POO (LEI/LCC)

2024/2025

Ficha Prática #01

Conteúdo

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

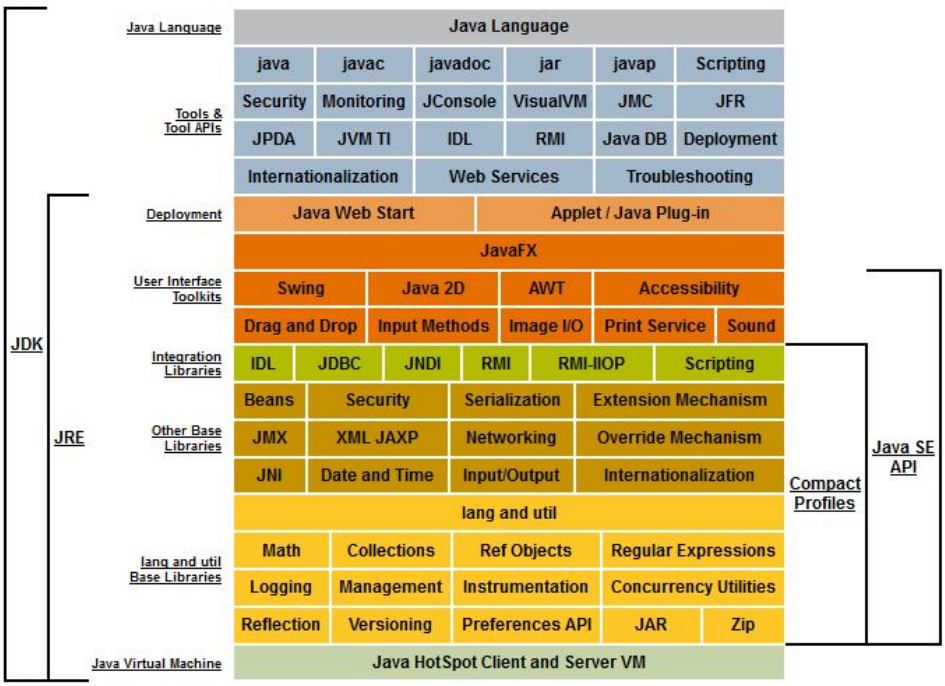
2 Introdução ao Java

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 da Oracle: disponível aqui[1]

A plataforma Java SE (a título de exemplo na figura está representada a release 8)

[1] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`



2.2 Compilação e execução de um programa Java

O meu primeiro programa Java

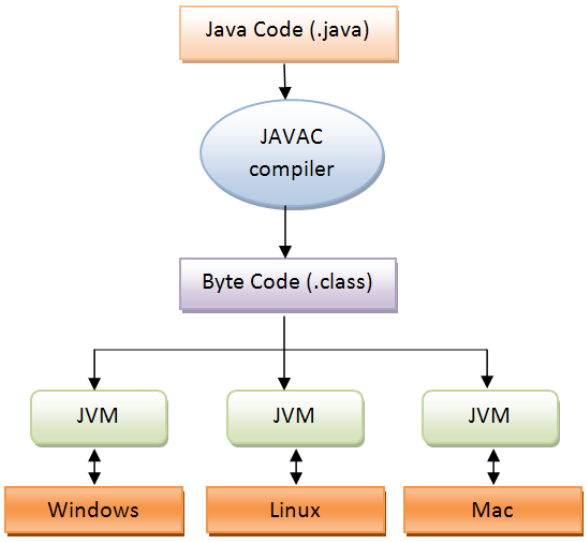
```
1 public class OlaMundo {
2     /**
3      * O meu primeiro método ç
4      */
5     public static void main(String[] args) {
6         // Colocar código aqui!
7         System.out.println("Olá Mundo!");
8     }
9 }
```

- O nome da classe deve começar (por convenção) 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:

```java
public class OlaAlguem {
    /** Método auxiliar */
    public static String geraSaudacao(String nome) {
        return "Olá "+nome+"!";
    }
    /** Inicio do programa */
    public static void main(String[] args) {
        String saudacao = geraSaudacao("Mundo");
        System.out.println(saudacao);
    }
}
```

Como executar o programa?

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



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 `OláMundo.class`

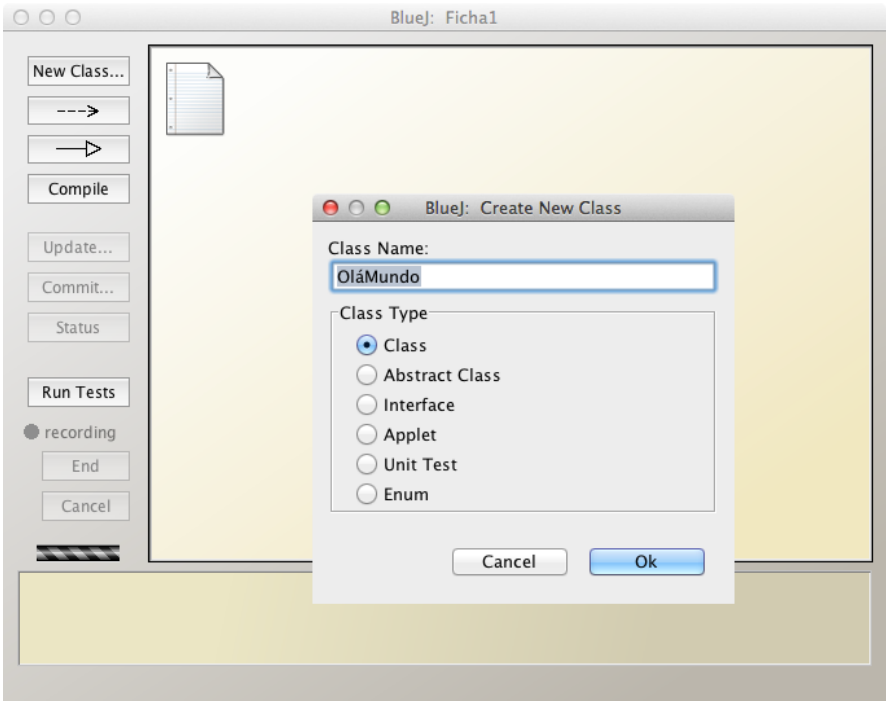
Execução: `$java OláMundo`

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

Utilizando o IDE BlueJ[2]

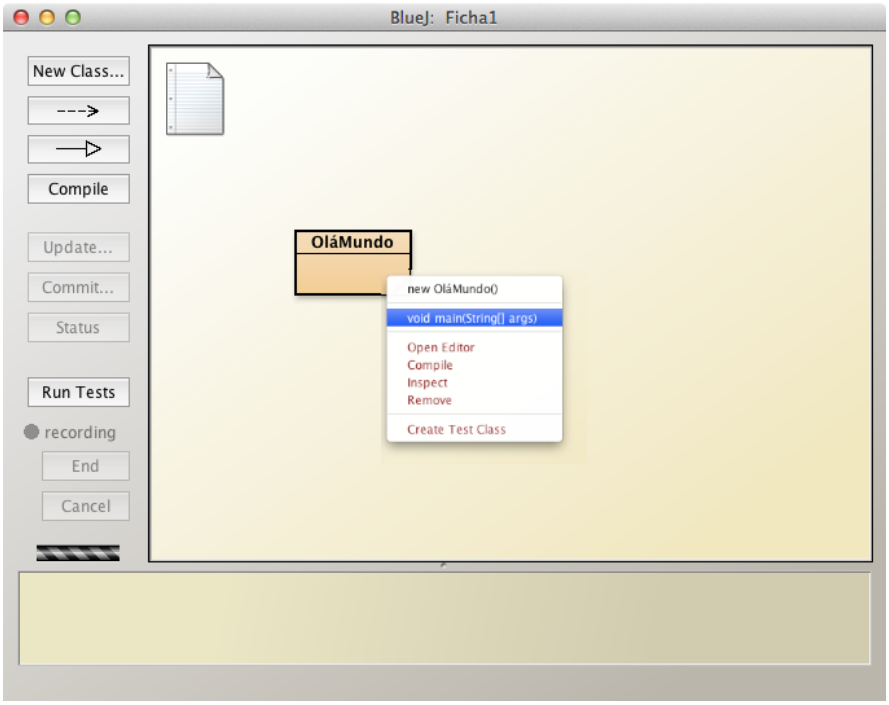
- BlueJ: Um ambiente de desenvolvimento para Java
- Pensado para o ensino da linguagem
- Disponível em: `http://www.bluej.org/`

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.

[2] A utilização do BlueJ não é obrigatória, pelo que poderá escolher o IDE que for da sua preferência.



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 programa 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	`++`	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 um 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 `https://docs.oracle.com/en/java/javase/23/docs/api/index.html` (neste caso para a versão 21, mas para a maioria das bibliotecas que utilizaremos não há diferenças em relação à documentação das versões anteriores).
- Essas classes estão organizadas numa estrutura hierárquica de pacotes (*packages*), que agrega conceitos similares

  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("OláMundo.");
  ```

Algumas classes interessantes

- `java.lang.Math`
  https://docs.oracle.com/en/java/javase/23/docs/api/java.base/java/lang/Math.html

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

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

- `java.lang.Integer` (`Double`, `Float`, etc.)
  https://docs.oracle.com/en/java/javase/23/docs/api/java.base/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`
  https://docs.oracle.com/en/java/javase/23/docs/api/java.base/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
  https://docs.oracle.com/en/java/javase/23/docs/api/java.base/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 natal2016 = LocalDateTime.of(2016, 12, 24, 12, 30);
  ```

  O dia de Natal do ano passado:
  ```
  LocalDateTime natal2021 = LocalDateTime.of(2021, Month.DECEMBER, 24, 12, 0);
  ```

  Exemplo de utilização das várias classes sobre data e hora:

  ```
  LocalDate date = LocalDate.of(2022, 2, 21); // 2022-02-21

  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(); // 2022
  int dayOfYear = date.getDayOfYear(); // 52
  int lengthOfYear = date.lengthOfYear(); // 365
  boolean isLeapYear = date.isLeapYear(); // false

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

```
int dayOfMonth = date.getDayOfMonth(); // 21
LocalDateTime startOfDay = date.atStartOfDay(); // 2022-02-21 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`
  `https://docs.oracle.com/en/java/javase/23/docs/api/java.base/java/lang/String.html`

  As strings em Java são objectos do tipo String.

  Constantes (representam-se entre aspas):

  `""`; `"Isto éuma 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 índex)` – 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
`print(valor)`	Escreve o valor passado como parâmetro
`println(??)`	Escreve e muda de linha
`printf(formato, lista_valores)`	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` (caratere), `b` (boleano), `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[3]

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
`next()`	Lê uma string
`nextLine()`	Lê uma linha de texto
`nextInt()`	Lê um **int**
`nextDouble()`	Lê um **double**
`nextFloat()`	Lê um **float**
...	...

- Exemplos:

```java
/**
 * Lê dois números e diz qual o maior.
 */
import java.util.Scanner;
public class Exemplo1 {
     /** Diz qual é o maior de dois números */
     public static void dizMaior(int i1, int i2) {
          if (i1 > i2)
            System.out.println("O maior é "+ i1);
          else
            System.out.println("O maior é "+ i2);
     }
     /** Início do programa */
```

[3] http://docs.oracle.com/javase/tutorial/java/data/numberformat.html

```
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          dizMaior(a, b); // ou Exemplo1.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  }
```

```java
/**
 * Calcular o somatório de 10 números
 */
import java.util.Scanner;
public class Exemplo3 {
     public static void main(String[] args) {
          int soma = 0;
          Scanner input = new Scanner(System.in);

          for (int i=0; i<10; i++) {
               System.out.print("Valor: ");
               soma += input.nextInt();
          }
          System.out.println("O somatório é: "+soma);
          input.close();
     }
}
```

```java
/**
 * Calcular o somatório de n números enquanto o utilizador
 * assim o quiser.
 */
import java.util.Scanner;
public class Exemplo4 {
     public static void main(String[] args) {
          int soma, n;
          String resp;
          Scanner input = new Scanner(System.in);

          do {
               System.out.println("Quantos números vai somar? ");
               n = input.nextInt();
               soma = 0;
               for (int i=0; i<n; i++) {
                    System.out.print("Valor: ");
                    soma += input.nextInt();
               }
               System.out.println("O somatório é: "+soma);
               System.out.print("Quer repetir? [S/n]");
               resp = input.next();
          } while (resp.charAt(0) != 'n');
          input.close();
          System.out.println("Adeus!");
     }
}
```

3 Exercícios I

Exercícios para fazer apenas com uma classe, com o método main e em que pode utilizar métodos auxiliares (da mesma classe). Notem que nesta fase estes programas são programas feitos em Java, mas seguem uma forma que é mais próxima dos programas em C que da programação por objectos. Servem apenas para praticar a escrita de código utilizando a nova linguagem!

1. 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

2. 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".

3. 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 $[15, 20]$.

4. 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 registou-se entre os dias __ e __, tendo a temperatura subido/descido[4] ___ graus.

5. 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.

[4] Utilizar a expressão apropriada.

6. 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.

   No fim da execução o utilizador deverá ter a possibilidade de jogar novamente.

7. 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.

4 Metodologia de programação de POO

Até ao momento vimos classes com o método **public static void** `main(String[] args)`, que é static e que acede a métodos static. Não faz sentido estarmos a programar recorrendo a definições globais (static), pelo que introduz-se agora a metodologia de programação, compatível com o paradigma de programação por objectos e que utilizaremos durante o semestre.
Essa metodologia assenta nos seguintes vectores:

- temos uma classe principal no programa que é a primeira a ser executada e que contém um método `main`.
- o método `main` declara objectos de classes computacionais e invoca nessas classes a execução de funcionalidades. Essas funcionalidades estão representadas nas classes como métodos (nesta fase, com alguma semelhança a funções)
- num modelo mais próximo daquele que será a prática desta UC a classe com o método main será a única onde serão efectuadas as actividades de leitura do teclado e escrita no ecran.

A título de exemplo, teremos então para os exercícios desta ficha um programa com 2 classes: a classe `TestePrograma`, com o método `main`, e uma outra `Ficha1` que terá métodos para cada um dos exercícios abaixo propostos.
De uma forma geral, a solução proposta passa por ter a seguinte definição de classes:

```
public class TestePrograma {

  public static void main(String[] args) {

    //inicialização de um scanner para leitura
    Scanner sc = new Scanner(System.in);

      Ficha1 f = new Ficha1();  // criar um objecto da classe
                                              // que implementa os métodos


      // ...
```

```
      //...
      // pergunta 3
      //

      System.out.println("Insira nome e saldo");
      String nome = sc.nextLine();
      float saldo = sc.nextFloat();
      String str = f.criaDescricaoConta(nome, saldo);
      System.out.println("Resposta =" + str);

      // ...

  }

}

public class Ficha1 {



  //implementação dos métodos que permitem
  //responder a cada um dos exercícios propostos

  // Pergunta 3: Ler um nome (String) e um saldo (decimal)
  // e imprimir uma String texto com os resultados.
  // Nota: o método devolve uma String que será
  // impressa no programa principal (na main)

   public String criaDescricaoConta(String nome, float saldo) {

      return "Nome: " + nome + ", saldo: " + saldo;
   }

    ...
    ...
    ...
    ...
  }

}
```

5 Exercícios II

Desenvolva cada um dos exercícios seguintes, de acordo com a metodologia apresentada anteriormente.

1. Ler um valor de temperatura em graus Celsius e devolver o correspondente valor em graus Farenheit.

   Assinatura do método: `public double celsiusParaFarenheit(double graus)`

2. Ler dois valores inteiros e invocar um método que determine o máximo dos dois. De seguida esse valor deve ser impresso no écran.

   Assinatura do método: `public int maximoNumeros(int a, int b)`

3. Ler um **nome** (String) e um **saldo** (decimal) e imprimir um texto com os valores obtidos na leitura.

   Assinatura do método: `public String criaDescricaoConta(String nome, double saldo)`

4. Ler um valor em **euros** e uma **taxa de conversão para libras** e imprimir o resultado da conversão em libras.

   Assinatura do método: `public double eurosParaLibras(double valor, double taxaConversao)`

5. Ler **dois inteiros** e escrevê-los por ordem decrescente, assim como a sua **média**.

6. Calcular o **factorial** de um valor inteiro passado como parâmetro ao método.

   `public long factorial(int num)`.

7. Escreva um método que determine a **data e hora do sistema**, calcule o **factorial de 5000**, determine a **hora após tal ciclo**, e calcule o **total de milissegundos** que tal ciclo demorou a executar.

   Assinatura do método `public long tempoGasto()`