercícios  | 14 |

## 1 Objectivos

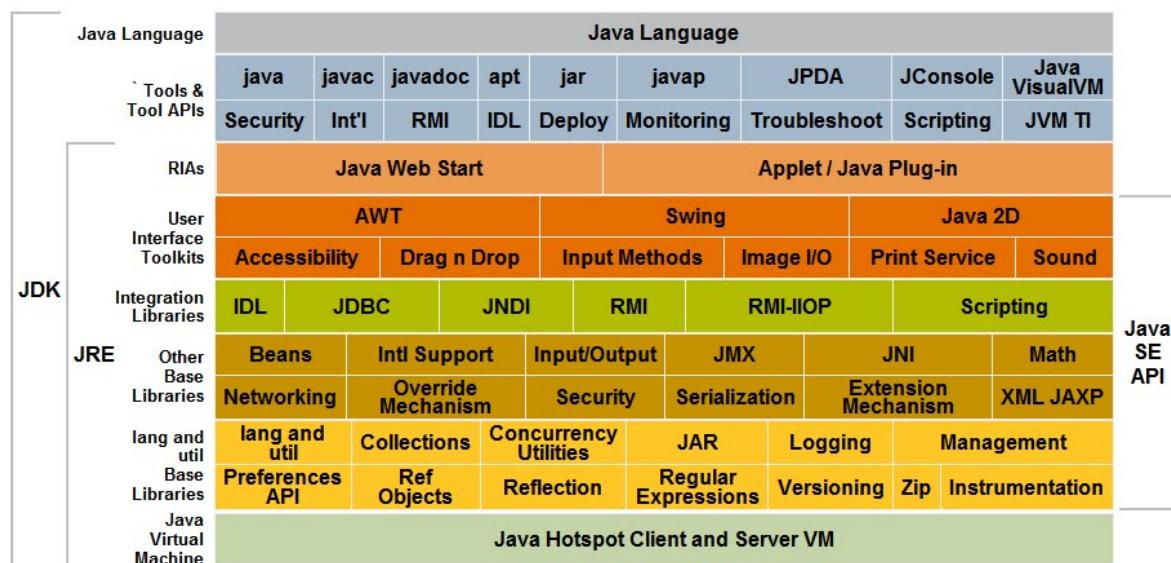
1. Conhecer o ambiente de desenvolvimento
2. Conhecer os Tipos de Dados e Estruturas de Controlo
3. Saber consultar a documentação online
4. Resolver exercícios simples

## 2 Introdução ao Java

### 2.1 A plataforma Java

- Java é uma linguagem de programação e uma plataforma computacional
- JRE (Runtime Environment): O ambiente de execução — necessário para executar software desenvolvido em Java
- JDK (Java Development Kit): O kit de desenvolvimento de software — necessário para programar em Java
  - A implementação da Java SE API que vamos utilizar é a Java JDK 7/8 da Oracle: disponível aqui<sup>1</sup>

A plataforma Java JDK 7



<sup>1</sup><http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## 2.2 Compilação e execução de um programa Java

O meu primeiro programa Java

---

```
1 public class OláMundo {  
2     /**  
3      * O meu primeiro método  
4      */  
5     public static void main(String[] args) {  
6         // Colocar código aqui!  
7         System.out.println("Olá Mundo!");  
8     }  
9 }
```

---

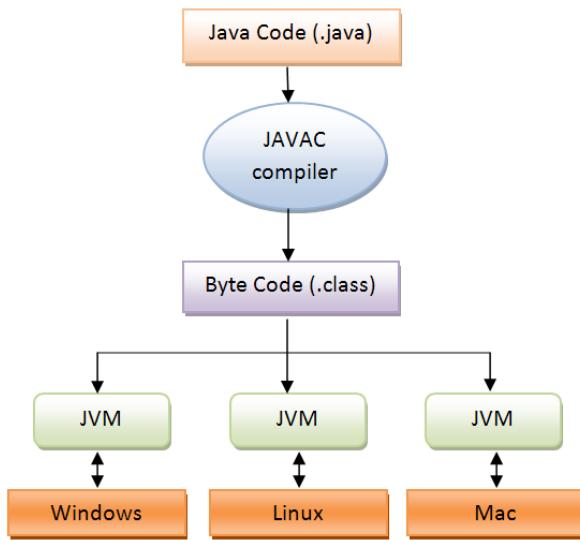
- O nome da classe deve começar por maiúscula (neste caso o nome é `OláMundo`)
- O ficheiro deve ter o mesmo nome da classe com a extensão `.java`
- Neste caso o ficheiro deve chamar-se `OláMundo.java`
- A execução do programa começa pelo método `main` (recebe um array de `String` com os parâmetros passados na linha de comando)
- Podemos definir métodos auxiliares:

```
1 public class OláAlguem {  
2     /** Método auxiliar */  
3     public static String geraSaudação(String nome) {  
4         return "Olá "+nome+"!";  
5     }  
6     /** Início do programa */  
7     public static void main(String[] args) {  
8         String saudação = OláAlguem.geraSaudação("Mundo");  
9         System.out.println(saudação);  
10    }  
11 }
```

---

Como executar o programa?

- Os programas Java são compilados para uma representação intermédia (o bytecode)
- O byte code é interpretado pela máquina virtual Java – um computador virtual



Utilizando a linha de comando

Compilação: `$javac OláMundo.java`

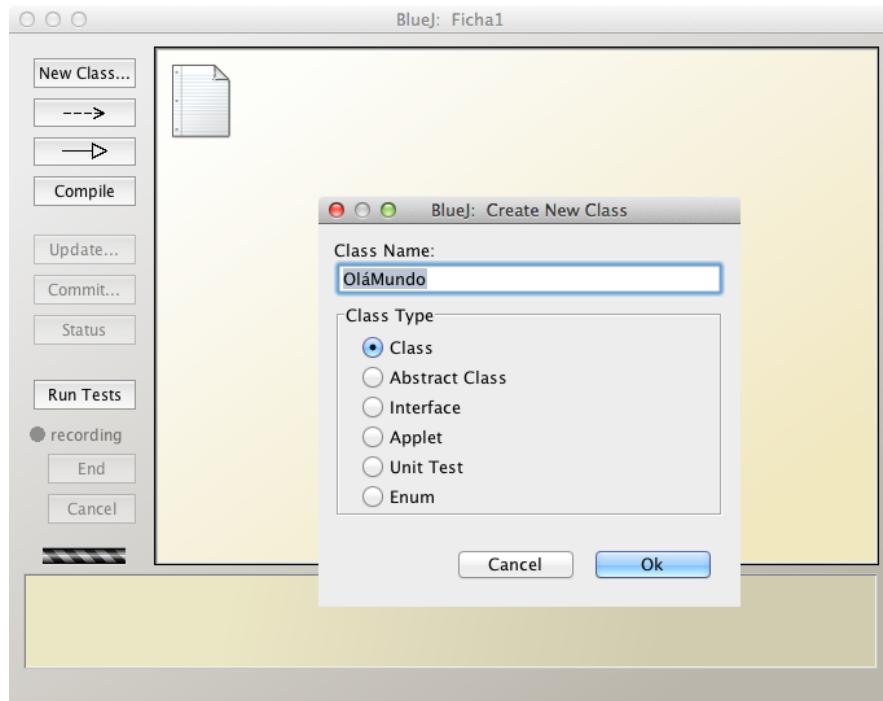
- `javac` é o compilador de Java para bytecode
- Se correu bem (ie. sem erros) foi gerado o ficheiro `OláMundo.class`

Execução: `$java OláMundo`

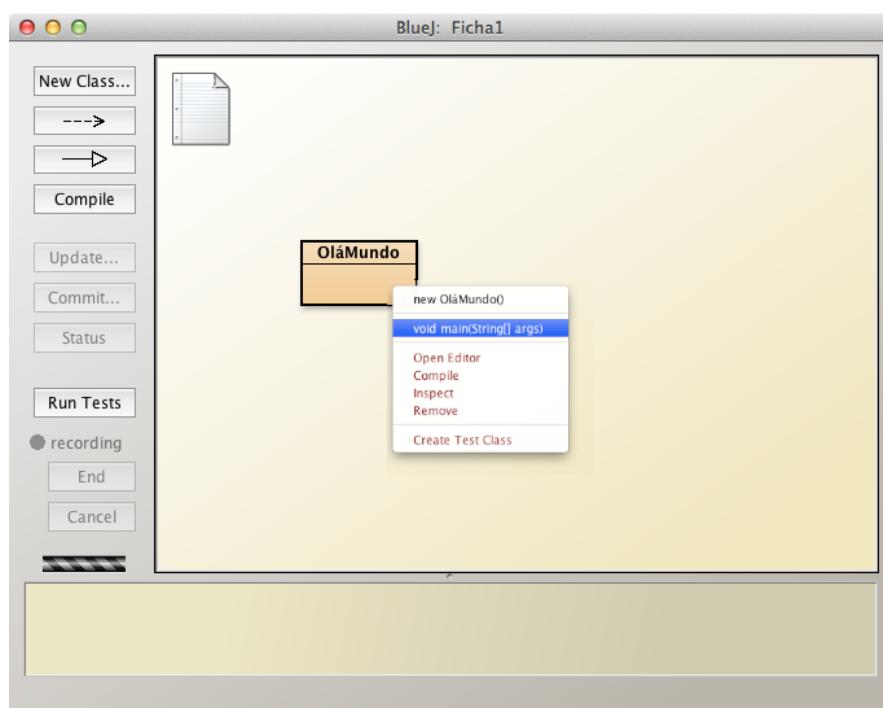
- `java` é a máquina virtual Java
- não se deve indicar a extensão do ficheiro (`.class`), apenas o nome da classe

Utilizando o IDE BlueJ

- BlueJ: Um ambiente de desenvolvimento para Java
  - Pensado para o ensino da linguagem
  - Disponível em: <http://www.bluej.org/>
1. Instale o Bluej
  2. Inicie o IDE e crie um novo projecto (`Project > New Project...`)
  3. Crie uma nova classe com o nome OláMundo (botão `New Class`)



4. Abra a classe e copie o código acima para o editor
5. Utilize o botão **Compile** para compilar o programa até que este não tenha erros.
6. Feche o editor e utilizando o menu de contexto execute o método **main** da classe.



7. Confirme os parâmetros do método e observe o resultado da execução.

## 2.3 Mais sobre a linguagem

### Comentários

- Comentar uma linha

```
// Este comentário termina no fim da linha!
```

- Comentar várias linhas

```
/* Este comentário pode ter várias linhas  
pois só termina quando aparecer  
o delimitador final */
```

- Gerar documentação

```
/** Este comentário vai gerar documentação via o javadoc */
```

### Declaração de variáveis

```
tipo nome_da_variável;  
tipo nome_da_variável[= valor];
```

#### Exemplos:

```
String nome;  
int i = 5, j = 4;  
long k = i + j;
```

### Declaração de Constantes

```
final tipo nome_da_constante = valor;
```

#### Exemplo:

```
final double PI = 3,1416;
```

### Tipos Primitivos

| Tipo    | Valor       | val. p/ Om. | Tamanho | Gama                                  |
|---------|-------------|-------------|---------|---------------------------------------|
| boolean | true, false | false       | 1       |                                       |
| char    | caracteres  | \u0000      | 16      | \u0000 a \uFFFF                       |
| byte    | inteiros    | 0           | 8       | -128 a +127                           |
| short   | inteiros    | 0           | 16      | -32768 a +32767                       |
| int     | inteiros    | 0           | 32      | -2147483648 a +2147483647             |
| long    | inteiros    | 0           | 64      | $\approx -1E+20$ a $+1E+20$           |
| float   | virg. flut. | 0.0         | 32      | $\approx \pm 3.4E+38$ a $\pm 1.4E-45$ |
| double  | virg. flut. | 0.0         | 64      | $\approx \pm 1.8E+308$ a $\pm 5E-324$ |

## Alguns Operadores

| Prioridade | Operação                 | Tipo dos Operandos | Assoc. | Descrição          |
|------------|--------------------------|--------------------|--------|--------------------|
| 1          | <code>++</code>          | aritméticos        | D      | pré/pós incremento |
|            | <code>-</code>           | aritméticos        | D      | pré/pós decremento |
|            | <code>+, -</code>        | aritméticos        | D      | sinal unário       |
|            | <code>!</code>           | boolean            | D      | negação            |
|            | <code>(tipo)</code>      | todos              | D      | conversão (cast)   |
| 2          | <code>*</code>           | aritméticos        | E      | multiplicação      |
|            | <code>/</code>           | aritméticos        | E      | divisão            |
|            | <code>%</code>           | aritméticos        | E      | resto              |
| 3          | <code>+</code>           | aritméticos        | E      | soma               |
|            | <code>-</code>           | aritméticos        | E      | subtração          |
| 5          | <code>&lt;, &lt;=</code> | aritméticos        | E      | menor (ou igual)   |
|            | <code>&gt;, &gt;=</code> | aritméticos        | E      | maior (ou igual)   |
| 6          | <code>==</code>          | primitivos         | E      | iguais             |
|            | <code>!=</code>          | primitivos         | E      | diferentes         |
| 7          | <code>&amp;</code>       | arit., char        | E      | e (bit a bit)      |
|            | <code>&amp;</code>       | boolean            | E      | e (booleano)       |
| 9          | <code>I</code>           | arit., char        | E      | ou (bit a bit)     |
|            | <code>I</code>           | boolean            | E      | ou (booleano)      |
| 10         | <code>&amp;&amp;</code>  | boolean            | E      | e (condicional)    |
| 11         | <code>  </code>          | boolean            | E      | ou (condicional)   |

## Estruturas de Controlo

- Em todos os casos abaixo, a `instrução` pode ser substituída por uma bloco de instruções:  
`{instrução_1; ...instrução_n;}`

### Condicionais:

```

if (condição) instrução
if (condição) instrução else instrução
switch (expressão) {
    case valor_1: instruções; [break;]
    ...
    case valor_n: instruções; [break;]
    default : instruções; [break;]
}

```

### Ciclos:

```

while (condição) instrução
do instrução while(condição)
for (inicialização; condição; incremento) instrução
for (variável: coleção_de_valores) instrução

```

### Outras:

tratamento de erros a ver mais tarde...

- Ver exemplos na Secção 2.5

## 2.4 A Java SE API

- A plataforma Java disponibiliza um conjunto alargado de Classes que podemos utilizar nos nossos programas.

– Ver documentação em <http://docs.oracle.com/javase/7/docs/api/>

- Essas classes estão organizadas numa estrutura hierárquica de pacotes (packages)

Por exemplo, o nome completo da classe `String` é `java.lang.String`

O que significa que a classe `String` está definida dentro do package `lang`, que por sua vez está dentro dentro do package `java`

- As classes do package `java.lang` estão disponíveis por omissão (daí podermos escrever apenas `String`)

- Classes de outros packages têm que ser importadas se quisermos evitar escrever os nomes completos:

```
import java.util.* (importa todas as classes do package — a evitar!)
import java.util.Scanner (importa apenas a classe Scanner)
```

- Podemos também importar métodos / variáveis de uma dada classe de modo a podermos utilizá-los sem referir a classe

```
import static java.lang.System.out
permite escrever
out.println("Olá Mundo");!
```

Algumas classes interessantes

- `java.lang.Math`

<http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html>

Classe com métodos para realizar operações numéricas: potência, raiz quadrada, funções trigonométricas, etc.

|   |                           |
|---|---------------------------|
| <code>Math.PI; Math.E</code>                | valores de PI e da base E |
| <code>tipo abs(tipo v)</code>               | valor absoluto de v       |
| <code>double sqrt(double v)</code>          | $\sqrt{v}$                |
| <code>double pow(double b, double e)</code> | $b^e$                     |
| <code>double random()</code>                | valor pseudo-aleatório    |
| <code>tipo max (tipo v1, tipo v2)</code>    |                           |
| <code>tipo min (tipo v1, tipo v2)</code>    |                           |
| <code>float round(double val)</code>        | arredondamento            |
| <code>int round(float v)</code>             |                           |
| <code>double sin(double val)</code>         | $\sin v$                  |
| <code>double cos(double val)</code>         | $\cos v$                  |
| <code>...</code>                            |                           |

- `java.lang.Integer` (Double, Float, etc.)

<http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html>

Classe correspondentes aos tipos primitivos: int, double, float, etc. Definem as constantes MAX\_VALUE e MIN\_VALUE (os valores máximo e mínimo que se podem representar em cada tipo). Por exemplo, Integer.MAX\_VALUE

- `java.util.GregorianCalendar`

<http://docs.oracle.com/javase/7/docs/api/java/util/GregorianCalendar.html>

Classe para o tratamento e registo de tempo (e datas).

Exemplos:

Criar um objecto com o valor do instante actual:

```
GregorianCalendar agora = new GregorianCalendar()
```

Criar um objecto com o valor de uma dada data/hora:

```
GregorianCalendar data =
    new GregorianCalendar(2012, Calendar.FEBRUARY, 22, 18, 29)
```

Mudar o ano da data para 2013:

```
data.set(GregorianCalendar.YEAR, 2013)
```

Calcular a diferença em mili-segundos entre as duas datas:

```
long dif = data.getTimeInMillis()-agora.getTimeInMillis()
```

- `java.lang.String`

<http://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

As strings em Java são objectos do tipo String.

Constantes (representam-se entre aspas):

```
""; "Isto é uma String"
```

Concatenação (juntar strings):

```
"Isto é" + "uma String"
```

Converter valores (de tipos primitivos) em Strings (método da classe String):

```
String valueOf(tipo val) – Exemplo: String str = String.valueOf(true);
```

Operações sobre strings (métodos de instância – voltaremos a isto!):

```
char charAt(int index) – Exemplo: "Olá".charAt(0) é 'O'
```

```
int length() – Exemplo: "Olá".length() é 3
```

```
String substring(int i, int f) – Exemplo: "Olá".substring(1,2) é "lá"
```

```
boolean equals(String str) – Exemplo: "Olá".equals("lá") é false
```

## 2.5 Input/output

### Escrever

- Instruções de leitura/escrita são enviadas aos objectos de onde se pretende ler/onde se pretende escrever
- Instruções de escrita no ecrã são enviadas ao objecto `System.out` (mais sobre ele em próximas aulas) — veja a linha 7 do programa da página 4
- Alguns métodos disponíveis para escrita

| Método                                      | Descrição                                |
|---|--|
| <code>print(valor)</code>                   | Escreve o valor passado como parâmetro   |
| <code>println(???)</code>                   | Escreve e muda de linha                  |
| <code>printf(formato, lista_valores)</code> | Escreve de acordo com o formato (à la C) |

- Exemplos:
  - `System.out.println("A idade é "+id);`
  - `System.out.print("A idade é "); System.out.println(id);`
  - `System.out.printf("A idade é %d\n",id);`
- Caracteres de conversão para o `printf`:
  - s (string), c (carácter), b (booleano), o (octal), h (hexadecimal), d (inteiro), f (real, vírgula fixa), e (real, vírgula flutuante), t (data) e \n (newline)

Informação detalhada sobre formatação com o `printf` pode ser encontrada aqui<sup>2</sup>

### Leitura de valores

- A leitura de valores é feita através de um objecto da classe `java.util.Scanner`
- Declarar e criar o objecto (mais sobre isto em próximas aulas):  
`Scanner input = new Scanner(System.in);` (assume-se que foi feito o `import`)
- Alguns métodos disponíveis para leitura

| Método                    | Descrição                 |
|---------------------------|---------------------------|
| <code>next()</code>       | Lê uma string             |
| <code>nextLine()</code>   | Lê uma linha de texto     |
| <code>nextInt()</code>    | Lê um <code>int</code>    |
| <code>nextDouble()</code> | Lê um <code>double</code> |
| <code>nextFloat()</code>  | Lê um <code>float</code>  |
| ...                       | ...                       |

- Exemplos:

---

<sup>2</sup><http://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

---

```
1  /**
2   * Lê dois números e diz qual o maior.
3   */
4 import java.util.Scanner;
5 public class Exemplo1 {
6     /** Diz qual é o maior de dois números */
7     public static void dizMaior(int i1, int i2) {
8         if (i1 > i2)
9             System.out.println("O maior é " + i1);
10        else
11            System.out.println("O maior é " + i2);
12    }
13    /** Início do programa */
14    public static void main(String[] args) {
15        int a, b;
16        Scanner ler = new Scanner(System.in);
17
18        System.out.print("Indique dois inteiros: ");
19        a = ler.nextInt();
20        b = ler.nextInt();
21        Exemplo1.dizMaior(a, b); // ou dizMaior(a,b)
22        ler.close();
23    }
24 }
```

---

```
1  /**
2   * Lê dois números e diz qual o maior.
3   * (agora, sem inventar a roda!)
4   */
5 import java.util.Scanner;
6 public class Exemplo2 {
7     /** Diz qual é o maior – utilizando a classe Math */
8     public static void dizMaior(int i1, int i2) {
9         System.out.println("O maior é " + Math.max(i2, i2));
10    }
11    /** Início do programa */
12    public static void main(String[] args) {
13        ...
14    }
15 }
```

---

---

```
1  /**
2   * Calcular o somatório de 10 números
3   */
4 import java.util.Scanner;
5 public class Exemplo3 {
6     public static void main(String[] args) {
7         int soma = 0;
8         Scanner input = new Scanner(System.in);
9
10        for (int i=0; i<10; i++) {
11            System.out.print("Valor: ");
12            soma += ler.nextInt();
13        }
14        System.out.println("O somatório é: "+soma);
15        per.close();
16    }
17 }
```

---

```
1  /**
2   * Calcular o somatório de n números enquanto o utilizador
3   * assim o quiser.
4   */
5 import java.util.Scanner;
6 public class Exemplo4 {
7     public static void main(String[] args) {
8         int soma, n;
9         String resp;
10        Scanner input = new Scanner(System.in);
11
12        do {
13            System.out.println("Quantos números vai somar? ");
14            n = input.nextInt();
15            soma = 0;
16            for (int i=0; i<n; i++) {
17                System.out.print("Valor: ");
18                soma += input.nextInt();
19            }
20            System.out.println("O somatório é: "+soma);
21            System.out.print("Quer repetir? [S/n]");
22            resp = input.next();
23        } while (resp.charAt(0) != 'n');
24        input.close();
25        System.out.println("Adeus!");
26    }
27 }
```

---

### 3 Exercícios

1. Ler um nome e uma idade e imprimir um texto com os resultados.
2. Ler três inteiros e escrevê-los por ordem crescente.
3. Ler 10 inteiros e determinar o maior inteiro introduzido.
4. Sendo  $n$  dado pelo utilizador, ler  $n$  reais e dar os resultados das suas potências de expoente  $exp$ , também introduzido pelo utilizador.
5. Ler uma sequência de inteiros positivos (terminada pelo valor -1) e determinar a diferença entre o maior e o menor inteiro lidos. Imprimir esse valor, bem como o maior e o menor.
6. Escrever um programa que calcule o factorial de um valor inteiro passado como parâmetro ao programa (e acessível através dos argumentos método `main(String[] args)`).
7. Escrever um programa que determine a data e hora do sistema, realize um ciclo com 10 milhões de incrementos unitários de uma dada variável, determine a hora após tal ciclo, e calcule o total de milissegundos que tal ciclo demorou a executar.
8. Escrever um programa que, dado o dia (1..31), mês (1..12) e ano, calcule o dia da semana.  
O dia da semana de datas entre Março de 1900 e Fevereiro de 2100 pode ser calculado do seguinte modo:
  - (a) Calcule o número total de dias entre 01/01/1900 e a data dada (ver algoritmo em baixo)
    - i. Subtraia 1900 do ano dado e multiplique por 365
    - ii. Adicione  $\frac{ano-1900}{4}$  (os anos bissextos)
    - iii. Se o ano dado for ele próprio bissexto, e o mês Janeiro ou Fevereiro, subtraia um ao resultado anterior.
    - iv. Adicione os dias já passados no corrente ano (considere 28 dias para Fevereiro)
  - (b) Calcule a divisão inteira desse número por 7
  - (c) O resto é o dia da semana: 0 – Domingo .. 6 – Sábado
9. Resolva agora o exercício anterior utilizando `GregorianCalendar`.
10. Escrever um programa que determine a diferença entre duas datas em dias, horas, minutos e segundos, utilizando um método auxiliar para o efeito. O método deverá aceitar duas datas e devolver uma string.
11. Escrever um programa que aceite  $n$  classificações (números reais) entre 0 e 20 e determine a sua média (usar `printf()` para os resultados).
12. Escrever um programa que aceite  $n$  temperaturas inteiras (pelo menos duas) e determine a média das temperaturas, o dia (2,3, ...) em que se registou a maior variação em valor absoluto relativamente ao dia anterior e qual o valor efectivo (positivo ou negativo) dessa variação. Os resultados devem ser apresentados sob a forma:

A média das  $n$  temperaturas foi de \_\_\_\_ graus.

A maior variação de temperatura registou-se entre os dias \_\_ e \_\_ e foi de \_\_\_ graus.

A temperatura entre o dia \_\_ e o dia \_\_ subiu/desceu \_\_\_ graus.

13. Escrever um programa que leia sucessivas vezes o raio (real) de um círculo e calcule a área e o perímetro respectivos com grande precisão (5 decimais). Usar printf() para os resultados. O programa apenas deverá terminar com a leitura de um raio = 0.0
14. Escrever um programa que faça a leitura de uma sequência não vazia de números reais terminada por 0.0 e calcule o seu somatório ( $\Sigma$ ) e o seu produtório ( $\Pi$ ) com precisão de 4 casas decimais no resultado.
15. Escrever um programa leia um inteiro  $n$  e imprima todos os números ímpares inferiores a  $n$ .
16. Escrever um programa que apresente ao utilizador um menu com as opções:

- 1.- Inserir
- 2.- Remover
- 3.- Consultar
- 4.- Gravar
- 5.- Sair

Em seguida, o programa deverá ler um inteiro, que apenas será válido se entre 1 e 5, e deverá apresentar ao utilizador, textualmente, a opção escolhida (Inserir, Remover, etc.) ou a mensagem ?Opção Inválida!?. O programa deverá repetir a apresentação do menu até que o utilizador seleccione a opção 5.- Sair.

17. Escrever um programa que gere um número aleatório entre 1 e 100. O programa dará 5 tentativas ao utilizador para acertar no número gerado. A cada tentativa do utilizador, o programa indicará se o número gerado é maior ou menor que o número dado pelo utilizador. À terceira tentativa falhada o utilizador perde. Quer perca quer acerte, o programa deve perguntar ao utilizador se quer continuar a jogar ou não. Se sim, novo número será gerado e o jogo retomado.
18. Escrever um programa que leia o ano, mês e dia de nascimento de uma pessoa e calcule a sua idade actual, indicando ao utilizador a data de nascimento lida, o dia de hoje e a idade que foi calculada.