


```

35     else
36         System.out.println("Sorry, no match");
37     }
38 }


```

Enter your lottery pick (two digits): 00 

The lottery number is 00

Exact match: you win \$10,000




Enter your lottery pick (two digits): 45 

The lottery number is 54

Match all digits: you win \$3,000




Enter your lottery pick: 23 

The lottery number is 34

Match one digit: you win \$1,000



Enter your lottery pick: 23 

The lottery number is 14

Sorry: no match



The program generates two random digits and concatenates them into the string **lottery** (lines 6–7). After this, **lottery** contains two random digits.

The program prompts the user to enter a guess as a two-digit string (line 12) and checks the guess against the lottery number in this order:

- First check whether the guess matches the lottery exactly (line 25).
- If not, check whether the reversal of the guess matches the lottery (line 27).
- If not, check whether one digit is in the lottery (lines 30–33).
- If not, nothing matches and display “Sorry, no match” (line 36).

4.6 Formatting Console Output

You can use the **System.out.printf** method to display formatted output on the console.



Often, it is desirable to display numbers in a certain format. For example, the following code computes interest, given the amount and the annual interest rate.

```

double amount = 12618.98;
double interestRate = 0.0013;
double interest = amount * interestRate;
System.out.println("Interest is $" + interest);

```

Interest is \$16.404674



Because the interest amount is currency, it is desirable to display only two digits after the decimal point. To do this, you can write the code as follows:

```
double amount = 12618.98;
double interestRate = 0.0013;
double interest = amount * interestRate;
System.out.println("Interest is $"
    + (int)(interest * 100) / 100.0);
```

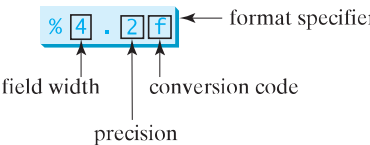


Interest is \$16.4

printf

However, the format is still not correct. There should be two digits after the decimal point: **16.40** rather than **16.4**. You can fix it by using the **printf** method, like this:

```
double amount = 12618.98;
double interestRate = 0.0013;
double interest = amount * interestRate;
System.out.printf("Interest is $%.2f",
    interest);
```



Interest is \$16.40

The syntax to invoke this method is

```
System.out.printf(format, item1, item2, ..., itemk)
```

format specifier

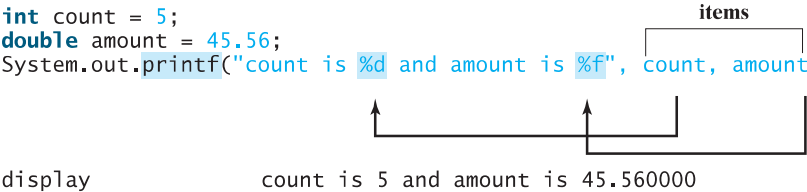
where **format** is a string that may consist of substrings and format specifiers. A *format specifier* specifies how an item should be displayed. An item may be a numeric value, a character, a Boolean value, or a string. A simple format specifier consists of a percent sign (%) followed by a conversion code. Table 4.11 lists some frequently used simple format specifiers.

TABLE 4.11 Frequently Used Format Specifiers

Format Specifier	Output	Example
%b	a Boolean value	true or false
%c	a character	'a'
%d	a decimal integer	200
%f	a floating-point number	45.460000
%e	a number in standard scientific notation	4.556000e+01
%s	a string	"Java is cool"

Here is an example:

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);
```



Items must match the format specifiers in order, in number, and in exact type. For example, the format specifier for **count** is **%d** and for **amount** is **%f**. By default, a floating-point value is displayed with six digits after the decimal point. You can specify the width and precision in a format specifier, as shown in the examples in Table 4.12.

TABLE 4.12 Examples of Specifying Width and Precision

Example	Output
%5c	Output the character and add four spaces before the character item, because the width is 5.
%6b	Output the Boolean value and add one space before the false value and two spaces before the true value.
%5d	Output the integer item with width at least 5. If the number of digits in the item is < 5, add spaces before the number. If the number of digits in the item is > 5, the width is automatically increased.
%10.2f	Output the floating-point item with width at least 10 including a decimal point and two digits after the point. Thus, there are 7 digits allocated before the decimal point. If the number of digits before the decimal point in the item is < 7, add spaces before the number. If the number of digits before the decimal point in the item is > 7, the width is automatically increased.
%10.2e	Output the floating-point item with width at least 10 including a decimal point, two digits after the point and the exponent part. If the displayed number in scientific notation has width less than 10, add spaces before the number.
%12s	Output the string with width at least 12 characters. If the string item has fewer than 12 characters, add spaces before the string. If the string item has more than 12 characters, the width is automatically increased.

If an item requires more spaces than the specified width, the width is automatically increased. For example, the following code

```
System.out.printf("%3d#%2s#%4.2f\n", 1234, "Java", 51.6653);
```

displays

```
1234#Java#51.67
```

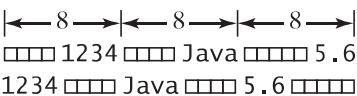
The specified width for **int** item **1234** is **3**, which is smaller than its actual size **4**. The width is automatically increased to **4**. The specified width for string item **Java** is **2**, which is smaller than its actual size **4**. The width is automatically increased to **4**. The specified width for **double** item **51.6653** is **4**, but it needs width 5 to display 51.67, so the width is automatically increased to **5**.

By default, the output is right justified. You can put the minus sign (-) in the format specifier to specify that the item is left justified in the output within the specified field. For example, the following statements

right justify
left justify

```
System.out.printf("%8d%8s%8.1f\n", 1234, "Java", 5.63);  
System.out.printf("%-8d%-8s%-8.1f \n", 1234, "Java", 5.63);
```

display



where the square box (□) denotes a blank space.

**Caution**

The items must match the format specifiers in exact type. The item for the format specifier `%f` or `%e` must be a floating-point type value such as `40.0`, not `40`. Thus, an `int` variable cannot match `%f` or `%e`.

**Tip**

The `%` sign denotes a format specifier. To output a literal `%` in the format string, use `%%`.

Listing 4.6 gives a program that uses `printf` to display a table.

LISTING 4.6 FormatDemo.java

```

1  public class FormatDemo {
2      public static void main(String[] args) {
3          // Display the header of the table
4          System.out.printf("%-10s%-10s%-10s%-10s%-10s\n", "Degrees",
5                          "Radians", "Sine", "Cosine", "Tangent");
6
7          // Display values for 30 degrees
8          int degrees = 30;
9          double radians = Math.toRadians(degrees);
10         System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
11                         radians, Math.sin(radians), Math.cos(radians),
12                         Math.tan(radians));
13
14         // Display values for 60 degrees
15         degrees = 60;
16         radians = Math.toRadians(degrees);
17         System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees,
18                         radians, Math.sin(radians), Math.cos(radians),
19                         Math.tan(radians));
20     }
21 }

```

display table header

values for 30 degrees

values for 60 degrees



Degrees	Radians	Sine	Cosine	Tangent
30	0.5236	0.5000	0.8660	0.5773
60	1.0472	0.8660	0.5000	1.7320

The statement in lines 4–5 displays the column names of the table. The column names are strings. Each string is displayed using the specifier `%-10s`, which left-justifies the string. The statement in lines 10–12 displays the degrees as an integer and four float values. The integer is displayed using the specifier `%-10d` and each float is displayed using the specifier `%-10.4f`, which specifies four digits after the decimal point.



4.22 What are the format specifiers for outputting a Boolean value, a character, a decimal integer, a floating-point number, and a string?

4.23 What is wrong in the following statements?

- (a) `System.out.printf("%5d %d", 1, 2, 3);`
- (b) `System.out.printf("%5d %f", 1);`
- (c) `System.out.printf("%5d %f", 1, 2);`