Loops

Topics

- The *while* loop.
- Auto-increment and auto-decrement operators.
- The *for* loop.
- The *do-while* loop
- The *for-each* loop
Flow Charts

if (test)
    statement;

Flow Charts (2)

if (test)
    s1;
else
    s2;

if (test)
    s1;
else
    s2;
The **while** Loop

```java
while (test) {
    statement;
}
```

```java
public double compound(double balance, double apr, int term) {
    int years = 0;
    while (years < term) {
        years += 1;
        balance += balance * apr;
    } // end while
    return balance;
} // end compound
```

- This is an example of **counter controlled repetition**.
- The loop is executed *term* times.
- The variable `years` is the loop counter.

**Compound Interest**

- Problem: Given an initial balance and interest rate, what is the balance after some years of compound interest?

```java
public double compound(double balance, double apr, int term) {
    int years = 0;
    while (years < term) {
        years += 1;
        balance += balance * apr;
    } // end while
    return balance;
} // end compound
```
Compound Interest (2)

- Problem: Given an initial balance and interest rate, how many years will it take before the balance reaches a certain amount?

```java
public int term(double balance, double apr, double goal) {
    int years = 0;
    while (balance < goal) {
        years += 1;
        balance += balance * apr;
    }
    return years;
}
```

- This is an example of sentinel value controlled repetition.
- The loop is executed until a specified balance (the sentinel) is reached.
- The variable `goal` is the sentinel value.

Counting Characters

- Problem: How many times does a given character occur within a String?

```java
public int charCount(String s, char c) {
    int length = s.length();
    int index = 0;
    int count = 0;
    while (index < length) {
        if (s.charAt(index) == c) {
            count += 1;
        }
        index += 1;
    }
    return count;
}
```

- This is another example of counter controlled repetition.
- The loop is executed `length` times (once for each character in the string).
- In this case, `index` is the loop counter.
Finding Words Within a String

```java
public void printWords(String phrase) {
    int start = 0;
    int end = 0;
    int last = phrase.length();

    while (start < last) {
        // find beginning of next word (skip over spaces)
        while (start < last && phrase.charAt(start) == ' ') {
            start += 1;
        }

        if (start < last) {  // found a word
            end = start + 1;
            while (end < last && phrase.charAt(end) != ' ')
                end += 1;
            String word = phrase.substring(start, end);
            System.out.println(word);
            start = end;
        }
    }
}
```

The Auto-Increment Operator: `++`

- The autoincrement operator increases the value of a variable by 1.
- It works on all the primitive types except boolean.
- The following are all equivalent ways to increment a number by 1:

```java
number = number + 1;
nombre += 1;
nombre++;
// post-increment form
++numero;
// pre-increment form
```
Pre-increment vs. Post-increment

- When the increment is part of a larger expression, the two forms produce different results.
- With the pre-increment form, the variable’s value is incremented before its value is used in the expression.
- With the post-increment form, the variable’s value is incremented after its value is used in the expression.

Auto-increment Examples

<table>
<thead>
<tr>
<th></th>
<th>(i)</th>
<th>(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbf{int (i) = 0;}</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>\textbf{int (j) = 0;}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(i++);</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(++i;)</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>(i = j++);</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(i = ++j;)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(i = j++ + 3;)</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>(i = ++j + 3;)</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>(i = i++;)</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>(j = ++i + i;)</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>(j = i + ++i;)</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>(j = i++ + i;)</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>(j = i + i++;)</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>

In binary ops, the left operand is evaluated before the right.

Post (pre) increment is applied immediately after (before) taking the value.
Auto-decrement Operator, --

- The pre- and post-decrement operations are similar to pre- and post-increment.

<table>
<thead>
<tr>
<th>Expression</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int i = 10;</code></td>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td><code>int j = 10;</code></td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td><code>i--;</code></td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td><code>--i;</code></td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td><code>i = j--;</code></td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td><code>i = --j;</code></td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td><code>i = j-- + 3;</code></td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td><code>i = --j + 3;</code></td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>

The for Loop

```
for (expr1; expr2; expr3)
statement;
```

- Initialization
- Re-Initialization
- Test
- Re-initialization
- Statement
- Usually a compound statement.

- expr1
- expr3
- expr2
- ?
- true
- false
- Usually a compound statement.
Counter Controlled Repetition using for

- Problem: Given an initial balance and interest rate, what is the balance after some years of compound interest?

```java
public double compound(double balance, double apr, int term) {
    for (int years = 0; years < term; years++) {
        balance += balance * apr;
    }
    return balance;
}
```

- A variable declared in a for statement (e.g. years) is only in scope within the for statement.

Scope of Variables Declared in For

- Problem: Given an initial balance and interest rate, how many years will it take before the balance reaches a certain amount?

```java
public int term(double balance, double apr, double goal) {
    int years;  // Must be declared outside of loop
    for (years = 0; balance < goal; years++) {
        balance += balance * apr;
    }
    return years;
}
```

- To fix the error, declare variable years before the for statement.

statement.
Optional Expressions

- In a for statement all three expressions (initialization, test, and re-initialization) are all optional.
  - Both semicolons are required.
  - If the test is omitted, the loop behaves as if the test is always true.

Optional Expression Examples

```
for ( ; test; ) {
  ;
}
```

is equivalent to

```
while (test) {
  ;
}
```

```
for ( ; ; ) {
  ;
}
```

is equivalent to

```
while (true) {
  ;
}
```
break and continue statements

- When a break statement is executed inside a loop, the loop is terminated and control passes to the next statement after the loop.
- When a continue statement is executed inside a loop, the *current iteration* of the loop is terminated and control passes back to the top of the loop.

break Examples

- What do the following code fragments do?

```java
for (int i = 0; i < 10; i++) {
    if (i == 5)
        break;
    System.out.println(i);
}
```

Print values: 0, 1, 2, 3, 4

```java
int j = 0;
while (j < 10) {
    if (j == 5)
        break;
    System.out.println(j++);
}
```

Print values: 0, 1, 2, 3, 4
continue Examples

➢ What do the following code fragments do?

```java
for (int i = 0; i < 10; i++) {
    if (i == 5)
        continue;
    System.out.println(i);
}
```

Print values: 0, 1, 2, 3, 4, 6, 7, 8, 9

```java
int j = 0;
while (j < 10) {
    if (j == 5)
        continue;
    System.out.println(j);
}
```

Print values: 0, 1, 2, 3, 4 then loop forever!

Fix the while loop to behave like for

```java
int j = 0;
while (j++ < 10) {
    if (j == 5)
        continue;
    System.out.println(j);
}
```

Print values: 1, 2, 3, 4, 6, 7, 8, 9, 10

```java
int j = 0;
while (j < 10) {
    if (j++ == 5)
        continue;
    System.out.println(j);
}
```

Print values: 1, 2, 3, 4, 5, 7, 8, 9, 10

```java
int j = -1;
while (j++ < 10) {
    if (j == 5)
        continue;
    System.out.println(j);
}
```

Print values: 0, 1, 2, 3, 4, 6, 7, 8, 9, 10
Fix the while loop to behave like for (2)

```java
int j = 0;
while (++j < 10) {
    if (j == 5)
        continue;
    System.out.println(j);
}
Print values: 1, 2, 3, 4, 6, 7, 8, 9
```

```java
int j = 0;
while (j < 10) {
    if (++j == 5)
        continue;
    System.out.println(j);
}
Print values: 1, 2, 3, 4, 6, 7, 8, 9, 10
```

```java
int j = -1;
while (++j < 10) {
    if (j == 5)
        continue;
    System.out.println(j);
}
Print values: 0, 1, 2, 3, 4, 6, 7, 8, 9
```

The do-while Loop

```java
do
    statement;
while (test);
```

Usually a compound statement.

Differs from a while loop in that the statement is executed once before the test.

This is usually wrong!
Common Errors

- Failure to move toward termination condition in a loop.
  - E.g. failure to advance loop counter

- Off-by-one errors
  - Execute loop one time too many or one time too few.
  - Start and end loop one unit too high or one unit too low.
    - E.g. execute loop for values 1 through 10 instead of 0 through 9, or vice versa.

- Always check the boundary cases carefully – what happens on first iteration and last iteration.

- Same for conditionals – what happens when the condition is just barely true or just barely false.