

# POO (MiEI/LCC)

2015/2016

---

## Ficha Prática #01

---

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

## Conteúdo

<b>1</b>	<b>Objectivos</b>	<b>3</b>
<b>2</b>	<b>Introdução ao Java</b>	<b>3</b>
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 . . . . .	12
<b>3</b>	<b>Exercícios</b>	<b>15</b>

## 1 Objectivos

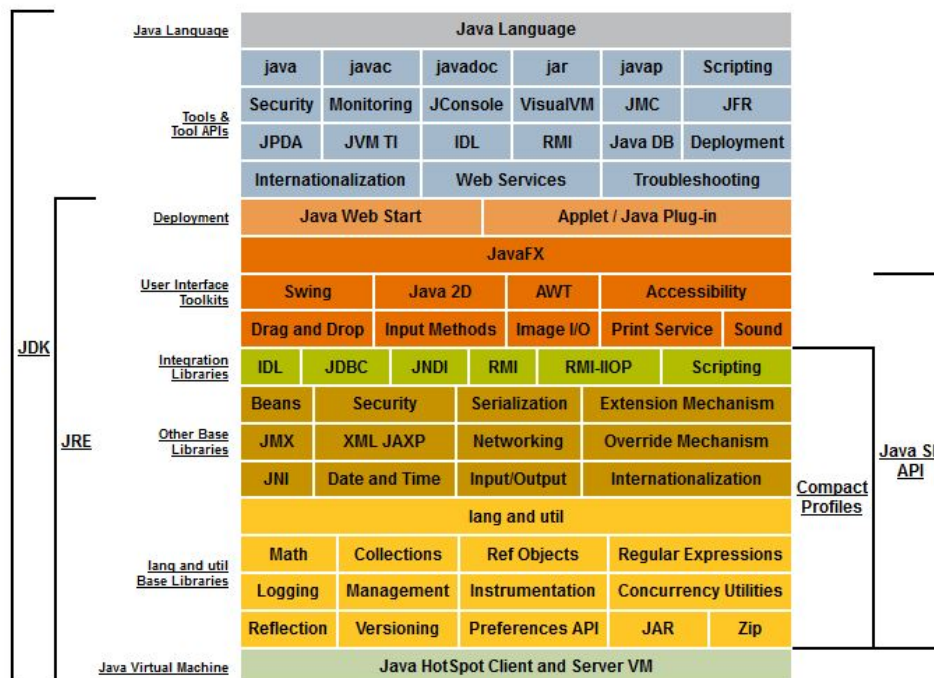
1. Conhecer o ambiente de desenvolvimento
2. Conhecer os Tipos de Dados e Estruturas de Controlo
3. Saber consultar a documentação online
4. Resolver exercícios simples

## 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
- Java SE API: Especificação que define a API que todas as implementações da *Standard Edition* de Java devem respeitar
  - A implementação da Java SE API que vamos utilizar é a Java JDK 8 da Oracle: disponível aqui<sup>1</sup>

#### A plataforma Java SE 8



<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 OlaMundo {
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 }
```

---

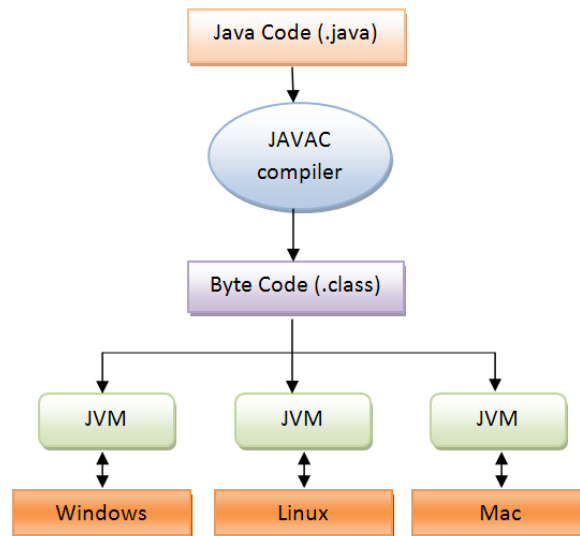
- O nome da classe deve começar por maiúscula (neste caso o nome é `OlaMundo`)
- O ficheiro deve ter o mesmo nome da classe com a extensão `java`
- Neste caso o ficheiro deve chamar-se `OlaMundo.java`
- A execução do programa começa pelo método `main` (recebe um array de `String` com os parâmetros passados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

```
1 public class OlaAlguem {
2     /** Método auxiliar */
3     public static String geraSaudacao(String nome) {
4         return "Olá "+nome+"!";
5     }
6     /** Inicio do programa */
7     public static void main(String[] args) {
8         String saudacao = OlaAlguem.geraSaudacao("Mundo");
9         System.out.println(saudacao);
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 OlaMundo.java`

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

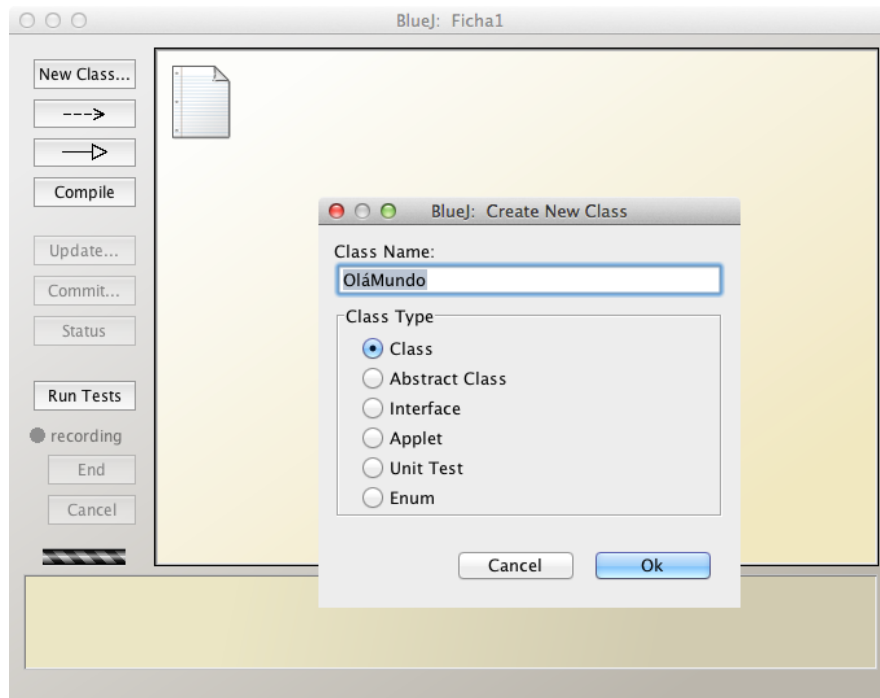
Execução: `$java OlaMundo`

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

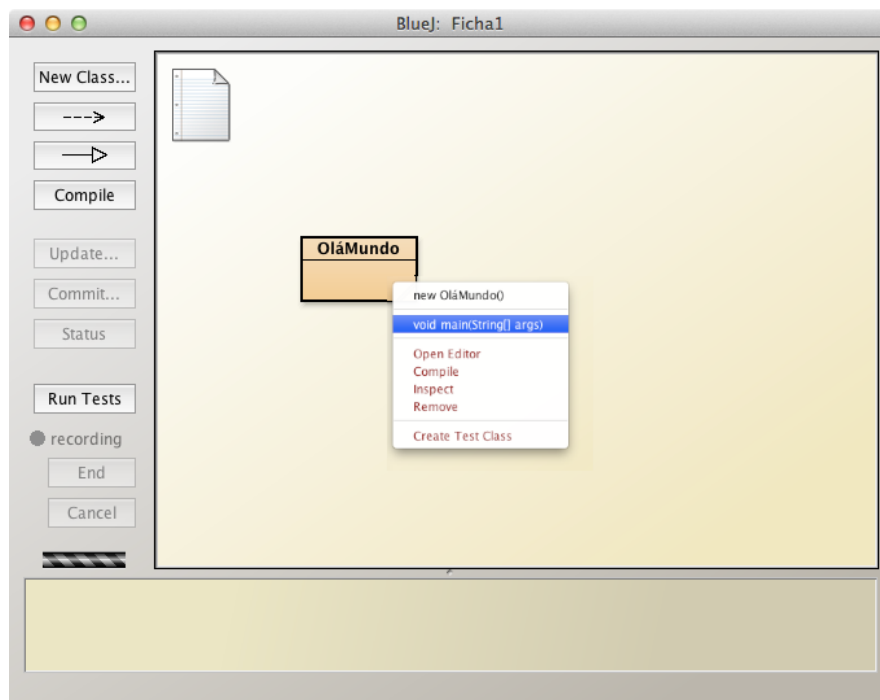
### Utilizando o IDE BlueJ

- 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 OlaMundo (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	$\approx -1\text{E}+20$ a $+1\text{E}+20$
float	virg. flut.	0.0	32	$\approx \pm 3.4\text{E}+38$ a $\pm 1.4\text{E}-45$
double	virg. flut.	0.0	64	$\approx \pm 1.8\text{E}+308$ a $\pm 5\text{E}-324$

### Alguns Operadores

Prioridade	Operação	Tipo dos Operandos	Assoc.	Descrição
1	++	aritméticos	D	pré/pós incremento
	-	aritméticos	D	pré/pós decremento
	+, -	aritméticos	D	sinal unário
	!	boolean	D	negação
	(tipo)	todos	D	conversão (cast)
2	*	aritméticos	E	multiplicação
	/	aritméticos	E	divisão
	%	aritméticos	E	resto
3	+	aritméticos	E	soma
	-	aritméticos	E	subtracção
5	<, <=	aritméticos	E	menor (ou igual)
	>, >=	aritméticos	E	maior (ou igual)
6	==	primitivos	E	iguais
	!=	primitivos	E	diferentes
7	&	arit., char	E	e (bit a bit)
	&	boolean	E	e (booleano)
9		arit., char	E	ou (bit a bit)
		boolean	E	ou (booleano)
10	&&	boolean	E	e (condicional)
11		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: colecção_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/8/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 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/8/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
<i>tipo</i> <code>abs(<i>tipo</i> 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
<i>tipo</i> <code>max(<i>tipo</i> v1, <i>tipo</i> v2)</code>	
<i>tipo</i> <code>min(<i>tipo</i> v1, <i>tipo</i> 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>	$\cos v$
...	

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

<http://docs.oracle.com/javase/8/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.time.LocalDate`

<http://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html>

Classe para o tratamento e registo de datas .

Exemplos:

Criar um objecto com o valor do instante actual:

```
LocalDate currentDate = LocalDate.now();
```

Criar um objecto com o valor de uma dada data:

```
LocalDate firstAug2014 = LocalDate.of(2014, 8, 1);
```

O dia X de um determinado ano:

```
LocalDate sixtyFifthDayOf2010 = LocalDate.ofYearDay(2010, 65)
```

- `java.time.LocalDateTime`

<http://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html>

Classe para o tratamento e registo de informação com data e tempo .

Exemplos:

Criar um objecto com o valor do instante actual:

```
LocalDateTime currentDateTime = LocalDateTime.now();
```

Criar um objecto com o valor de uma dada data/hora (ex: 2016-12-24 12:30):

```
LocalDateTime natal = LocalDateTime.of(2016, 12, 24, 12, 30);
```

O dia de Natal do ano passado:

```
LocalDateTime natal2016 = LocalDateTime.of(2014, Month.DECEMBER, 24, 12, 0);
```

Exemplo de utilização das várias classes sobre data e hora:

```
LocalDate date = LocalDate.of(2014, 2, 15); // 2014-02-15
```

```
boolean isBefore = LocalDate.now().isBefore(date); // false
```

```
// information about the month
```

```
Month february = date.getMonth(); // FEBRUARY
```

```
int februaryIntValue = february.getValue(); // 2
```

```
int minLength = february.minLength(); // 28
```

```
int maxLength = february.maxLength(); // 29
```

```
Month firstMonthOfQuarter = february.firstMonthOfQuarter(); // JANUARY
```

```
// information about the year
int year = date.getYear(); // 2014
int dayOfYear = date.getDayOfYear(); // 46
int lengthOfYear = date.lengthOfYear(); // 365
boolean isLeapYear = date.isLeapYear(); // false

DayOfWeek dayOfWeek = date.getDayOfWeek();
int dayOfWeekIntValue = dayOfWeek.getValue(); // 6
String dayOfWeekName = dayOfWeek.name(); // SATURDAY

int dayOfMonth = date.getDayOfMonth(); // 15
LocalDateTime startOfDay = date.atStartOfDay(); // 2014-02-15 00:00

// time information
LocalTime time = LocalTime.of(15, 30); // 15:30:00
int hour = time.getHour(); // 15
int second = time.getSecond(); // 0
int minute = time.getMinute(); // 30
int secondOfDay = time.toSecondOfDay(); // 55800
```

- `java.lang.String`

<http://docs.oracle.com/javase/8/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 índice) – 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 écran 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 ( <i>à la C</i> )

- 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 <b>int</b>
<code>nextDouble()</code>	Lê um <b>double</b>
<code>nextFloat()</code>	Lê um <b>float</b>
...	...

- 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(i1, 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 += input.nextInt();
13         }
14         System.out.println("O somatório é: "+soma);
15         input.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** (String) e um **saldo** (decimal) e imprimir um texto com os resultados.
2. Ler **dois inteiros** e escrevê-los por ordem decrescente, assim como a sua **média**.
3. Ler 10 inteiros e determinar a **quantidade** de números introduzidos com valor **superior** a 5.
4. Sendo  $n$  dado pelo utilizador, ler  $n$  reais e calcular as suas **raízes quadradas**.
5. Ler uma sequência de inteiros (terminada pelo valor 0) e determinar a sua **soma**. Imprimir esse valor, bem como o **maior** e o **menor** valor introduzidos.
6. Escrever um programa que calcule o **factorial** de um valor inteiro passado como parâmetro ao programa (e acessível através do argumento do método `main(String[] args)`<sup>3</sup>). O factorial do número deverá ser calculado num método auxiliar.
7. Escrever um programa que determine a **data e hora do sistema**, calcule o **factorial de 1,000,000**, determine a **hora após tal ciclo**, e calcule o **total de milissegundos** que tal ciclo demorou a executar.
8. Escrever um programa que, dada um data em dia (1..31), mês (1..12) e ano, **calcule o dia da semana** dessa data. 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 usando o seguinte algoritmo:
    - 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. Escrever um programa que determine a **soma** de duas datas em dias, horas, minutos e segundos, utilizando um **método auxiliar** para o efeito. O método deverá aceitar as duas datas e devolver uma string no formato “ddD hhH mmM ssS”.
10. Escrever um programa que aceite  $n$  **classificações** (números reais) de uma UC, e indique o **número de classificações** em cada um dos intervalos:  $[0, 5[$ ,  $[5, 10[$ ,  $[10, 15[$  e  $[15, 20]$ .
11. 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:

---

<sup>3</sup>Procure na classe `Integer` como converter a String recebida num número inteiro.

A média das  $_n$  temperaturas foi de  $_{\_}\_{\_}\_{\_}$  graus.

A maior variação registou-se entre os dias  $_{\_}\_{\_}$  e  $_{\_}\_{\_}$ , tendo a temperatura subido/descido<sup>4</sup>  $_{\_}\_{\_}$  graus.

12. Escrever um programa que **leia** sucessivas vezes a **base e altura** de um triângulo retângulo (valores reais) e calcule a **área e o perímetro** respectivos. Usar `printf()` para apresentar os resultados com uma precisão de 5 casas decimais. O programa apenas deverá terminar com a leitura de uma base = 0.0.
13. Escrever um programa que faça a leitura do **preço** (valor real) de um producto, e de uma **sequência não vazia de valores de stock** para esse producto (terminada por 0). Calcule o total de stock ( $\Sigma$ ) e o valor desse stock, apresentando o resultado com uma precisão de 4 casas decimais.
14. Escrever um programa que leia um inteiro  $n$  e imprima todos os **números primos** inferiores a  $n$ . Utilize um método auxiliar para determinar de um número é ou não primo.
15. Escrever um programa que apresente ao utilizador **um menu** com as opções:
  - 1.- Login
  - 2.- Registo
  - 3.- Informações
  - 0.- Sair

Em seguida, o programa deverá **ler um inteiro**, que apenas será válido se entre 0 e 3. Deverá haver também um **método auxiliar** para cada opção (excepto 0), que será invocado de acordo com a opção correspondente. Cada método deverá apresentar ao utilizador, textualmente, o **nome da operação** a que corresponde (Login, Registo, etc.), ou a mensagem *Opção Inválida*. O programa deverá **repetir a apresentação do menu** até que o utilizador seleccione a opção 0.

16. Escrever um programa que **gera dois números aleatórios** entre 1 e 100. O programa dará 5 tentativas ao utilizador para **acertar em um dos dois números** gerados. A cada tentativa do utilizador, o programa indicará a **distância do número mais próximo**. No fim do jogo o utilizador deverá ter a possibilidade de jogar novamente.
17. Escrever um programa que **leia o ano, mês e dia de nascimento** de uma pessoa e calcule a sua **idade actual em horas**, assim como a **data e hora** em que esse cálculo foi efectuado.

---

<sup>4</sup>Utilizar a expressão apropriada.