28

Saída formatada

OBJETIVOS

- Neste capítulo, você aprenderá:
- Como entender os fluxos de entrada e saída.
- Como utilizar a formatação do método **`printf`**.
- Como imprimir com larguras e precisão de campo.
- Como utilizar flags de formatação na string de formato **`printf`**.
- Como imprimir com um índice de argumento.
- Como gerar saída de literais e seqüências de escape.
- Como formatar a saída com a classe **`Formatter`**.

Sumário

28.1 Introdução

- **Método `printf`:**
  - **Formata e gera a saída de dados para o fluxo de saída padrão, `System.out`.**
- **Classe `Formatter`:**
  - **Formata e gera a saída de dados para um destino especificado.**
    - **Por exemplo, o fluxo de saída de um arquivo ou string.**

28.2 Fluxos

- **Fluxos:**
  - **Seqüências de bytes.**
  - **Freqüentemente, podem ser redirecionadas.**
    - **Entrada-padrão — teclado.**
    - **Saída-padrão — tela.**
    - **Erro-padrão — tela.**
    - **Mais informações nos capítulos 14 e 24.**

28.3 Formatando a saída com `printf`

- `Printf`:
  - **Formatação precisa da saída.**
    - **Especificações de conversão: flags, larguras de campo, precisões etc.**
  - **Podem realizar:**
    - **vinculação;**
    - **alinhamento de colunas;**
    - **justificação à esquerda/direita;**
    - **inserção de caracteres literais;**
    - **formato exponencial;**
    - **formato octal e hexadecimal;**
    - **largura fixa e precisão; e**
    - **formato de data e hora.**

28.3 Formatando a saída com `printf` (*Continuação*)

- **String de formato:**
  - **Descreve o formato de saída.**
  - **Consiste em texto fixo e no especificador de formato.**
- **Especificador de formato:**
  - **Marcador de lugar para um valor.**
  - **Especifica o tipo dos dados na saída.**
  - **Começa com um sinal de porcentagem (%) e é seguido por um caractere de conversão.**
    - **Por exemplo, `%s`, `%d`.**
  - **Informações de formação opcional:**
    - **Índice de argumentos, flags, largura de campo, precisão.**
    - **Especificadas entre % e o caractere de conversão.**

28.4 Imprimindo inteiros

- **Inteiro:**
  - **Número inteiro (nenhum ponto de fração decimal):**
  - `25, 0, -9`
  - **Negativo, positivo ou zero.**
  - **Somente o sinal de subtração é impresso por padrão (mais tarde alteraremos isso).**
- **Formato:**
  - `printf(` ***string de formato, lista de argumentos*** `);`
    - ***string de formato:***
      - **Descreve o formato de saída.**
    - ***lista de argumentos:***
      - **Contém o valor que corresponde a cada especificador de formato.**

Caractere de conversão	Descrição
d	Exibe um inteiro decimal (base 10).
o	Exibe um inteiro octal (base 8).
x ou X	Exibe um inteiro hexadecimal (base 16). O X faz com que os dígitos de 0–9 e as letras de A a F sejam exibidas e o x exibe os dígitos entre 0 e 9 e as letras entre a e f.

Figura 28.1 | Caracteres de conversão de inteiros.

Resumo

```
1  // Fig. 28.2: IntegerConversionTest.java
2  // Usando os caracteres integrais de conversão.
3
4  public class IntegerConversionTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%d\n", 26 );
9        System.out.printf( "%d\n", +26 );
10       System.out.printf( "%d\n", -26 );
11       System.out.printf( "%o\n", 26 );
12       System.out.printf( "%x\n", 26 );
13       System.out.printf( "%X\n", 26 );
14    } // fim do main
15 } // fim da classe IntegerConversionTest
```

Gera saída de inteiros positivos e negativos

Gera saída de inteiros no formato

Gera saída de inteiros no formato de hexadecimal com letras maiúsculas

```
26
26
-26
32
1a
1A
```

IntegerConversionT

10

a 11

13

Saída do programa

28.5 Imprimindo números de ponto flutuante

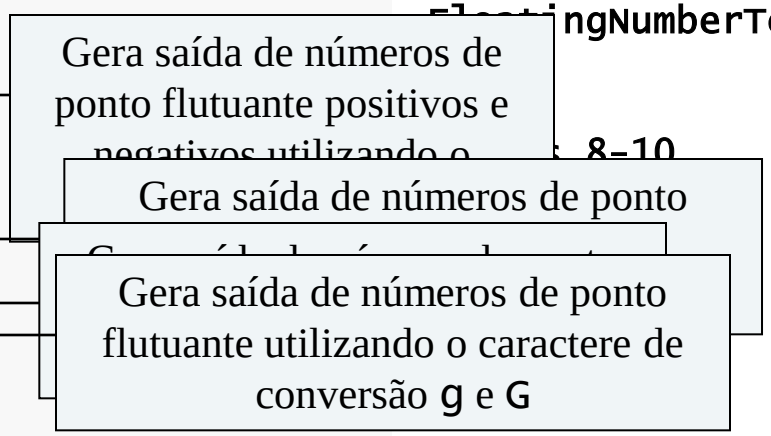
- **Números de ponto flutuante:**
  - **Tem um ponto de fração decimal (`33.5`).**
  - **Notação científica computadorizada (notação exponencial):**
    - `150.4582` **é** `1.504582 x 10²` **em notação científica.**
    - `150.4582` **é** `1.504582e+02` **em notação exponencial (`e` significa expoente).**
    - **Utiliza `e` ou `E`.**
  - `f` **— imprime números de pontos flutuante com pelo menos um dígito à esquerda da fração decimal.**
  - `g` **(ou `G`) — imprime em `f` ou `e` (`E`) .**
    - **Utiliza a exponencial se a magnitude for menor que 10^{-3} ou maior ou igual a 10^{7} .**

Caractere de conversão	Descrição
`e` ou `E`	Exibe um valor de ponto flutuante em notação exponencial. Quando o caractere de conversão `E` é utilizado, a saída é exibida em letras maiúsculas.
`F`	Exibe um valor de ponto flutuante no formato decimal.
`g` ou `G`	Exibe um valor de ponto flutuante no formato de ponto flutuante `f` ou no formato exponencial e com base na magnitude do valor. Se a magnitude for menor que 10–3, maior ou igual a 10^7, o valor de ponto flutuante será impresso com e (ou `E`). Caso contrário, o valor é impresso no formato `f`. Quando o caractere de conversão `G` é utilizado, a saída é exibida em letras maiúsculas.
`a` ou `A`	Exibe um número de ponto flutuante no formato hexadecimal. Quando o caractere de conversão `A` é utilizado, a saída é exibida em letras maiúsculas.

Figura 28.3 | Caracteres de conversão de ponto flutuante.

Resumo

```
1  // Fig. 28.4: FloatingNumberTest.java
2  // Utilizando caracteres de conversão de ponto flutuante.
3
4  public class FloatingNumberTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%e\n", 12345678.9 );
9        System.out.printf( "%e\n", +12345678.9 );
10       System.out.printf( "%e\n", -12345678.9 );
11       System.out.printf( "%E\n", 12345678.9 );
12       System.out.printf( "%f\n", 12345678.9 );
13       System.out.printf( "%g\n", 12345678.9 );
14       System.out.printf( "%G\n", 12345678.9 );
15    } // fim do main
16 } // fim da classe FloatingNumberTest
```



Saída do programa

```
1.234568e+07
1.234568e+07
-1.234568e+07
1.234568E+07
12345678.900000
1.23457e+07
1.23457E+07
```

28.6 Imprimindo strings e caracteres

- **Caractere de conversão `c` e `C`:**
  - **Requer `char`.**
  - **`C` exibe a saída em letras maiúsculas.**
- **Caractere de conversão `s` e `S`:**
  - `String`
  - `Object`
    - **Utiliza implicitamente o método `toString` do objeto.**
  - **`S` exibe a saída em letras maiúsculas.**

Erro comum de programação 28.1

Utilizar `%c` para imprimir uma string resulta em uma `IllegalFormatConversionException` — uma string não pode ser convertida em um caractere.

Resumo

```
1  // Fig. 28.5: CharStringConversion.java
2  // Utilizando caractere e caracteres de conversão de string.
3
4  public class CharStringConversion
5  {
6     public static void main( String args[] )
7     {
8        char character = 'A';  // inicializa char
9        String string = "This is also a string";  // objeto String
10       Integer integer = 1234;  // inicializa inteiro (autoboxing)
11
12       System.out.printf( "%c\n", character );
13       System.out.printf( "%s\n", "This is a string" );
14       System.out.printf( "%s\n", string );
15       System.out.printf( "%S\n", string );
16       System.out.printf( "%s\n", integer ); // chamada i
17    } // fim do main
18 } // fim da classe CharStringConversion
```

Exibe o caractere com o caract

Exibe a string com o ersão s

Exibe string com os caracteres de conversão s e S

Exibe o objeto Integer com os caracteres de conversão

```
A
This is a string
This is also a string
THIS IS ALSO A STRING
1234
```

CharStringConversion.java

Linha 12

a 13

5

16

Saída do programa

28.7 Imprimindo datas e horas

- **Caracteres de conversão `t` e `T`:**
  - **Imprimem datas e horas em vários formatos.**
  - **Seguidos por um caractere de sufixo de conversão.**
  - **Requerem que o argumento correspondente seja do tipo `long`, `Long`, `Calendar` ou `Date`.**
- **Caracteres de sufixo de conversão:**
  - **Especificam o formato de data e/ou data/hora.**
  - **Formatam composições de data e hora.**
  - **Formatam de data.**
  - **Formatam de data/hora.**

Caractere de sufixo de conversão	Descrição
c	Exibe a data e hora formatadas como `day month date hour:minute:second time-zone year` com os três caracteres para `day` e `month`, dois dígitos para `date`, `hour`, `minute` e `second` e quatro dígitos para `year` — por exemplo, `Wed Mar 03 16:30:25 GMT-05:00 2004`. O relógio de 24 horas é utilizado. Neste exemplo, `GMT-05:00` é o fuso horário.
F	Exibe a data formatada como `year-month-date` com quatro dígitos para o `year` e dois dígitos cada para `month` e a `date` (por exemplo, `2004-05-04`).
D	Exibe a data formatada como `month/day/year` com dois dígitos cada para o `month`, `day` e `year` (por exemplo, 03/03/04).
r	Exibe a hora formatada como `hour:minute:second AM|PM` com dois dígitos cada para a `hour`, `minute` e `second` (por exemplo, `04:30:25 PM`). O relógio de 12 horas é utilizado.
R	Exibe a hora formatada como `hour:minute` com dois dígitos cada para a `hour` e `minute` (por exemplo, `16:30`). O relógio de 24 horas é utilizado.
T	Exibe a hora formatada como `hour:minute:second` com dois dígitos para o `hour`, `minute` e `second` (por exemplo, `16:30:25`). O relógio de 24 horas é utilizado.

Figura 28.6 | Caractere de sufixo de conversão para composição de data e hora.

Caractere de sufixo de conversão	Descrição
`A`	Exibe o nome completo do dia da semana (por exemplo, `Wednesday`).
`a`	Exibe o nome abreviado com três caracteres do dia da semana (por exemplo, `Wed`).
`B`	Exibe o nome completo do mês (por exemplo, `March`).
`b`	Exibe o nome abreviado com três caracteres do mês (por exemplo, `Mar`).
`d`	Exibe o dia do mês com dois dígitos, preenchendo com zeros à esquerda conforme necessário (por exemplo,`03`).
`m`	Exibe o mês com dois dígitos, preenchendo com zeros à esquerda conforme necessário (por exemplo, `07`).
`e`	Exibe o dia de mês sem zeros à esquerda (por exemplo, `3`).
`Y`	Exibe o ano com quatro dígitos (por exemplo, `2004`).
`y`	Exibe os dois últimos dígitos do ano com zeros à esquerda conforme necessário (por exemplo, `04`).
`j`	Exibe o dia do ano com três dígitos, preenchendo com zeros à esquerda conforme necessário (por exemplo, `016`).

Figura 28.7 | Caractere de sufixo de conversão de formatação de data.

Caractere de sufixo de conversão	Descrição
`H`	Exibe horas no relógio de 24 horas com um zero à esquerda conforme necessário (por exemplo, `16`).
`I`	Exibe horas no relógio de 12 horas com um zero à esquerda conforme necessário (por exemplo, `04`).
`k`	Exibe horas em relógio de 24 horas sem zeros à esquerda (por exemplo, `16`).
`l`	Exibe horas em relógio de 12 horas sem zeros à esquerda (por exemplo, `4`).
`M`	Exibe minutos com um zero à esquerda conforme necessário (por exemplo, `06`).
`S`	Exibe segundos com um zero à esquerda conforme necessário (por exemplo, `05`).
`Z`	Exibe a abreviação para o fuso horário (por exemplo, `GMT-05:00`, significa Eastern Standard Time, que está 5 horas atrás do Greenwich Mean Time).
`P`	Exibe marcador de manhã ou de tarde em letras minúsculas (por exemplo, `pm`).
`p`	Exibe marcador de manhã ou de tarde em letras maiúsculas (por exemplo, `PM`).

Figura 28.8 | Caractere de sufixo de conversão de formatação de tempo.

Resumo

DateTimeTest.java

(1 de 2)

```
1  // Fig. 28.9: DateTimeTest.java
2  // Formatando datas e horas com o caractere de conversão t e T.
3  import java.util.Calendar;
4
5  public class DateTimeTest
6  {
7     public static void main( String args[] )
8     {
9        // obtém a data e hora atual
10       Calendar dateTime = Calendar.getInstance();
11
12       // imprimindo com caracteres de conversão para composições de data/hora
13       System.out.printf( "%tc\n", dateTime );
14       System.out.printf( "%tF\n", dateTime );
15       System.out.printf( "%tD\n", dateTime );
16       System.out.printf( "%tr\n", dateTime );
17       System.out.printf( "%tT\n", dateTime );
18
19       // imprimindo com caracteres de conversão para data
20       System.out.printf( "%1$tA, %1$tB %1$td, %1$tY\n", dateTime );
21       System.out.printf( "%1$TA, %1$TB %1$Td, %1$TY\n", dateTime );
22       System.out.printf( "%1$ta, %1$tb %1$te, %1$ty\n", dateTime );
23
```

Obtém um `Calendar` com a data e hora atuais

Utiliza o objeto `Calendar` nas instruções `printf` como o valor a ser formatado com o caractere de conversão `t`

Utiliza o índice de argumento opcional para indicar que todos os especificadores de formato na string de formato utilizam o primeiro argumento

Resumo

```
24       // imprimindo com caracteres de conversão para hora
25       System.out.printf( "%1$tH:%1$tM:%1$tS\n", dateTime );
26       System.out.printf( "%1$tZ %1$tI:%1$tM:%1$tS %tP", dateTime );
27    } // fim de main
28 } // fim da classe DateTimeTest
```

```
Tue Jun 29 11:17:21 GMT-05:00 2004
2004-06-29
06/29/04
11:17:21 AM
11:17:21
Tuesday, June 29, 2004
TUESDAY, JUNE 29, 2004
Tue, Jun 29, 04
11:17:21
GMT-05:00 11:17:21 AM
```

DateTimeTest.java

(2 de 2)

Saída do programa

28.8 Outros caracteres de conversão

- **Caracteres de conversão restantes:**
  - `b` **ou** `B`
    - **Valor** `boolean` **ou** `Boolean`**.**
  - `h` **ou** `H`
    - **Representação de string do código de hash de um objeto no formato hexadecimal.**
  - **%**
    - **Caractere de porcentagem.**
  - **n**
    - **Separador de linha específico de plataforma.**
      - `\r\n` **no Windows**
      - `\n` **no UNIX\Linux**

Erro comum de programação 28.2

Tentar imprimir um caractere de porcentagem literal utilizando % em vez de %% na string de formato resultaria em um erro de lógica difícil de detectar. Quando % aparece em uma string de formato, ele deve ser seguido por caractere de conversão na string. O caractere de porcentagem único poderia acidentalmente ser seguido por um caractere de conversão legítimo, resultando, assim, em um erro de lógica.

Caractere de conversão	Descrição
`b` ou `B`	Imprime `"true"` ou `"false"` para o valor de um `boolean` ou `Boolean`. Esses caracteres de conversão também podem formatar o valor de qualquer referência. Se a referência for não-`null`, `"true"` será impresso; do contrário, `"false"` será impresso. Quando o caractere de conversão `B` é utilizado, a saída é exibida em letras maiúsculas.
`h` ou `H`	Imprime a representação de string de um valor de código de hash do objeto em formato hexadecimal. Se o argumento correspondente for uma referência `null`, `"null"` será impresso. Quando o caractere de conversão `H` é utilizado, a saída é exibida em letras maiúsculas.
`%`	Imprime o caractere de porcentagem.
`n`	Imprime o separador de linha específico à plataforma (por exemplo, `\r\n` no Windows ou `\n` no UNIX/LINUX).

Figura 28.10 | Outros especificadores de conversão.

Resumo

```
1  // Fig. 28.11: OtherConversion.java
2  // Utilizando os caracteres de conversão b, B, h, % e n.
3
4  public class OtherConversion
5  {
6     public static void main( String args[] )
7     {
8        Object test = null;
9        System.out.printf( "%b\n", false );
10       System.out.printf( "%b\n", true );
11       System.out.printf( "%b\n", "Test" );
12       System.out.printf( "%B\n", test );
13       System.out.printf( "Hashcode of \"hello\" is %h\n", "hello" );
14       System.out.printf( "Hashcode of \"Hello\" is %h\n", "Hello" );
15       System.out.printf( "Hashcode of null is %H\n", test );
16       System.out.printf( "Printing a %% in a format str
17       System.out.printf( "Printing a new line %nnext li
18    } // fim do main
19 } // fim da classe OtherConversion
```

```
false
true
true
FALSE
Hashcode of "hello" is 5e918d2
Hashcode of "Hello" is 42628b2
Hashcode of null is NULL
Printing a % in a format string
Printing a new line
next line starts here
```

OtherConversion.java

Linhas 9-10

Linhas 11-12

Linhas 13-14

Imprime o valor de valores **boolean** **false** e

Associa uma **String** e um objeto **null** a **%b** e **%B**

Imprime as rep... do código de ... para as strings "Hello" e "**Hello**"

Imprime **null** em letras maiúsculas

Imprime o caractere % em uma string e um separador de linha específico de plataforma

28.9 Imprimindo com larguras e precisões de campos

- **Largura do campo:**
  - **Tamanho do campo em que os dados são impressos.**
  - **Se a largura for maior que os dados, é alinhado à direita por padrão.**
    - **Se a largura do campo for menor, aumenta para que os dados se ajustem.**
    - **O sinal de subtração utiliza uma posição de caractere no campo.**
  - **Largura de inteiro inserida entre % e o especificador de conversão.**
    - **Por exemplo, %4d — largura de campo de 4.**
  - **Pode ser utilizado com todos os especificadores de formato, exceto o separador de linha (%n).**

28.9 Imprimindo com larguras e precisões de campos (*Cont.*)

- **Precisão:**
  - **Significado varia dependendo do tipo de dado.**
  - **Ponto flutuante:**
    - **Número de dígitos que aparece depois da fração decimal (`e` ou `E` e `f`).**
    - **Número máximo de dígitos significativos (`g` ou `G`).**
  - **Strings:**
    - **Número máximo de caracteres a ser gravado na string.**
  - **Formato:**
    - **Utiliza um ponto (`.`) e, então, o número de precisão depois de `%`.**
      - **Por exemplo, `%.3f`.**

28.9 Imprimindo com larguras e precisões de campos (*Continuação*)

- **Largura e precisão de campo:**
  - **Podem ser especificadas:**
    - `%width.precision`
      - `%5.3f`
  - **Largura de campo negativa — dados alinhados à esquerda.**
  - **Largura positiva de campo — dados alinhados à direita.**
  - **A precisão deve ser positiva.**
    - **Exemplo:**
      - `printf( "%9.3f", 123.456789 );`

Erro comum de programação 28.3

Não fornecer uma largura de campo suficientemente grande para tratar um valor a ser impresso pode deslocar outros dados sendo impressos e produz saídas confusas. Conheça seus dados!

Resumo

```
1  // Fig. 28.12: FieldWidthTest.java
2  // Alinhando inteiros à direita nos campos.
3
4  public class FieldWidthTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%4d\n", 1 );
9        System.out.printf( "%4d\n", 12 );
10       System.out.printf( "%4d\n", 123 );
11       System.out.printf( "%4d\n", 1234 );
12       System.out.printf( "%4d\n\n", 12345 ); // dados muito grandes
13
14       System.out.printf( "%4d\n", -1 );
15       System.out.printf( "%4d\n", -12 );
16       System.out.printf( "%4d\n", -123 );
17       System.out.printf( "%4d\n", -1234 ); // dados muito grandes
18       System.out.printf( "%4d\n", -12345 ); // dados muito grandes
19    } // fim do main
20 } // fim da classe RightJustifyTest
```

Imprime números positivos com a largura de campo

Imprime números negativos com a largura de campo

```
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345
```

FieldWidthTest.java

Linhas 8-12

Linhas 14-18

Saída do programa

Resumo

```
1  // Fig. 28.13: PrecisionTest.java
2  // Utilizando a precisão para números de ponto flutuante e strings.
3  public class PrecisionTest
4  {
5     public static void main( String args[] )
6     {
7        double f = 123.94536;
8        String s = "Happy Birthday";
9
10       System.out.printf( "Using precision for floating-point numbers\n" );
11       System.out.printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );
12
13       System.out.printf( "Using precision for strings\n" );
14       System.out.printf( "\t%.11s\n", s );
15    } // fim do main
16 } // fim da classe PrecisionTest
```

PrecisionTest.java

Linha 11

Imprime o mesmo número de ponto flutuante com a mesma precisão, mas um caractere de versão diferente

Imprime uma string com precisão

```
Using precision for floating-point numbers
        123.945
        1.239e+02
        124

Using precision for strings
        Happy Birth
```

Saída do programa

28.10 Utilizando flags na string de formato `printf`

- **Flags:**
  - **Podem ser utilizados com o método `printf` para suplementar suas capacidades de formatação de saída.**
  - **Para utilizar um flag em uma string de formato, posicione o flag imediatamente à direita do sinal de porcentagem.**
  - **Vários flags podem ser combinados.**

Flag	Descrição
- (sinal de subtração)	Alinha a saída à esquerda dentro do campo especificado.
+ (sinal de adição)	Exibe um sinal de adição precedendo valores positivos e um sinal de subtração precedendo valores negativos.
Espaço	Imprime um espaço antes de um valor positivo não impresso com o flag +.
#	O prefixo 0 para o valor de saída quando utilizado com o caractere de conversão octal o. Adiciona o prefixo 0x ao valor de saída quando utilizado com o caractere de conversão hexadecimal x.
0 (zero)	Preenche um campo com zeros à esquerda.
, (vírgula)	Utiliza o separador específico de localidade de milhares (isto é, ',' para localidade nos Estados Unidos) para exibir números decimais e de ponto flutuante.
(	Inclui números negativos dentro de parênteses.

Figura 28.14 | Flags de string de formato.

Resumo

```
1  // Fig. 28.15: MinusFlagTest.java
2  // Valores para o alinhamento à direita e à esquerda.
3
4  public class MinusFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.println( "Columns:" );
9        System.out.println( "01234567890123456789012345678901234567890123456789\n" );
10       System.out.printf( "%10s%10d%10c%10f\n\n", "hello", 7, 'a', 1.23 );
11       System.out.printf(
12          "%-10s%-10d%-10c%-10f\n", "hello", 7, 'a', 1.23 );
13    } // fim do main
14 } // fim da classe MinusFlagTest
```

```
Columns:
01234567890123456789012345678901234567890123456789

     hello         7         a  1.230000

hello     7         a         1.230000
```

MinusFlagTest java

Linha 10

Alinha à direita uma string, um inteiro, um caractere e um número de ponto flutuante

Alinha à esquerda uma string, um inteiro, um caractere e um número de ponto flutuante

Saída do programa

Resumo

PlusFlagTest.java

```
1  // Fig. 28.16: PlusFlagTest.java
2  // Imprimindo números com e sem o flag +.
3
4  public class PlusFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%d\t%d\n", 786, -786 );
9        System.out.printf( "%+d\t%+d\n", 786, -786 );
10    } // fim do main
11 } // fim da classe PlusFlagTest
```

Imprime um número positivo com um sinal de adição

Saída do programa

```
786     -786
+786    -786
```

Resumo

```
1  // Fig. 28.17: SpaceFlagTest.java
2  // Imprimindo um espaço antes de valores não-negativos.
3
4  public class SpaceFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "% d\n% d\n", 547, -547 );
9     } // fim do main
10 } // fim da classe SpaceFlagTest
```

Prefixa um espaço para o número positivo com o flag de espaço.

SpaceFlagTest.java

```
 547
-547
```

Saída do programa

Resumo

PoundFlagTest.java

Linhas 10-11

```
1  // Fig. 28.18: PoundFlagTest.java
2  // Utilizando o flag # com os caracteres de conversão o e x.
3
4  public class PoundFlagTest
5  {
6     public static void main( String args[] )
7     {
8        int c = 31;       // inicializa o c
9
10       System.out.printf( "%#o\n", c );
11       System.out.printf( "%#x\n", c );
12    } // fim do main
13 } // fim da classe PoundFlagTest
```

Utiliza o flag # para prefixar 0 para o valor octal e 0x para o valor hexadecimal

Saída do programa

```
037
0x1f
```

Resumo

```
1  // Fig. 28.19: ZeroFlagTest.java
2  // Imprimir com flags 0 (zero) preenche com zeros à esquerda.
3
4  public class ZeroFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%+09d\n", 452 );
9        System.out.printf( "%09d\n", 452 );
10       System.out.printf( "% 9d\n", 452 );
11    } // fim do main
12 } // fim da classe ZeroFlagTest
```

Combina o flag + e o flag 0,

Imprime 452 em um campo

Imprime 452 em um campo de largura 9 utilizando somente o flag de espaço

```
+00000452
000000452
      452
```

Resumo

```
1  // Fig. 28.20: CommaFlagTest.java
2  // Using the comma (,) flag to display numbers with thousands separator.
3
4  public class CommaFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%,d\n", 58625 );
9        System.out.printf( "%,.2f", 58625.21 );
10       System.out.printf( "%,.2f", 12345678.9 );
11    } // end main
12 } // end class CommaFlagTest
```

CommaFlag

Utiliza o flag vírgula para exibir um número decimal e um número de ponto flutuante com o separador de milhar.

```
58,625
58,625.21
12,345,678.90
```

Saída do programa

Resumo

```
1  // Fig. 28.21: ParenthesesFlagTest.java
2  // Utilizando o flag ( para colocar números negativos entre parênteses.
3
4  public class ParenthesesFlagTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf( "%(d\n", 50 );
9        System.out.printf( "%(d\n", -50 );
10       System.out.printf( "%(.1e\n", -50.0 );
11    } // fim do main
12 } // fim da classe ParenthesesFlagTest
```

Inclui números negativos entre parênteses utilizando o flag (

ParenthesesFlagTes
t.java

s 8-10

Saída do programa

```
50
(50)
(5.0e+01)
```

28.11 Imprimindo com índices de argumento

- **Índice de argumentos:**
  - **Inteiro decimal opcional seguido por um sinal de $.**
  - **Indica a posição do argumento na lista de argumentos.**
    - **Por exemplo, 1$ — primeiro argumento.**
  - **Uso:**
    - **Reordena a saída.**
    - **Evita argumentos duplicados.**

Resumo

```
1  // Fig. 28.22: ArgumentIndexTest
2  // Reordenando a saída com índices de argumentos.
3
4  public class ArgumentIndexTest
5  {
6     public static void main( String args[] )
7     {
8        System.out.printf(
9           "Parameter list without reordering: %s %s %s %s\n",
10          "first", "second", "third", "fourth" );
11       System.out.printf(
12          "Parameter list after reordering: %4$s %3$s %2$s %1$s\n",
13          "first", "second", "third", "fourth" );
14    } // fim do main
15 } // fim da classe ArgumentIndexTest
```

```
Parameter list without reordering: first second third fourth
Parameter list after reordering: fourth third second first
```

ArgumentIndexTest

Linhas 11-13

Imprime os argumentos da lista de argumentos na ordem inversa utilizando o índice de argumentos

28.12 Imprimindo literais e seqüências de escape

- **Imprimindo literais:**
  - **A maioria dos caracteres pode ser impressa.**
  - **Há certos caracteres ‘problemáticos’, como aspas ("), que delimitam a própria string de formato.**
  - **Vários caracteres de controle devem ser representados por seqüências de escape.**
    - **Uma seqüência de escape é representada por uma barra invertida (\) seguida por um caractere de escape.**

Erro comum de programação 28.4

Tentar imprimir como dados literais em uma instrução `printf` um caractere de aspa dupla ou de barra invertida sem preceder esse caractere com uma barra invertida para formar uma seqüência de escape adequada resultaria em um erro de sintaxe.

Seqüência de escape	Descrição
`\'` (aspas simples)	Gera saída do caractere de aspa simples (`'`).
`\"` (aspas duplas)	Gera a saída do caractere de aspas duplas (`"`).
`\\` (barras invertidas)	Gera a saída do caractere de barra invertida (`\`).
`\b` (backspace)	Move o cursor de volta uma posição na linha atual.
`\f` (nova página ou avanço de formulário)	Move o cursor para o início da próxima página lógica.
`\n` (nova linha)	Move o cursor para o começo da próxima linha.
`\r` (retorno de carro)	Move o cursor para o começo da linha atual.
`\t` (tabulação horizontal)	Move o cursor para a próxima posição da tabulação horizontal.

Figura 28.23 | Seqüências de escape.

28.13 Formatar a saída com a classe `Formatter`

- Classe `Formatter`:
  - Fornece as mesmas capacidades de formatação como `printf`.
  - Gera a saída dos dados formatados para um destino especificado.
    - Por exemplo, um arquivo em disco.
  - Por padrão, `Formatter` cria uma string na memória.
- Método `String static format`:
  - Cria uma string na memória sem `Formatter`.

Resumo

FormatterTest.java

Linha 11

Linha 12

```
1  // Fig. Fig. 28.24: FormatterTest.java
2  // String de formato com a classe Formatter.
3  import java.util.Formatter;
4  import javax.swing.JOptionPane;
5
6  public class FormatterTest
7  {
8     public static void main( String args[] )
9     {
10       // cria Formatter e formata a saída
11       Formatter formatter = new Formatter();
12       formatter.format( "%d = %#o = %#X", 10, 10, 10 );
13
14       // exibe a saída em JOptionPane
15       JOptionPane.showMessageDialog( null, formatter.toString() );
16    } // fim do main
17 } // fim da classe FormatterTest
```

Cria um objeto `Formatter` utilizand
que con
memóri

Invoca o método `format` para formatar a saída

Invoca o método `Formatter` `toString` para obter os dados formatados como uma string