While and Do-While Loops

15-110 Summer 2010
Margaret Reid-Miller
Loops

• Within a method, we can alter the flow of control using either conditionals or loops.

• The loop statements `while`, `do-while`, and `for` allow us execute a statement(s) over and over.

• Like a conditional, a loop is controlled by a boolean expression that determines how many times the statement is executed.

E.g., You may want to calculate the interest paid on a mortgage for each year of the loan term.
The while statement

- The form of the `while` statement is
  ```
  while (<boolean_expression>)
  <statement>
  ```
- If `boolean_expression` evaluates to `true`, then `statement` is executed.
- Then, the `boolean_expression` is evaluated again. If it evaluates to `true`, `statement` is executed again.
- This repetition continues until the `boolean_expression` evaluates to `false`.

How is the while loop different from the if statement?
The if Flowchart

- boolean_expression
- true
- statement (body of loop)
- false
The **while** Flowchart

![Diagram of the while flowchart with the boolean expression node, true path leading to the statement (body of loop) box, and false path leading back to the boolean expression node.](diagram.png)
A while Example

Print n asterisks

```
int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
}
System.out.println();
```

<table>
<thead>
<tr>
<th>i</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>**</td>
</tr>
<tr>
<td>2</td>
<td>***</td>
</tr>
<tr>
<td>3</td>
<td>****</td>
</tr>
<tr>
<td>4</td>
<td>*****</td>
</tr>
<tr>
<td>5</td>
<td>******</td>
</tr>
</tbody>
</table>

n = 5
The Loop Control Variable

• The variable $i$ (known as the loop control variable) is used in three ways: it is *initialized, tested*, and *updated*.

```java
int i = 0;  // initialize
while (i < n) {  // test
    System.out.print("*");
    i++;  // update
}
System.out.println();
```

• All three things must be coordinated in order for the loop to work correctly!
Off-by-1 Errors

```
int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
}
System.out.println();
```

For n = 5 the output is
***** (5 asterisks)

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

Output?
Off-by-1 Errors

```java
int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
}
System.out.println();
```

For `n = 5` the output is `*****` (5 asterisks)

Output?
Warning!

int i = 0;
while (i < n) {
    System.out.print("*");
    i--;
}
System.out.println();
Infinite Loops

```java
int i = 0;
while (i < n) {
    System.out.print("*");
    i--;
}
System.out.println();
```

Do you know which company has this address?

Apple Computer
1 Infinite Loop
Cupertino, CA 95014
A while Example

int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
}
System.out.println();
Exercise

• Write a method with a while loop to prints 1 through n in square brackets. For example, if n = 6 print

Exercise: Cumulative Sum

- Write a method with a while loop that computes the sum of first n positive integers:
  \[ \text{sum} = 1 + 2 + 3 + \ldots + n \]

Examples:

- \( n = 5 \) \quad \text{sum} = 15
- \( n = 19 \) \quad \text{sum} = 190
Exercise: Fencepost Loop

• Write a method with a while loop that prints 1 through n, separated by commas. E.g., for n = 9 print
  1, 2, 3, 4, 5, 6, 7, 8, 9
The **do** Statement

- The form of the **do** statement is
  
  ```
  do
  <statement>
  while (<boolean_expression>);
  ```

- First, *statement* is executed.
- Then, the *boolean_expression* is evaluated. If it evaluates to **true**, *statement* is executed again.
- This repetition continues until the *boolean_expression* evaluates to **false**.
The do Flowchart

true

false

boolean_expression

statement

false
Example

```java
int i = 0; // initialize
do {
    System.out.print("*");
i++;
} while (i < n); // test
System.out.println();
```

For n = 7 what is the output?
How is it different from the while loop?
User Input

Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the month [1-12]: ");
int month = keyboard.nextInt();

What if the user enters a month outside the range?
User Input (cont’d)

- Use a do-while loop to test whether a user has entered data of the correct form and, if not, ask repeatedly until the data entered is correct.

```java
Scanner keyboard = new Scanner(System.in);
int month;
do {
    System.out.print(
        "Please enter the month [1-12]: ");
    month = keyboard.nextInt();
} while ( month < 1 || month > 12 );
```

- Must be declared outside the loop
- Outside the scope of the loop
User Input

• Sometimes it is easier to think of what you want the input to be and negate.

Scanner keyboard = new Scanner(System.in);
int month;
do {
    System.out.print("Please enter the month [1-12]: ");
    month = keyboard.nextInt();
} while (!(month >= 1 && month <= 12));

Use de Morgan’s law to prove the Boolean expressions are the same!
Sentinel Controlled Loops

• Suppose you want to find the maximum of the data entered from the keyboard.
• It is not known in advanced how many data values a user might want to enter. (And the user may not want to count them!)
• A *sentinel* is a special value that is used to detect a special condition, in this case that the user is done entering values.
• The sentinel, of course, must be distinct from any value the user may want to input.
Sentinel Example

Scanner console = new Scanner(System.in);
System.out.print("Enter count (enter -1 to quit): ");
int count = console.nextInt();
int maxSoFar = count;

while (count != -1) {
    if (count > maxSoFar) maxSoFar = count;

    System.out.print("Enter count (enter -1 to quit): ");
    count = console.nextInt();
}

if (maxSoFar > -1)
    System.out.println("The maximum is " + maxSoFar);
else
    System.out.println("No counts entered");