Motivations
Suppose that you need to print a string (e.g., "Welcome to Java") a hundred times.

System.out.println("Welcome to Java!");

How do you solve this problem?

Objectives
- To use while, do-while, and for loop statements to control the repetition of statements (§ 4.2-4.4).
- To know the similarities and differences between three types of loops (§ 4.5).
- To write nested loops (§ 4.6).
- To learn the techniques for minimizing numerical errors (§ 4.7).
- To implement program control with break and continue (§ 4.9).
- (GUI) To control a loop with a confirmation dialog (§ 4.10).
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}

(count < 2) is still true since count is 1

Increase count by 1

(count < 2) is false since count is 2 now
Trace while Loop

```java
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

The loop exits. Execute the next statement after the loop.

Ending a Loop with a Sentinel Value

Often the number of times a loop is executed is not predetermined. You may use an input value to signify the end of the loop. Such a value is known as a sentinel value.

Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

```java
SentinalValue.java
```

Problem: Guessing Numbers

Write a program that randomly generates an integer between 0 and 100, inclusive. The program prompts the user to enter a number. The program then tells the user whether the input is correct, too low or too high.

Modify the above program so that it prompts the user to enter a number continuously until the number matches the randomly generated number. For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently.

```java
GuessNumberOneTime.java
GuessNumber.java
```

Problem: An Advanced Math Learning Tool

The Math subtraction learning tool program generates just one question for each run. You can use a loop to generate questions repeatedly. This example gives a program that generates five questions and reports the number of the correct answers after a student answers all five questions.

```java
SubtractionQuizLoop.java
```

do–while Loop

```java
do {
    // Loop body;
    Statement(s);
} while (loop-continuation-condition);
```

Sentinel Example

Use do-while loop to rewrite the program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

```java
SentinelValueDoWhile.java
```
for Loops

for (initial-action; loop-continuation-condition; action-after-each-iteration) {
    // loop body;
    Statement(s);
}

for (int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}

Trace for Loop

int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

Execute initializer
i is now 0

Print Welcome to Java

Execute adjustment statement
i now is 1
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

(i < 2) is still true since i is 1

Print Welcome to Java

(i < 2) is false since i is 2

Exit the loop. Execute the next statement after the loop

Note
The initial-action and action-after-each-iteration in a for loop can be a list of zero or more comma-separated expressions.

```java
for (int i = 1; i < 100; System.out.println(i++));

for (int i = 0, j = 0; (i + j < 10); i++, j++) {
    // Do something
}
```
Note

If the loop-continuation-condition in a for loop is omitted, it is implicitly true.

```java
for (; ; ) {
    // Do something
}
```

Equivalent

```java
while (true) {
    // Do something
}
```

Better!

---

Minimizing Numerical Errors

- Floating-point numbers are represented approximately in the computer
- Avoid using floating point numbers in the loop condition

Example: compute the sum for a series of numbers: 0.01, 0.02, ..., 1.0.

TestSum.java

---

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

Logic Error

```java
for (int i=0; i<10; i++) {
    System.out.println("i is " + i);
}
```

Correct

```java
for (int i=0; i < 10; i++) {
    System.out.println("i is " + i);
}
```

---

Compare floating-point numbers

Don't use == or != to compare floating-point numbers

```java
double data = Math.pow(Math.sqrt(2), 2) - 2;
if (data == 0) // wrong
    System.out.println("data is zero");
else
    System.out.println("data is not zero");
```

---

Comparing Floating-Point Numbers

To compare floating-point numbers test whether they are close enough:

```
|x - y| ≤ ε
```

where ε is a small number such as $10^{-14}$

```java
final double EPSILON = 1E-14;
if (Math.abs(x - y) <= EPSILON) // x is approximately equal to y
```
Which Loop to Use?

- while and for loops are pre-test loops, do-while are post-test loops
- The three loops are expressively equivalent.

A while loop in (a) below can always be converted into the for loop in (b):

```
while (loop-continuation-condition) {
  // Loop body;
}
```

A for loop in (a) below can generally be converted into the while loop in (b) except in certain special cases (see Review Question 3.19 for one of them):

```
for (initial-action;
     loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
}
```

Recommendations

- Use the one that is most intuitive and comfortable for you.
- A for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
- A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

Nested Loops

- Outer loop for triangle rows
- Inner loop for triangle columns for each row

```
for (int i = 1; i < n; i++) {
  for (int j = 1; j <= i; j++)
    System.out.print("* ");
  System.out.println();
}
```

Modify the nested loops to print a square instead of a triangle?

Example

Write a program that uses nested for loops to print a multiplication table.

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
</tr>
</tbody>
</table>
```

How to: Implementing Loops

- Step 1. List the work that needs to be done in every step of the loop body
- Step 2. Find out how often the loop is repeated and where we can determine the loop is finished – determine whether to use while or for loops
- Step 3. Implement the loop by putting the operations from Step 1 into the loop body
- Step 4. Double check variable initializations and updates and check for infinite loop and off-by-one errors

Problem: Finding the Greatest Common Divisor

Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor. Suppose you enter two integers 4 and 2, their greatest common divisor is 2. Suppose you enter two integers 16 and 24, their greatest common divisor is 8.

How do you find the greatest common divisor?

Let the two input integers be $n_1$ and $n_2$. You know number 1 is a common divisor, but it may not be the greatest common divisor. So you can check whether $k$ (for $k = 2, 3, 4$, and so on) is a common divisor for $n_1$ and $n_2$, until $k$ is greater than $n_1$ or $n_2$.

```
GreatestCommonDivisor.java
```
Problem: Finding the Sales Amount

Problem: You have just started a sales job in a department store. Your pay consists of a base salary and a commission. The base salary is $5,000. The scheme shown below is used to determine the commission rate.

<table>
<thead>
<tr>
<th>Sales Amount</th>
<th>Commission Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.01–$5,000</td>
<td>8 percent</td>
</tr>
<tr>
<td>$5,000.01–$10,000</td>
<td>10 percent</td>
</tr>
<tr>
<td>$10,000.01 and above</td>
<td>12 percent</td>
</tr>
</tbody>
</table>

Your goal is to earn $30,000 in a year. Write a program that will find out the minimum amount of sales you have to generate in order to make $30,000.

FindSalesAmount.java

Examples

- Using break
  Adds the integers from 1 to 20 in this order until the sum is greater than or equal to 100
  TestBreak.java

- Using continue
  Adds all integers from 1 to 20 except 10 and 11.
  TestContinue.java

Problem: Displaying a Pyramid of Numbers

Problem: Write a program that prompts the user to enter an integer from 1 to 15 and displays a pyramid. For example, if the input integer is 12, the output is shown below.

PrintPyramid.java

Guessing Number Problem Revisited

Rewrite the guessing number program using a break statement.

The program prompts the user to enter a number continuously until the number matches the randomly generated number. For each user input, the program tells the user whether the input is too low or too high, so the user can choose the next input intelligently.

GuessNumberUsingBreak.java

Using break and continue

- break – ends the innermost loop that contains it. Breaks out of a loop. (Can be used in loops and switch statements)
- continue – ends the current iteration. Continues to the next iteration.
- Normally used with if statement to break or continue the loop in a certain condition
- You can always write a program using loops without break and continue

Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:

- Start with number 1
- Determine whether the number is prime
- Print the number if it is prime
- Count the prime numbers so far.
- If count < 50, repeat the above steps

PrimeNumber.java
Displaying Prime Numbers - Algorithm

Set an initial count to 0 (to track the number of prime numbers)
Set an initial number to 2
while (count < 50) {
    test whether number is prime;
    if number is prime
        print the number;
        update count;
}

to test whether a number is prime, check whether it is divisible by 2, 3, 4, up to number/2.

(GUI) Controlling a Loop with a Confirmation Dialog

A sentinel-controlled loop can be implemented using a confirmation dialog. The answers Yes or No to continue or terminate the loop. The template of the loop may look as follows:

```
int option = 0;
while (option == JOptionPane.YES_OPTION) {
    System.out.println("continue loop");
    option = JOptionPane.showConfirmDialog(null, "Continue?");
}
```

SentinelValueUsingConfirmationDialog.java