Faculdade de Design,
Tecnologia e Comunicação
Universidade Europeia

Entrada e Saída em Java

Fundamentos da Programação

Fernando Marson

fernando.marson@universidadeeuropeia.pt

Conceito

- Como apresentado anteriormente, o computador trabalha com a **entrada de informações**, realizando um **processamento** dessas informações e gerando um resultado através de uma **saída**.

- A **entrada** refere-se ao **recebimento de dados** para **processamento**, enquanto a **saída** refere-se à **apresentação dessas informações**.

Entrada e Saída

Por trabalhar com diferentes dispositivos, o processo de **entrada** em **Java** é feito através da **especificação da fonte de entrada** e da **criação de um leitor** para essa fonte (**Scanner**).

- **System.in** é o objeto que está **associado** normalmente ao **teclado.**

- **System.out** é o objeto que está **associado** normalmente à **tela.**

Saída

Em **Java** a **saída na tela** é feita através do objeto **System.out** e possui **três formas distintas**:

- **println( )** imprime mudando para a próxima linha no final
- **print( )** imprime sem mudar para a próxima linha
- **printf( )** imprime o texto seguindo as regras de formatação aplicadas

```
1 System.out.println("Teste de texto.");      //imprime o texto
2 String texto = "Um texto qualquer.";
3 System.out.println(texto);                  //imprime o conteúdo da variável
4 int idade = 35;
5 System.out.println("Idade: " + idade);      //imprime o texto e depois o conteúdo da variável
```

```
Teste de texto.
Um texto qualquer.
Idade: 35
```

Saída

O método **print** do objeto **System.out** imprime na tela, **mantendo o cursor na mesma linha** onde o conteúdo foi impresso:

```
1     System.out.print("Imprimiu");
2     System.out.print("sem mudar de linha.");
```

```
Imprimiusem mudar de linha.
```

Saída

- O método **printf** do objeto **System.out** permite imprimir na tela **formatando a saída**, controlando o número de **casas decimais**, do **alinhamento**, do **preenchimento das posições vazias**, entre outras funcionalidades.

```
1  System.out.printf("'%5.2f'%n", 2.1234567);
2  System.out.printf("'%05.2f'%n", 2.1234567);
3  System.out.printf("'%5.4f'%n", 2.1234567);
4  System.out.printf("'%05.4f'%n", 2.1234567);
5  System.out.printf("'%010.2f'%n", 2.1234567);
6  System.out.printf("'%10.2f'%n", -2.1234567);
7  System.out.printf("'%010.2f'%n", -2.1234567);
8  System.out.printf("'%010.5f'%n", 1234567.12);
9  System.out.printf("'%010.5f'%n", 1234567.1234567);
```

```
' 2.12'
'02.12'
'2.1235'
'2.1235'
'0000002.12'
'     -2.12'
'-000002.12'
'1234567.12000'
'1234567.12346'
```

Saída

Formatação de números inteiros em **Java**:

```java
int numero = 15;
System.out.printf("Número decimal: %d\n", numero);                    // 15
System.out.printf("Número octal: %o\n", numero);                      // 17
System.out.printf("Número hexadecimal: %x\n", numero);                // f
System.out.printf("Número hexadecimal capitalizado: %X\n", numero);   // F
```

Saída:

```
Número decimal: 15
Número octal: 17
Número hexadecimal: f
Número hexadecimal capitalizado: F
```

Saída - Sequências de Escape

- Combinações de caracteres que consistem em uma barra invertida \ **seguida de uma letra** ou de uma **combinação de dígitos** são chamadas de **sequências de escape**.

- Para representar um **caractere de nova linha**, uma **aspas única** ou outros caracteres em **uma constante de caracteres**, é necessário usar **sequências de escape**.

- Uma sequência de escape **é considerada um único caractere** e, portanto, é válida como uma constante de caractere.

Saída - Sequências de Escape

Sequências de Escape	Significado
\t	Tabulação
\n	Nova linha
\’	Aspas simples
\”	Aspas dupla
\\	Barra invertida

Entrada

- Para ler dados em **Java**, utilizamos um leitor chamado **Scanner** atrelado ao objeto **System.in**, que, por padrão, associado ao teclado. Para facilitar a memorização, podemos chamar o objeto do tipo **Scanner** de **input**, **teclado** ou **keyboard**, mas qualquer nome válido irá funcionar.

```java
import java.util.Scanner; // habilita o uso da classe Scanner

public class Main
{
    public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in); /* cria um objeto para ler da
                                                 entrada System.in (teclado) */

        // resto do código
    }
}
```

Entrada

- Em **Java**, cada tipo de dado possui um **método de leitura associado.**

Tipo de dado	Método	Observação
boolean	nextBoolean()	Para qualquer tipo de dado primitivo diferente de char e String, o padrão é sempre o mesmo: nextTIPO().
byte	nextByte()	
short	nextShort()	
int	nextInt()	
long	nextLong()	
float	nextFloat()	
double	nextDouble()	
String	next()	**Retorna** apenas uma **String**, sem espaços em branco.
char	next().charAt(0)	**Método alternativo**, pois **não existe** um método específico para o tipo **char**.
String	nextLine()	**Retorna tudo o que for digitado** até pressionar a tecla **enter**. Quando o **nextLine()** for utilizado após qualquer outro comando de leitura que não seja o próprio **nextLine()** é aconselhável testar o tamanho da **String** lida com o método **length()** da **String**. Se for **zero**, ler novamente. Usar com atenção.

Exemplo - Entrada e Saída

```java
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);

        System.out.print("Informe o seu nome completo: ");  // O nextLine retorna neste uma String com tudo o que foi
        String nome = teclado.nextLine();                   // digitado. Se o nextLine for utilizado antes de outros
        System.out.print("Informe a sua idade: ");          // métodos de leitura, ele funcionará corretamente sempre.
        int idade = teclado.nextInt();
        System.out.print("Informe o seu peso: ");
        float peso = teclado.nextFloat();
        System.out.printf("Seu nome é %s você tem %d anos e pesa %.2f quilos.%n", nome, idade, peso);
    }
}
```