A floating-point value is a number that can have a fractional part (e.g. 3.14). A floating-point variable must be declared as either `float` or `double`. Seasoned Java programmers use `double` unless memory conservation is an issue. The sizes of these data types are shown below. Java stores floating-point data in binary according to the IEEE 754 specification.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size</th>
<th>Can Hold a Value up to</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float</code></td>
<td>32 bits</td>
<td>approximately 7 significant digits</td>
</tr>
<tr>
<td><code>double</code></td>
<td>64</td>
<td>approximately 15 significant digits</td>
</tr>
</tbody>
</table>

Floating-point literals are typed as a sequence of digits, preceded by an optional `+` or `−` sign, with the fractional part identified by a decimal point. The underscore (`_`) can be used as a group separator.

```
Example

double a = 150.;
double b = +150.;
double c = -.015;
double d = 1_000;
```

Although it is not required by Java’s grammar, seasoned programmers emphasize the decimal point by typing a digit on both of its sides.

```
Example

double a = 150.0;
double b = +150.0;
double c = -0.015;
double d = 1_000.0;
```
You can use a suffix on the floating-point literal to indicate its data type – D for a `double` or F for `float`. Lacking either, the literal is taken to be a `double` value.

**Example**

```java
double a = 1_024.0D;
float b = 1_024.0F;
```

**Example**

```java
float x = 1_024.0;
```

Wrong!  The 64-bit literal `1_024.0` may not be stored in the 32-bit variable `x`.

A floating-point literal may use *scientific notation*.

**Examples**

```java
double a = -1.5E2;
double b = -1.5E-2;
double c = +1E+6;
float d = +1.5E2F;
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>-150.0</td>
</tr>
<tr>
<td>b</td>
<td>-0.015</td>
</tr>
<tr>
<td>c</td>
<td>1000000.0</td>
</tr>
<tr>
<td>d</td>
<td>150.0</td>
</tr>
</tbody>
</table>

**Exercises**

Write the Java statement for each.

1. Declare `d1` a `double` initialized to negative one million using scientific notation.
2. Declare `d2` a `double` initialized to negative one million without using scientific notation.
3. Declare `d3` a `double` initialized to one millionth using scientific notation.
4. Declare `d4` a `double` initialized to one millionth without using scientific notation.
5. Declare `f1` a `float` initialized to one million using scientific notation.
6. Declare `f2` a `float` initialized to one million without using scientific notation.