

POO (LEI/LCC)

2023/2024

Ficha Prática #01

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	12
3	Exercícios I	15
4	Metodologia de programação de POO	16
5	Exercícios II	17

1 Objectivos

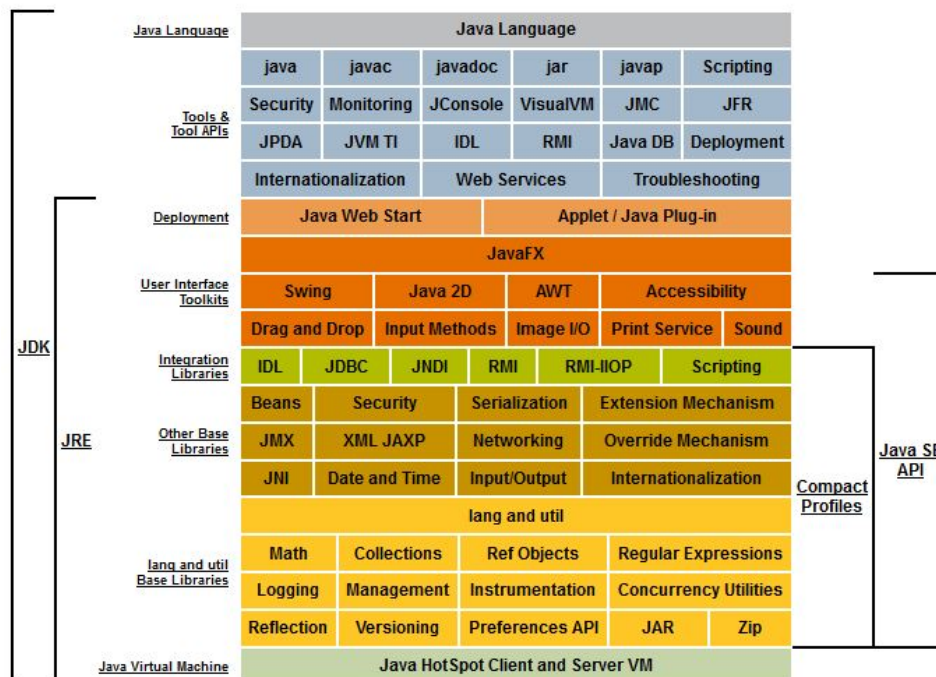
1. Conhecer o ambiente de desenvolvimento
2. Conhecer os Tipos de Dados e Estruturas de Controlo
3. Saber consultar a documentação online
4. Resolver exercícios simples

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 aqui¹

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

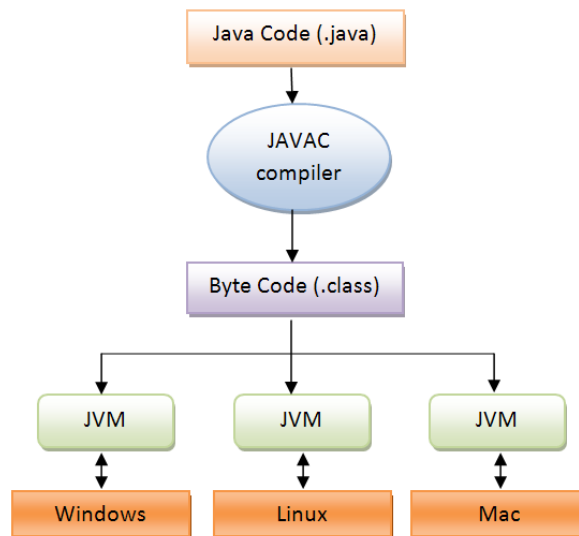
```
1 public class OlaMundo {
2     /**
3      * O meu primeiro método
4      */
5     public static void main(String[] args) {
6         // Colocar código aqui!
7         System.out.println("Olá Mundo!");
8     }
9 }
```

- 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:

```
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 = 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



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`

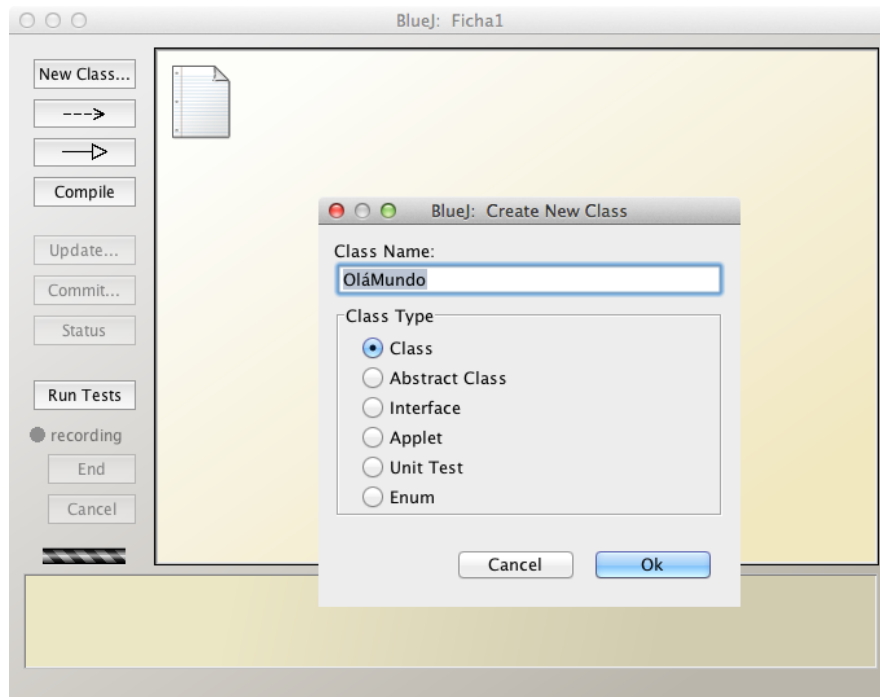
- `java` é a máquina virtual Java
- não se deve indicar a extensão do ficheiro (`.class`), apenas o nome da classe

Utilizando o IDE BlueJ²

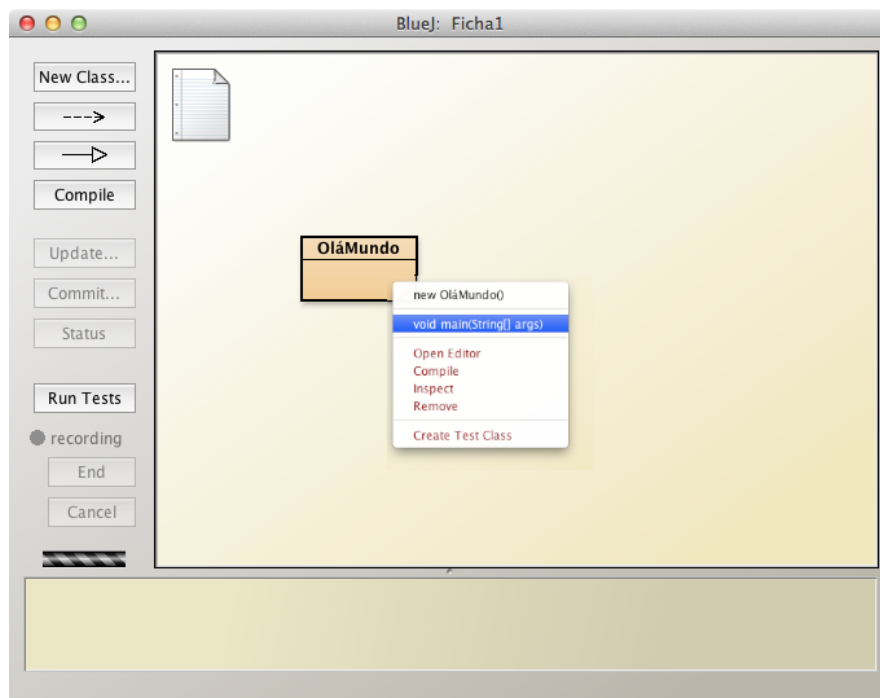
- BlueJ: Um ambiente de desenvolvimento para Java
- Pensado para o ensino da linguagem
- Disponível em: <http://www.bluej.org/>

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)

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



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 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	$\approx -1\text{E}+20$ 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 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/21/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 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`

<https://docs.oracle.com/en/java/javase/21/docs/api/java.base/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.)
<https://docs.oracle.com/en/java/javase/21/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/21/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/21/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/21/docs/api/java.base/java/lang/String.html)

<https://docs.oracle.com/en/java/javase/21/docs/api/java.base/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 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 (caratere), 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 aqui³

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>
...	...

³<http://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

- Exemplos:

```
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         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 }
```

```
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 }

```

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 ecrã.

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 Fahrenheit.

Assinatura do método: `public double celsiusParaFahrenheit(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 programa (e acessível através do argumento do método `main(String[] args)`⁵). O factorial do número deverá ser calculado num método com a assinatura
`public long factorial(int num)`.
7. Escreve um programa 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()`

⁵Procure na classe `Integer` como converter a `String` recebida num número inteiro.