FICHA PRÁTICA 1

LABORATÓRIO BASE DE JAVA6

TIPOS PRIMITIVOS &
ESTRUTURAS DE CONTROLO

Prof. F. Mário Martins

DI/UM

Versão 3.0

2012

FICHA PRÁTICA 1

LABORATÓRIO BASE DE JAVA

SÍNTESE TEÓRICA

JAVA é uma linguagem de programação por objectos. Porém, a tecnologia JAVA é muito mais do que a linguagem de programação em que se baseia. A figura seguinte mostra a arquitectura de software correspondente ao ambiente JSE6 que é necessário instalar nas nossas máquinas para executarmos e criarmos programas escritos em JAVA (na sua versão mais actual JAVA6).

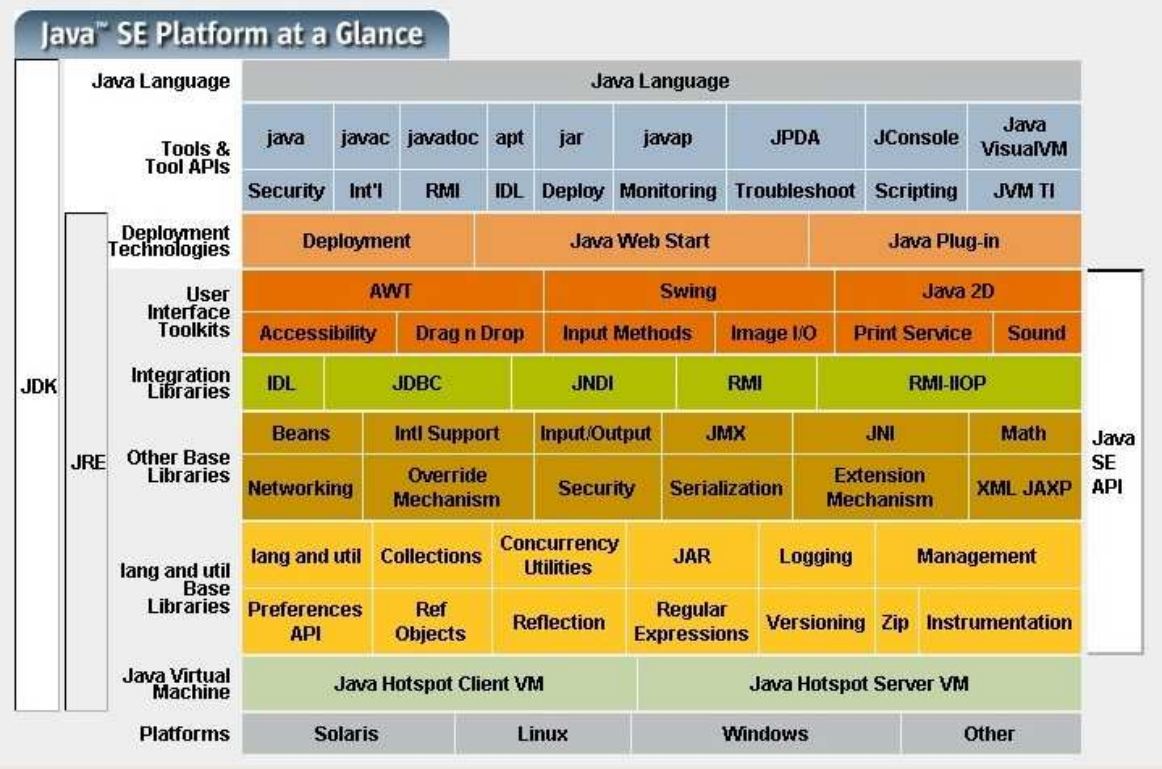


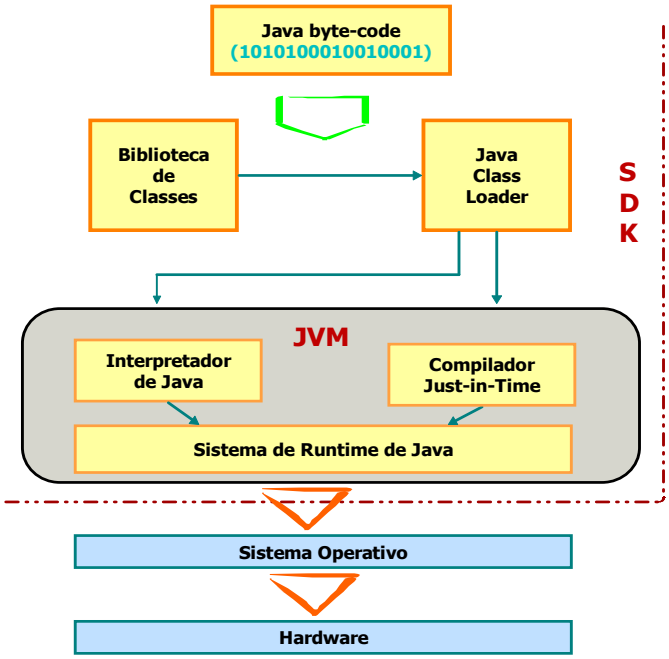
Figura 1 - Arquitectura JSE6

Quando programamos em JAVA6 temos à nossa disposição todas estas bibliotecas predefinidas, que possuem disponíveis classes para quase todas as mais diferentes funcionalidades necessárias às nossas aplicações.

Porém, o nosso objectivo neste contexto é conhecermos o núcleo fundamental da linguagem, e as suas construções básicas para realizarmos **programação sequencial**, mas seguindo princípios rigorosos da Engenharia de Software que são mais facilmente respeitados se utilizarmos correctamente características e propriedades disponíveis no paradigma da Programação por Objectos e suas linguagens (cf. C++, C# e JAVA).

A execução de um programa JAVA passa fundamentalmente pela compilação do seu código fonte para um código intermédio, designado **byte-code**. Este *byte-code*, que é o resultado da compilação, é um código standard que poderá ser em seguida executado (interpretado) por uma qualquer Java Virtual Machine (JVM). Naturalmente que, para cada sistema operativo e arquitectura, existirá uma JVM específica que interpreta correctamente o *byte-code* em tal contexto

(cf. Windows, Linux, Solaris, PDA, Java Card, etc.). Neste facto reside a grande portabilidade e flexibilidade da linguagem JAVA.



SINTAXE ESSENCIAL

1.- ESTRUTURA BASE DE UM PROGRAMA JAVA

Em JAVA tudo são classes. Um programa JAVA é uma classe especial que, entre outros, possui obrigatoriamente um método **main()** pelo qual se inicia a execução do código do programa. O nome do programa (classe) deverá ser igual ao do ficheiro fonte que a contém. Exemplo: a **public class Teste1** deverá ser guardada no ficheiro **Teste1.java.**

```
public class Teste1 {
    public static void main(String[] args) {

        // declarações e código
        // ............
    }
}
```

Porém, e por razões de estruturação do código, nada impede que se criem métodos externos ao método **main()** que pertencem igualmente ao programa e que podem ser invocados a partir do método **main()** e, até, invocarem-se entre si.
A figura seguinte mostra este tipo um pouco mais complexo de estruturação do programa, mas que é apenas um caso particular do primeiro.
Finalmente, e por razões a ver posteriormente, todos estes métodos devem possuir o atributo **static**, podendo ser **public** ou não (usaremos de momento sempre **public**).

```
public class Teste2 {

   public static <tipoRes> metodo_Aux1 (argumentos opcionais) {
      // .....
   }

   public static <tipoRes> metodo_Aux2 (argumentos opcionais) {
      // .....
   }

   public static void main(String[] args) {
      // ............

      // declarações e código
      // ............

   }
}
```

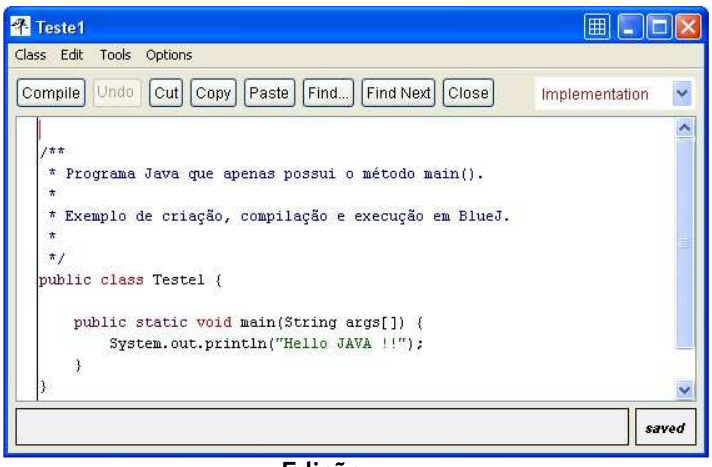
2.- Compilação e Execução a partir de linha de comandos

Ficheiro fonte: Teste1.java
Compilação: > javac Teste1.java ⇨ cria ficheiro Teste1.class
Execução: > java Teste1

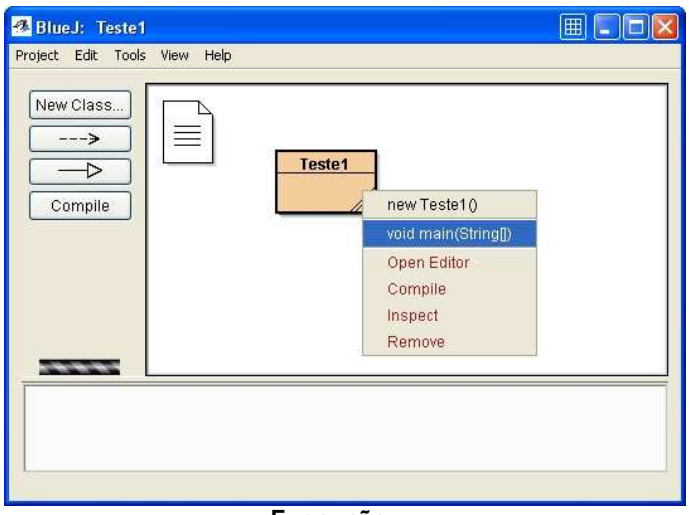
3.- Edição, Compilação e Execução usando BlueJ

- Invocar o BlueJ (eventualmente usar opção New Project ...);
- Criar o ficheiro fonte usando a opção New Class (ou editar ficheiro existente);
- Se fizermos apenas Save é criado um ícone sombreado com o nome do ficheiro criado, que se apresentará a tracejado indicando que não se encontra compilado ainda;
- Pode no entanto usar-se a opção Compile imediatamente após o fim da edição;
- Se não existirem erros, aparece o mesmo ícone mas sem qualquer tracejado;
- Para ficheiros não compilados, clicar no botão direito do rato e executar a opção Compile
- Para executar o programa, sobre o ícone clicar no botão direito do rato e seleccionar o método main()
- Os resultados são apresentados na Terminal Window do BlueJ

EXEMPLO:



Edição



Execução

Resultados

4.- Tipos Primitivos e Operadores

Os tipos primitivos de JAVA são os usuais tipos simples que se associam a variáveis nas quais pretendemos guardar **valores** (cf. Pascal, C, etc.). A tabela seguinte mostra os tipos disponíveis, gama de valores aceites e número de bits de representação.

Tipo	Valores	Omissão	Bits	Gama de valores
`boolean`	false, true	false	1	false a true
`char`	caracter unicode	\u0000	16	\u0000 a \uFFFF
`byte`	inteiro c/ sinal	0	8	-128 a +127
`short`	inteiro c/ sinal	0	16	-32768 a +32767
`int`	inteiro c/ sinal	0	32	-2147483648 a 2147483647
`long`	inteiro c/ sinal	0L	64	≈ -1E+20 a 1E+20
`float`	IEEE 754 FP	0.0F	32	± 3.4E+38 a ± 1.4E-45 (7 d)
`double`	IEEE 754 FP	0.0	64	± 1.8E+308 a ± 5E-324 (15d)

Tabela 1 – Tipos primitivos de Java

5.- Declaração e Inicialização de variáveis.

Associadas a um tipo de dados, poderemos ter uma sequência de declarações de variáveis separadas por vírgulas, eventualmente com atribuições dos valores de expressões para a inicialização de algumas delas, cf. a forma geral e os exemplos seguintes:

id_tipo *id_variável [= valor] [, id_variável [= valor] …] ;*

```
int dim = 20, lado, delta = 30;
char um = '1'; char newline = '\n';
byte b1 = 0x49;          // hexadecimal, cf. 0x como prefixo
```

```
long diametro = 34999L; double raio = -1.7E+5;
double j = .000000123;      // parte inteira igual a 0
int altura = dim + delta;    // inicialização usando cálculo de expressão
```

Associados a estes tipos simples existe um conjunto de operadores necessários à sua utilização, comparação, etc. Vejamos os mais importantes na tabela seguinte.

Precedência	Operador	Tipos dos Operandos	Associação	Operação
◎	+, -	número	D	sinal; unário
◎	!	boleano	D	negação
12	*, /	número, número	E	multipl, divisão
◎	/,%	inteiro, inteiro	E	quoc. int e módulo
11	+, -	número, número	E	soma e subtracção
9	<, <=	números	E	comparação
◎	>, >=	aritméticos	E	comparação
8	==	primitivos	E	igual valor
◎	!=	primitivos	E	valor diferente
◎	^	boleanos	E	OUEXC lógico
◎	|	boleanos	E	OU lógico
4	&&	boleano	E	E condicional
3	||	boleano	E	OU condicional
1	=	variável, qualquer	D	atribuição
◎	*= /= %=	variável, qualquer	D	atribuição após oper.
◎	+= -=	variável, qualquer	D	atribuição após oper.
◎	<<= >>=	variável, qualquer	D	atribuição após oper.
◎	>>>= &=	variável, qualquer	D	atribuição após oper.
◎	^= |=	variável, qualquer	D	atribuição após oper.

Tabela 2 – Operadores de JAVA para tipos simples

6.- Declaração de Constantes.

As constantes JAVA são também, por razões de estilo e de legibilidade, identificadas usando apenas **letras maiúsculas**, em certos casos usando-se também o símbolo _. As seguintes declarações,

```
final double PI = 3.14159273269;
final double R_CLAP = 8.314E+7;
```

definem, num dado contexto, **constantes identificadas**, cujos valores não poderão ser alterados por nenhuma instrução. O atributo **final** garante que um erro de compilação será gerado caso haja a tentativa de modificar tal valor.

7.- Comentários.

// este é um comentário monolinha; termina no fim desta linha
/* este é multilinha; só termina quando aparecer o delimitador final ***/**
/** este é um comentário de documentação para a ferramenta **javadoc */**

8.- Estruturas de Controlo

```
if (condição) instrução;
if (condição) { instrução1; instrução2; … }
if (condição) instrução1; else instrução2; ou { instruções }
if (condição) { instrução1; instrução2; … } else { instrução1; instrução2; …}

switch (expressão_simples) {
    case valor_1 : instruções; [break;]
    …
    case valor_n : instruções; [break;]
    default: instruções;
}

for (inicialização; condição de saída; incremento) instrução; ou { instruções }

for (Tipo variável : array de elementos de tipo Tipo) instrução; ou { instruções }

while (condição de execução) { instruções }

do { instruções; } while(condição de repetição);
```

9.- Importação Normal e Estática – Reutilização de Classes

Importação por *omissão*: import java.lang.*;
Importação global de um package: import java.util.*;
Importação selectiva de classes: import java.lang.Math;
Importação estática (elimina prefixos): import static java.lang.Math.abs;

10.- Output Básico e Formatado

10.1- As instruções básicas de escrita (*output*) de JAVA são dirigidas ao dispositivo básico de saída, o monitor, que a partir do programa é visto como um "ficheiro" de caracteres do sistema designado por **System.out**. Assim, são mensagens básicas todas as *strings* (cf. "abc" mais operador de concatenação **+** para *strings* e valores de tipos simples) enviadas para tal ficheiro usando a instrução **println()**, o que se escreve:

```
System.out.println("Bom dia e …\tbom trabalho!\n\n\n");
System.out.println("Linguagem : " + lp + 5);
System.out.println(nome + ", eu queria um " + carro + "!");
```

Se nos programas em que tivermos que realizar muitas operações de saída escrevermos no seu início a cláusula de importação **import static java.lang.System.out;**, então, em vez de se escrever **System.out.println(.)** bastará escrever-se **out.println(.)**, cf. em:

```
out.println("Hello Java!"); out.println(nome + "tem " + idade + " anos.");
```

10.2.- JAVA possui também uma forma formatada de escrita de dados, baseada em especial no método **printf()**.
O método **printf(formatString, lista_valores)** permitirá a saída de uma lista de valores (sejam variáveis, constantes ou expressões), que serão formatados segundo as directivas dadas na ***string* de formatação,** que é o primeiro parâmetro, e que pode incluir **texto livre**. A forma geral **de formatação** de valores de tipos primitivos é a seguinte (para cada valor a formatar):

```
%[índice_arg$] [flags] [dimensão][.decimais] conversão
```

Os **caracteres de conversão** são os que indicam o tipo de valor resultado da conversão do parâmetro, sendo: **c** (carácter), **b** (boleano), **o** (octal), **h** (hexadecimal), **d** (inteiro), **f** (real, vírgula fixa), **s** (*string*), **e** (real, vírgula flutuante), **t** (data) e **n** (*newline* independente da plataforma). Um valor de dado tipo se formatado para outro tipo compatível é automaticamente convertido.
As *flags* podem permitir alinhar os resultados à esquerda (**-**), obrigar a incluir sempre o sinal (**+**), colocar zeros no início (0), colocar espaços no início (**' '**) ou colocar parêntesis (**(**) se o número for negativo.

```
float f1 = 123.45f; double d2 = 234.678; double d3 = 12.45E-10;
out.printf("R1 %5.2f R2 %3$-12.7f Exp1 %2$8.4e%n",f1, d2, d3);
```

Por exemplo, usando apenas caracteres de conversão podemos automaticamente fazer a conversão de um número inteiro na base 10 para a base 8 e para a base 16.

```
int x = 1261;
out.printf("Inteiro %d = Octal %o = Hexa %h%n", x,x,x);
out.printf("Inteiro %d = Octal %1$o = Hexa %1$h%n", x);
```

11.- Input com classe Scanner

A classe **java.util.Scanner** possui métodos para realizar leituras de diversos tipos a partir de diversos ficheiros, e não só. Interessa-nos aqui ver como podemos usar esta classe para realizar leituras de valores de tipos simples a partir do teclado. O teclado está, por omissão, associado a uma variável designada **System.in**. Assim, teremos que associar um **Scanner** ao teclado para se poder ler os valores primitivos necessários aos programas. Vejamos os passos:

1.- Escrever no início do ficheiro a cláusula de importação **import java.util.Scanner**;
2.- Criar um **Scanner** que se vai associar ao teclado, e que vamos designar por **input**:

```
Scanner input = new Scanner(System.in);
```

3.- Tendo o scanner **input** associado ao teclado, usar métodos de **Scanner** para ler os valores, cf.:

```
String nome = input.next();            // lê uma string
String linha = input.nextLine();   // lê uma linha de texto terminada por \n
int x = input.nextInt();               // lê um inteiro válido
double pi = input.nextDouble();        // lê um real válido, sendo , o separador
input.next TipoSimples();                 // lê um valor de qualquer tipo simples
input.close();                         // fecha o scanner
```

Nota: Posteriormente usaremos uma classe Input especialmente desenvolvida

12.- Classes Predefinidas Importantes

```
java.lang.Math
   Math.PI; Math.E:          // valores de PI e da base E com grande precisão
   tipo_numérico abs(tipo_numérico val); // valor absoluto
   double sqrt(double val);              // raiz quadrada
   double pow(double base, double exp);  // potenciação
   double random();                      // resultado no intervalo [0.0 1.0[
   tipo_numérico max (tipo_numérico val1, tipo_numérico val2);
   tipo_numérico min (tipo_numérico val1, tipo_numérico val2);
   int round(float val);      float round(double val);
   double sin(double val);  double cos(double val);      // seno e coseno
```

```
java.lang.Integer java.lang.Double java.lang.Float …
    <classe>.MAX_VALUE; <classe>.MIN_VALUE       // máximo e mínimo definidos

java.util.GregorianCalendar    // classe útil para tratamento de datas
    GregorianCalendar agora = new GregorianCalendar();  // idem
    GregorianCalendar hoje = new GregorianCalendar(2007, Calendar.MARCH,
                                                          10, 23, 15);  // define data
    hoje.set(GregorianCalendar.YEAR, 2009);   // modifica o campo ANO
    int mês = hoje.get(GregorianCalendar.MONTH);   // consulta o campo MÊS

    // tempo decorrido desde o início da ERA até à data e hora actuais em ms
    long milis = agora.getTimeInMillis();
    // diferença entre duas datas (agora e inicio) em número de dias
   long dif_milis = agora.getTimeInMillis() - inicio.getTimeInMillis();
   int dias = dif_milis/(24*60*60*1000);
    // escrever a hora actual e escrever a data actual
    out.printf("%tT%n", agora)                              // 12:23:35
    out.printf("%1$tY/%1$tm/%1$td%n", agora); // 2005/03/21

java.lang.String
    Constantes: "" "abcd" "Uma linha\n" "Exemplo\t\tFinal\n\n"
    Concatenação: "abc" + "25" "Luís " + "Carlos" "Linha1\n" + "Linha2\n"
    String valueOf(tipo_simples val); // converte para string um valor simples
    char charAt(int índex);      // devolve o carácter na posição índex da string
    int length();                          //devolve o comprimento da string
    String substring(int inic, int fim);     // devolve uma substring
    boolean equals(String str)               // igualdade de strings
```

EXERCÍCIOS RESOLVIDOS:

Ex1: Ler um nome e uma idade e imprimir um texto com os resultados.

```
1  import java.util.Scanner;
2  public class Leitura1 {
3     public static void main(String[] args) {
4        String nome; int idade;
5        // Scanner: classe para leitura
6        Scanner input = new Scanner(System.in); // lê via teclado
7        System.out.print("Nome: "); nome = input.next();
8        System.out.print("Idade: "); idade = input.nextInt();
9        System.out.println(nome + " tem " + idade + " anos.");
10    }
11 }
```

Ex2: Ler 10 inteiros e determinar o maior inteiro introduzido.

```
1  import java.util.Scanner;
2   public class MaxInt {
3         public static void main(String[] args) {
4            Scanner input = new Scanner(System.in); // lê via teclado
5            int valor; int maior = Integer.MIN_VALUE;
6            for(int i = 1; i <= 10; i++) {
7               System.out.print("Inteiro " + i + " : "); valor = input.nextInt();
9               if (valor > maior) maior = valor;
10           }
11           System.out.println("Maior inteiro = " + maior);
12        }
13  }
```

Ex3: Sendo N dado pelo utilizador, ler N reais e dar os resultados das suas potências de expoente Exp, também introduzido pelo utilizador.

```
import static java.lang.System.out;
import java.util.Scanner;
public class Expoentes {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in); // lê via teclado
    int total; double valor; int exp;
    out.print("Total de valores a ler: "); total = input.nextInt();
    out.print("Expoente a que os quer elevar: "); exp = input.nextInt();
    for(int i = 1; i <= total; i++) {
        out.print("Introduza valor real: "); valor = input.nextDouble();
        out.printf("Valor %7.2f elevado a %2d = %8.4f%n", valor, exp,
                    Math.pow(valor, exp));
    }
  }
}
```

Ex4: 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.

```
import java.util.Scanner;
import static java.lang.System.out;
public class DiferencaInt {
  public static void main(String[] args) {
    int valor; int maior = Integer.MIN_VALUE; int menor = Integer.MAX_VALUE;
    int diferenca;
    Scanner input = new Scanner(System.in); // lê via teclado
    valor = input.nextInt();
    while(valor != -1) {
        if (valor > maior) maior = valor;
        if (valor < menor) menor = valor;
    }
    diferenca = maior - menor;
    out.println("Maior = " + maior + " Menor = " + menor + "\n");
    out.println("A diferença entre eles é : " + diferenca);
  }
}
```

Ex5: Escrever um programa que calcule o factorial de um valor inteiro, dado como argumento do método **main()** através dos argumentos deste método.

```
import static java.lang.System.out;
public class FactorialInt{
  public static long factorial(long n) {
    if (n==1) return 1; else return n*factorial(n-1);
  }
  public static void main(String[] args) {
    long i = Integer.parseInt(args[0]);
    out.println( i + "! = " + factorial(i) );
  }
}
```

Execução: > FactorialInt 4 **(em Bluej em args introduzir {"4"})**

Nota: Para outro tipo teríamos **parseDouble()**, **parseFloat()**, **parseByte()**, etc.

Ex6: 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.

```
import java.util.Scanner;
import java.util.Calendar;
import static java.lang.System.out;
public class Ex6 {
    public static void main(String[] args) {
        int x = 0;
        Calendar inicio = Calendar.getInstance();
        for(int i = 0; i < 10000000; i++) x = x +1;
        Calendar fim = Calendar.getInstance();
        long milisegs = fim.getTimeInMillis() - inicio.getTimeInMillis();
        out.println("O ciclo demorou " + milisegs + " milisegundos a executar.");
        out.println("A variável x tem o valor: " + x);
    }
}
```

Ex7: Escrever um programa que use um método auxiliar que aceite duas datas e determine a sua diferença em dias, horas, minutos e segundos. O resultado do método deverá ser uma *string*.

```
import java.util.Scanner;
import java.util.GregorianCalendar;
import static java.lang.System.out;
public class Ex7 {

   public static String difDatas(GregorianCalendar fim,
                                 GregorianCalendar inicio) {
       long totalmilis = fim.getTimeInMillis() - inicio.getTimeInMillis();
       long milis = totalmilis%1000;
       long totalseg = totalmilis/1000; long seg = totalseg%60;
       int totalmin = (int) totalseg/60; int min = totalmin%60;
       int totalhoras = totalmin/60; int horas = totalhoras%60;
       int totaldias = totalhoras/24;

     return totaldias + " dias, " + horas +" horas, "+ min+ " minutos e "
              + seg + "," + milis + " segundos.";
   }

   public static void main(String[] args) {
      GregorianCalendar inicio = new GregorianCalendar();
      inicio.set(GregorianCalendar.YEAR, 2007); inicio.set(GregorianCalendar.MONTH, 2);
      inicio.set(GregorianCalendar.DAY_OF_MONTH, 8);
      inicio.set(GregorianCalendar.HOUR, 12); inicio.set(GregorianCalendar.MINUTE, 20);
      inicio.set(GregorianCalendar.SECOND, 33);
      inicio.set(GregorianCalendar.MILLISECOND, 111);
      // ou new GregorianCalendar(2007, Calendar.MARCH, 8, 23, 15);
      GregorianCalendar fim = new GregorianCalendar();
      fim.set(GregorianCalendar.YEAR, 2009); fim.set(GregorianCalendar.MONTH, 2);
      fim.set(GregorianCalendar.DAY_OF_MONTH, 13);
      fim.set(GregorianCalendar.HOUR, 22); fim.set(GregorianCalendar.MINUTE, 12);
      fim.set(GregorianCalendar.SECOND, 15);
      fim.set(GregorianCalendar.MILLISECOND, 444);
      // ou new GregorianCalendar(2007, Calendar.MARCH, 10, 12, 15);
      String difTempo = difDatas(fim, inicio);
      out.println("Diferença: " + difTempo);
   }
}
```

Ex8: Escrever um programa aceite N classificações (números reais) entre 0 e 20 e determine a sua média (usar printf() para os resultados).

```
import static java.lang.System.out;
public class Ex8 {
   public static void main(String[] args) {
      double nota, soma = 0.0;
      int total;
      out.print("Total de notas a ler: ");
      total = Input.lerInt();
      for(int i = 1; i <= total; i++) {
         out.println("Nota N. " + i);
         nota = Input.lerDouble(); soma += nota;
      }
      out.printf("A média das %2d notas é %4.2f%n", total, soma/total);
   }
}
```

Ex9: Escrever um programa 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.
```

```
import java.util.Scanner;
import static java.lang.System.out;
import static java.lang.Math.abs;
public class Temperaturas {
   public static void main(String[] args) {
      int temp; int anterior; int dia = 0; int total;
      int maiorDif = 0; int soma = 0;
      int difReal = 0, difAbs = 0;
      Scanner input = new Scanner(System.in); // lê via teclado
      out.print("Total de temperaturas a ler: ");
      total = Input.lerInt();
      out.print("Temperatura " + 1 + " : ");
      temp = Input.lerInt(); soma = temp;
      for(int i = 2; i <= total; i++) {
         anterior = temp;
         out.print("Temperatura " + i + " : ");
         temp = Input.lerInt();
         soma += temp; difReal = temp - anterior;
         difAbs = abs(temp - anterior);
         if (difAbs > maiorDif) {
            dia = i; maiorDif = difAbs;
         }
      }
      // resultados
      out.printf("A média das %2d temperaturas é %4.2f%n", total, ((double) soma)/total);
      out.println("Maior variação de temperaturas entre os dias " + (dia-1) + " e " + dia);
      String txt = difReal > 0 ? "subiu " : "desceu ";
      out.println("A temperatura entre esses dias " + txt + difAbs + " graus.");
   }
}
```

Ex10: 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

```
import static java.lang.Math.PI;
import static java.lang.System.out;
public class AreaCirculo {
   public static void main(String[] args) {
      double raio;
      out.print("Valor do raio: ");
      raio = Input.lerDouble();
      while(raio != 0.0) {
         out.printf("Raio = %7.3f => Área = %9.5f e Perímetro = %9.5f%n", raio,
                    PI*raio*raio, 2*PI*raio);
         out.print("Valor do raio: "); raio = Input.lerDouble();
      }
   }
}
```

Ex11: Escrever um programa que faça a leitura de uma sequência não vazia de números reais terminada por 0.0 e calcule o seu somatório (∑) e o seu produtório (∏) com precisão de 4 casas decimais no resultado.

```
import static java.lang.System.out;
public class Ex11 {
    public static void main(String[] args) {
       double soma = 0; double prod = 1; double num;
       int conta = 0;
       out.print("Número Real: "); num = Input.lerDouble();
       while(num != 0.0) {
          soma += num; prod *= num; conta++;
          out.print("Número Real: "); num = Input.lerDouble();
       }
       out.printf("Soma dos %2d números reais = %8.4f ; Produto = %12.4f%n%n",
              conta, soma, prod);
   }
}
```

Ex12: Escrever um programa leia um inteiro N e imprima todos os números ímpares inferiores a N.

```
import static java.lang.System.out;
public class Impares {
   public static void main(String[] args) {
      int limite;
      out.print("Ler número limite: ");
      limite = Input.lerInt();
      out.println("Números ímpares menores ou iguais a " + limite);
      for(int i = 1; i <= limite; i = i + 2) out.println(i);
   }
}
```

Ex13: Escrever um programa que apresente ao utilizador um menu vertical com as opções:

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

Em seguida, o programa deverá ler um **int**, 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.

```
import static java.lang.System.out;
public class Ex13 {
   public static void menu() {
      out.println("-----------------------------------------");
      out.println("--     OPCOES DISPONÍVEIS     ---");
      out.println("-----------------------------------------");
      out.println(" 1. INSERIR ");
      out.println(" 2. REMOVER ");
      out.println(" 3. CONSULTAR ");
      out.println(" 4. GRAVAR ");
      out.println(" 5. SAIR ");
      out.println("-----------------------------------------");
   }

   public static int opcao() {
      int opcao;
      boolean valida = false;
      do {
         out.print("OPCAO: "); opcao = lnput.lerInt();
         valida = (opcao >= 1) && (opcao <= 5);
         if(!valida) out.println("OPCAO INVÁLIDA !!");
        }
      while(!valida);
      return opcao;
   }
   public static void texto_opcao(int op) {
      switch(op) {
         case 1 : out.println("INSERIR"); break;
         case 2 : out.println("REMOVER"); break;
         case 3 : out.println("CONSULTAR"); break;
         case 4 : out.println("GRAVAR"); break;
         case 5 : out.println("SAIR"); break;
      }
   }

   public static void main(String[] args) {
      int escolha;
      do {
         menu();
         escolha = opcao();
         texto_opcao(escolha);
      }
      while(escolha != 5);
      out.println("FIM DE PROGRAMA !!");
   }
}
```

Ex14: 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.

```
import static java.lang.System.out;
import static java.lang.Math.random;
public class Ex14 {
   public static int geraNumero() {
      int numAleat = 1 + (int) (random() * 100);
      return numAleat;
   }
```

```
   public static void main(String[] args) {
      int numero; int opcao; int tentativa; int palpite;
      int resposta;
      boolean acertou;
      do {
        numero = geraNumero(); tentativa = 1;
        out.print("Tentativa " + tentativa + " - Qual o numero gerado? ");
        palpite = Input.lerInt();
        acertou = false;
        while( tentativa < 5 && !acertou) {
           if(palpite == numero) acertou = true;
           else {
                 if (palpite > numero) out.println("PARA BAIXO !!");
                 else out.println("PARA CIMA !!");
                 tentativa++;
                 out.print("Tentativa " + tentativa + " - Qual o numero gerado? ");
                 palpite = Input.lerInt();
               }
         }
         // verifica se acertou ou não
         if (acertou) out.println("PARABÉNS ACERTOU !!");
         else out.println("FALHOU ! O número era o " + numero);

         out.println("Continuar Jogo (1) ou Sair(outro): ");
         resposta = Input.lerInt();
      }
      while(resposta == 1);
      out.println("FIM DO JOGO .....");
   }
}
```

Ex15: 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.

```
import java.util.GregorianCalendar;
import static java.lang.System.out;
public class Ex15 {
    public static void main(String[] args) {
      int ano, mes, dia; String resp = "";
      do {
       out.print("Ano de nascimento: ");
       ano = Input.lerInt();
       do {
          out.print("Mês de nascimento: ");
          mes = Input.lerInt();
       }
       while(mes <= 0 || mes > 12);
       do {
          out.print("Dia de nascimento: ");
          dia = Input.lerInt();
       }
       while(dia <= 0 || dia > 31);

       // Nota: Janeiro = 0 para um Calendar !!
       GregorianCalendar dataNascimento =
         new GregorianCalendar(ano, mes-1, dia);
       out.printf("NAscido a %1$td/%1$tm/%1$tY%n", dataNascimento);
       GregorianCalendar hoje = new GregorianCalendar();
```

```
        // diferença de anos
        int anos = hoje.get(GregorianCalendar.YEAR) -
                        dataNascimento.get(GregorianCalendar.YEAR);
        out.printf("DATA ACTUAL : %1$td-%1$tm-%1$tY%n", hoje);
        out.println("IDADE NÃO CORRIGIDA: " + anos + " anos.");
        GregorianCalendar diaDeAnosEsteAno = dataNascimento;
        // dia de anos este ano = data de nascimento + idade não corrigida
        diaDeAnosEsteAno.add(GregorianCalendar.YEAR, anos);
        out.printf("ANOS A %1$td-%1$tm-%1$tY%n", diaDeAnosEsteAno);
        // já fez anos ou não ??
        if (hoje.before(diaDeAnosEsteAno))
             out.println("Ainda lá não chegamos !!"); anos--;
        out.println("A pessoa tem " + anos + " anos !");
        out.println("----------------------------------------");
        out.print("Quer calcular mais (S/*) ? "); resp = Input.lerString();
        out.println();
       }
       while(resp.equals("S") || resp.equals("s"));
       out.println("---- FIM DO CALCULADOR DE IDADES .... ");
    }
}
```

Ex16: Escrever um programa que leia duas datas (ano, mês e dia) e determine o número de anos entre tais datas.

```
/**
 * Cálculo da Diferença em Anos entre duas Datas
 *
 * @author FMM
 * @version 1.0/2009
 */
import java.util.GregorianCalendar;
import static java.lang.System.out;
public class Ex16 {

     public static GregorianCalendar lerData() {
       int ano, mes, dia;
      do {
        out.print("Ano a considerar: ");
        ano = Input.lerInt();
        }
        while(ano <= 0);

       do {
          out.print("Mês (Janeiro = 0) : ");
          mes = Input.lerInt();
       }
       while(mes <= 0 || mes > 12);

       do {
          out.print("Dia: ");
          dia = Input.lerInt();
       }
       while(dia <= 0 || dia > 31);

       return new GregorianCalendar(ano, mes, dia);
     }
```

```
    public static int totalAnos(GregorianCalendar data1,
                                GregorianCalendar data2) {
      long dif = data1.getTimeInMillis() - data2.getTimeInMillis();
      // de milisegundos para anos
      long totalSeg = dif/1000; int totalMin = (int) totalSeg/60;
      int totalDias = totalMin/(60*24);
      int anos = totalDias/(30*12);
      return anos;
    }

  public static void main(String[] args) {
     out.println("DATAS EM COMPARAÇÃO : ");
     GregorianCalendar data1, data2;
     data1 = lerData(); data2 = lerData();
     out.printf("DATA1 %1$td/%1$tm/%1$tY%n", data1);
     out.printf("DATA1 %1$td/%1$tm/%1$tY%n", data2);
     int anos = 0;
     if(data1.before(data2))
     anos = data2.get(GregorianCalendar.YEAR) - data1.get(GregorianCalendar.YEAR);
      if(data2.get(GregorianCalendar.MONTH)<
         data1.get(GregorianCalendar.MONTH))  anos--;
     out.println("Diferem de " + anos + " anos!");
     // ou ainda
     long anos1 = (data1.before(data2)) ? totalAnos(data2, data1) :
                           totalAnos(data1, data2);
     if(data2.get(GregorianCalendar.MONTH) <
       data1.get(GregorianCalendar.MONTH)) anos--;
     out.println("Diferem de " + (int) anos1 + " anos!");
  }
}
```