LEITURA DE DADOS

Para realizar leitura é necessário trabalhar com a classe Scanner e outras funcionalidades de I/O.
A **classe Scanner**, tem justamente a finalidade de facilitar a entrada de dados no modo Consola. Essa classe apareceu a partir do Java 5, antes dessa versão era complicado criar programas que recebiam valores de variáveis no modo Consola.

A classe **Scanner** tem como objetivo separar a entrada dos textos em blocos, gerando os conhecidos *tokens* que são sequências de carateres separados por delimitadores que, por padrão, correspondem aos espaços em branco, tabulações e mudança de linha.

Com esta classe podem ser convertidos textos para tipos primitivos (void, char, byte, short, int, long, float e double), podendo esses textos ser considerados como objetos do tipo String, InputStream (classe abstrata que define o comportamento padrão de um fluxo de entrada - é possível ler bytes) e ficheiros.

Quando invocada a classe Scanner, o compilador pedirá para fazer a seguinte importação:

```
import java.util.Scanner;
```

Para fazer essa ação na prática, é necessário criar um objeto do tipo Scanner que passa como argumento o objeto *System.in* dentro do construtor, do seguinte modo:

Declarações do Scanner

```
import java.util.Scanner;
public class TestaDeclaracaoScanner {
   public static void main(String[] args) {
     //Lê a partir da linha de comando
        Scanner sc1 = new Scanner(System.in); //1-instanciar um objeto da classe scanner
        System.out.print("Insira o seu nome: ");
     // String nome=new Scanner(System.in).nextLine(); ou substituir 1 e 2 por esta instrução
        String nome=sc1.nextLine();  // 2- obtenção do valor lido
        System.out.print("\nOlá "+nome+"!");
        }
}
```

O objeto *System.in* é o que faz a leitura do que se escreve no teclado.
Para cada um dos dados primitivos existe uma chamada do método para retornar o valor especificado na entrada de dados, sempre seguindo o formato **nextTipodado().**

Métodos da Classe Scanner

Método	Descrição
close()	Fecha o acesso ao objeto de leitura, impedindo-a. Qualquer tentativa de leitura de tokens a partir do objeto Scanner resulta numa exceção.
findInLine()	Encontra a próxima ocorrência de um padrão, ignorando máscaras ou strings e ignorando delimitadores.
hasNext()	Retorna um valor booleano verdadeiro (true) se o objeto Scanner tem mais dados de entrada.
hasNextXyz()	Retorna um valor booleano como verdadeiro (true) se a próxima entrada, for do tipo Xyz, sendo Xyz Boolean, Byte, Short, Int, Long, Float ou Double.
match()	Retorna o resultado da pesquisa da último operação realizada pelo objeto Scanner atual.
next()	Procura e retorna a próxima informação do objeto Scanner que satisfaça uma condição.
nextBigDecimal(), nextBigInteger()	Varre a próxima entrada como BigDecimal ou BigInteger.
nextXyz()	Varre a próxima entrada do tipo Xyz, sendo Xyz Boolean, Byte, Short, Int, Long, Float ou Double.
nextLine()	Mostra a linha atual do objeto Scanner e avança para a próxima linha.
radix()	Retorna o índice atual do objeto Scanner.
skip()	Salta para a próxima pesquisa de um padrão especificado ignorando delimitadores.
toString()	Retorna uma string que é uma representação do objeto Scanner.

Resumindo, para utilizar a classe Scanner numa aplicação Java deve-se proceder da seguinte maneira:

[1]	importar o pacote java.util: `import java.util.Scanner;`
[2]	Instanciar e criar um objeto Scanner: `Scanner ler = new Scanner(System.in);`
[3]	Ler valores através do teclado:

[3.1]	Ler um valor inteiro:
	`int n;`
	
	`System.out.printf("Informe um número para a tabuada: ");`
	`n = ler.nextInt();`

[3.2]	Lendo um valor real:
	`float preco;`
	
	`System.out.printf("Informe o preço da mercadoria = R$ ");`
	`preco = ler.nextFloat();`

[3.3]	Lendo um valor real:
	`double salario;`
	`System.out.printf("Informe o salário do Funcionário = R$ ");`
	`salario = ler.nextDouble();`

[3.4]	Ler um valor Booleano: **boolean** cliente; System.out.printf("é cliente? "); cliente = ler.**nextBoolean**();
[3.5]	Ler uma String. Usado na leitura de palavras simples que não usam o caráter de espaço (ou barra de espaço): **String** s; System.out.printf("Informe uma palavra simples:\n"); s = ler.**next**();
[3.6]	Ler uma String. Usado na leitura de palavras compostas, por exemplo, Pato Branco: **String** s; System.out.printf("Insira uma cadeia de carateres:\n"); s = ler.**nextLine**();

[3.7]	Na leitura consecutiva de valores numéricos e String deve-se esvaziar o buffer do teclado antes da leitura do valor String, por exemplo:
	int n;
	String s;
	
	System.out.printf("Insira um Número Inteiro: ");
	n = ler.**nextInt**();
	
	ler.**nextLine**(); *// esvazia o buffer do teclado*
	
	System.out.printf("Insira uma cadeia de carateres:\n");
	s = ler.**nextLine**();

[3.8]	Ler um caráter usando o método **read()**, do pacote de classes **System.in**:
	public static void main(String args[]) **throws** Exception {
	  char c;
	
	  System.out.printf("Insira um Caráter: ");
	  c = (**char**)System.in.**read**();
	}

Exemplo de contagem de tokens numa string (transcreve o programa e executa-o para observares o output)

```
import java.util.Scanner;
public class ContaTokens {
        public static void main(String[] args) {
                int i = 0;
                Scanner sc = new Scanner(System.in);
                System.out.print("Digite um texto:");
                while(sc.hasNext()){
                        i++;
                        System.out.println("Token: "+sc.next());
                }
                sc.close(); //Encerra o objeto Scanner
        }
}
```

Método printf()

Outra novidade no Java 1.5 foi a incorporação da função **printf()** do **C** como um método do pacote de classes **System.out**.
O método **printf()**, utilizado para realizar uma saída de dados no fluxo de saída padrão **System.out**, tem a seguinte forma geral:

```
System.out.printf(expressão_de_controle, argumento1, argumento2, ...);
```

A *expressão_de_controle* deve ser uma sequência de carateres (portanto, delimitada por aspas duplas) que determina as informações que serão mostradas no ecrã. Nesta expressão podem existir dois tipos de informações: carateres comuns e códigos de controlo (ou especificadores de formato). Os códigos de controlo, mostrados na **Tabela 1**, são precedidos por um **%** e são aplicados aos argumentos na mesma ordem em que aparecem. É importante destacar que deve existir para cada código de controlo um argumento correspondente.

Código	Formato (tipo de dados)
%c	Caractere simples (char)
%s	Cadeia de caracteres (String)
%d	Inteiro decimal com sinal (int)
%i	Inteiro decimal com sinal (int)
%ld	Inteiro decimal longo (long)
%f	Real em ponto flutuante (float ou double)
%e	Número real em notação científica com o "e" minúsculo (float ou double)
%E	Número real em notação científica com o "E" maiúsculo (float ou double)
%%	Imprimir o próprio caráter%

Tabela 1. Códigos de controlo ou especificadores de formato.

Aplicações Java
Para demonstrar a utilização dos métodos da classe **Scanner** do pacote **java.util** em operações de entrada de dados e do método **printf()** na formatação da saída através da utilização de códigos de controle foram implementadas as seguintes aplicações Java:

Aplicação 1. Tabuada de um número.

```
import java.util.Scanner; // importar a classe Scanner

public class Exemplo1 {
  public static void main(String[] args) {
//  instanciar e criar um objeto Scanner
    Scanner ler = new Scanner(System.in);
    int i, n;
    System.out.printf("Indique o número para a tabuada:\n");
```

```
        n = ler.nextInt(); // entrada de dados (lendo um valor inteiro)
        System.out.printf("\n+--Resultado--+\n");
        for (i=1; i<=10; i++) {
          System.out.printf("| %2d * %d = %2d |\n", i, n, (i*n));
        }
        System.out.printf("+-------------+\n");
      }
    }
```

Aplicação 2. Calcular o Índice de Massa Corporal (IMC).

```
import java.util.Scanner; // 1. Importar a classe Scanner
public class Exemplo2 {
  public static void main(String args[]) {
// instanciar e criar um objeto Scanner
    Scanner ler = new Scanner(System.in);
    double pc, alt, vlrIMC;
    System.out.printf("Indique o seu  Peso em kg: ");
    pc = ler.nextDouble(); //  entrada de dados (ler um valor real)
    System.out.printf("Indique a sua Altura em metros...: ");
    alt = ler.nextDouble(); // entrada de dados (ler um valor real)
    System.out.printf("\n=======================================\n");
    vlrIMC = IMC(pc, alt);
    System.out.printf("IMC = %.2f (%s)\n", vlrIMC, interpretacaoIMC(vlrIMC));
    System.out.printf("=======================================\n");
  }
  public static double IMC(double pc, double alt) {
    return(pc / (alt * alt));
  }
  public static String interpretacaoIMC(double vlrIMC) {
    if (vlrIMC < 20)
       return("Magro");
    else if ((vlrIMC >= 20) && (vlrIMC <= 24))
        return("Normal");
         else if ((vlrIMC >= 25) && (vlrIMC <= 29))
                 return("Acima do peso");
               else if ((vlrIMC >= 30) && (vlrIMC <= 34))
                   return("Obeso");
               else // acima de 34
                     return("Muito obeso");
  }
}
```

Aplicação 3. Entrada de dados do tipo caráter.

```
import java.io.IOException; //importar a classe IOException
import java.util.Scanner; // importar a classe Scanner
    public class Exemplo3 {
    // através da cláusula throws indicamos que não iremos
    // tratar possíveis erros na entrada de dados realizada
    // através do método "read" do pacote de classes System.in

      public static void main(String[] args) throws IOException {
    //  instanciar e criar um objeto Scanner
        Scanner ler = new Scanner(System.in);
        String nome;
        char genero;
        System.out.printf("Insira um nome:\n");
        nome = ler.nextLine(); // 3.5 entrada de dados (ler uma String)
        System.out.printf("\nInsira o género (M/F): ");
    //  entrada de dados (ler um caráter)
        genero = (char)System.in.read();
        System.out.printf("\nResultado:\n");
        if ((genero == 'M') || (genero == 'm'))
           System.out.printf("Seja bem vindo Sr. \"%s\".\n", nome);
        else
            System.out.printf("Seja bem vinda Sra. \"%s\".\n", nome);
      }
    }
```

Aplicação 4. Mostrar os nomes dos meses.

```
public class Exemplo4 {
  public static void main(String[] args) {
    String nomeMes[] = {"Janeiro", "Fevereiro", "Março",
      "Abril", "Maio", "Junho", "Julho", "Agosto",
      "Setembro", "Outubro", "Novembro", "Dezembro"};

    System.out.println("================");
    System.out.println("Mês- Nome do Mês");
    System.out.println("================");
    for (int i=0; i<12; i++) {
      System.out.printf(" %0,2d- %s\n", (i+1), nomeMes[i]);
    }
    System.out.println("================");
  }
}
```

Exercício 1:

Escreve o código relativo a cada uma das aplicações e executa-o.

Exercício 2:

Cria um projeto que permitirá converter notas quantitativas em qualitativas (usa para ler a nota um método):
Se a nota for inferior a 10 ->Insuficiente
Se a nota for maior ou igual que 10 e menor que 13 -> Suficiente
Se a nota for maior ou igual que 13 e menor que 16 -> Bom
Se a nota for maior ou igual que 16 e menor que 18 -> Muito Bom
Se a nota for maior 18 -> Excelente

A avaliação da nota deverá ser feita numa função com parâmetro de entrada da nota (real) e que devolva uma String.
O nível deverá ser atribuído e a nota qualitativa impressa no ecrã.

Bibliografia:
Objetos e classes em Java - Javatpoint
https://www.w3schools.com/java/java_oop.asp
Jesus, C. (2013). Curso Prático de Java.Lisboa:FCA
Coelho, P (2016). Programação em JAVA – Curso Completo. Lisboa: FCA
https://www.devmedia.com.br/metodos-atributos-e-classes-no-java/25404
https://www.delftstack.com/pt/howto/java/java-word-count/