JAVA Textbook

Chapter 5

Writing Java Programs Using Loops

Objectives

In this chapter, you will learn about:

- Increment and decrement operators in Java.
- While loops in Java.
- For loops in Java.
- Do-while loops in Java.
- Nested loops in Java.
- Using loops to
  - Accumulate totals.
  - Validate user input.

Increment and Decrement Operators

- Increment operator: ++
  - Adding 1 to an value
  - *number++;* is equivalent to: *number = number + 1;*
- Decrement operator: --
  - Subtracting 1 from an value
  - *Number--;* is equivalent to: *number = number - 1;*

Increment and Decrement Operators

```
x = 5;
y = x++;  // Postfix form
          // y is assigned the value of x,
          // then x is incremented.
          // Value of x is 6.
          // Value of y is 5.
```

```
x = 5;
y = x;
x = x + 1;
```

```
x = 5;
y = ++x;  // Prefix form
          // x is incremented first, then
          // the value of x is assigned to y.
          // Value of x is 6.
          // Value of y is 6.
```

```
x = 5;
x = x + 1;
y = x;
```

- Prefix form: ++*number* or *--number*
  - Value is incremented/decremented BEFORE being used
- Postfix form: *number*++ or *number*--
  - Value is incremented/decremented AFTER being

While Loops in Java

- Three essential steps:
  - Initialize a loop control variable.
  - Compare the loop control variable to a sentinel value to decide whether the loop continues or stops.
  - Alter the value of the loop control variable within the loop.

While Loops in Java

```
while(expression)
    statement;
```

```
final int NUM_TIMES = 3;
int num = 0;
while(num < NUM_TIMES)
{
   System.out.println("Welcome to Java Programming.");
   num++;
}
```

- Loop body may consist of a single statement or a block statement.
  - A block statement is several statements within a pair of curly braces.
- No semicolon after the ending parenthesis.
  - Placing a semicolon after the ending parenthesis is not a syntax error, but a logic error, which results in an infinite loop, “while the condition is true, do nothing forever.”

While Loops in Java

```
final int NUM_TIMES = 3;
int num = 0;
while(num < NUM_TIMES)
{
    System.out.println("Welcome to Java Programming.");
    num++;
}
```

```
final int NUM_TIMES = 3;
int num = 0;
while(num++ < NUM_TIMES)
    System.out.println("Welcome to Java Programming.");
```

- When postfix increment operator is used, the value of num is not incremented until after the comparison is made.
- If the prefix increment operator is used in the expression ++num < NUM_TIMES, the loop executes twice instead of three times.

Counter-Controlled Loops

```
int count = 0;
while(count < 4)
{
    System.out.println("Hello");
    count++;
}
```

- With a counter, you set up the loop to execute a specified number of times.

Loops Controlled by Sentinel Value

Priming read:

- Performed before a loop executes to input a value used to control the loop.
- Must perform another read within the loop body to get the next input value.

```
name = JOptionPane.showInputDialog(
       "Enter employee's name or XXX to quit: ");
while(name.compareTo(QUIT) != 0)
{
   // This is the work done in the detailLoop() method
   grossString = JOptionPane.showInputDialog(
                 "Enter employee's gross pay: ");
   gross = Double.parseDouble(grossString);
   deduct = gross * RATE;
   net = gross - deduct;
   System.out.println("Name: " + name);
   System.out.println("Gross Pay: " + gross);
   System.out.println("Deductions: " + deduct);
   System.out.println("Net Pay: " + net);
   name = JOptionPane.showInputDialog(
          "Enter employee's name or XXX to quit: ");
}
// This is the work done in the endOfJob() method
System.out.println(END_LINE);
```

For Loops in Java

```
for(expression1; expression2; expression3)
   statement;
```

```
int number = 0;
int count;
final int NUM_LOOPS = 10;
for(count = 0; count < NUM_LOOPS; count++)
{
   number += count;
   System.out.println("Value of number is: " + number);
}
```

- The first time the for loop is encountered, the first expression is used to initialize a loop control variable.
- Next, the second expression is evaluated. If evaluates to true, the loop statement executes; if false, the loop is exited.
- After the loop statement executes, the third expression is evaluated to increment or decrement the loop control variable.
- After the third expression is evaluated, the second expression is evaluated again to determine whether to continue with or to exit the loop.
- Process continues until the second expression evaluates to false

For Loops in Java

```
int counter;
final int NUM_LOOPS = 4;
for(counter = 0; counter < NUM_LOOPS; counter++)
{
   System.out.println("Hello");
}
```

```
int count = 0;
while(count < 4)
{
   System.out.println("Hello");
   count++;
}
```

- A for loop may be implemented by a while loop, and vice versa.

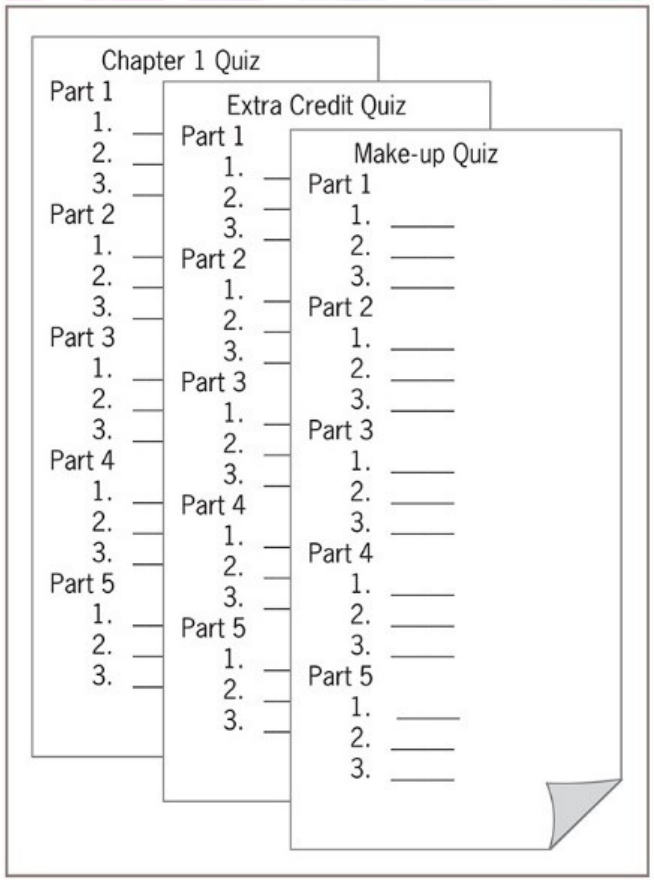
Do-While Loops in Java

```
do
    statement;
while(expression);
```

```
int counter = 0;
final int NUM_LOOPS = 4;
do
{
    System.out.println("Hello");
    counter++;
} while(counter < NUM_LOOPS);
```

- Can be stated as “do *a* while *b* is true.”
- Similar to a while loop, but not the same.
  - With a while loop, condition is tested **before** the loop body is executed.
  - With a do while loop, condition is tested **after** the loop body is executed once.
- Choose a do while loop when logic requires the loop body to execute at least once.

Nested Loops




```
int partCounter;
int questionCounter;
final int PARTS = 5;
final int QUESTIONS = 3;
final String PART_LABEL = "Part ";
final String LINE = ". _____";
partCounter = 1;
while(partCounter <= PARTS)
{
   System.out.println(PART_LABEL + partCounter);
   questionCounter = 1;
   while(questionCounter <= QUESTIONS)
   {
      System.out.println(questionCounter + LINE);
      questionCounter++;
   }
   partCounter++;
}
```

Figure 5-7 Quiz answer sheets

- Include a loop within another loop.
- Must use multiple control variables to control the

Accumulating Totals in Loop

- The loop body includes an accumulator and a counter.
- Java program will not compile if testTotal is not initialized before being used.
- Calculate the

```
String stringNum, stringScore;
int numStudents, stuCount, testScore;
double testTotal, average;
// Get user input to control loop.
stringNum = JOptionPane.showInputDialog(
            "Enter number of students: ");
// Convert number String to int.
numStudents = Integer.parseInt(stringNum);
// Initialize accumulator variable to 0.
testTotal = 0;
// Loop for each student.
for(stuCount = 0; stuCount < numStudents; stuCount++)
{
   // Input student test score.
   stringScore = JOptionPane.showInputDialog(
                 "Enter student's score: ");
   // Convert to integer.
   testScore = Integer.parseInt(stringScore);
   // Accumulate total of test scores.
   testTotal += testScore;
}
// Calculate average test score.
average = testTotal / stuCount;
```

Validating Input Using Loop

- Need to validate input from a user to avoid problems caused by

```
String answer;
answer = JOptionPane.showInputDialog(
         "Do you want to continue? Enter Y or N.");
while((answer.compareTo("Y") != 0) &&
      (answer.compareTo("N") != 0))
{
   answer =
     JOptionPane.showInputDialog(
     "Invalid Response. Please type Y or N.");
}
```

```
String stringAnswer;
int answer;
final int MIN_NUM = 1;
final int MAX_NUM = 4;
stringAnswer = JOptionPane.showInputDialog(
     "Please enter a number in the range of " + MIN_NUM +
     " to " + MAX_NUM + ": ");
answer = Integer.parseInt(stringAnswer);
while(answer < MIN_NUM || answer > MAX_NUM)
{
   stringAnswer = JOptionPane.showInputDialog(
        "Number must be between " + MIN_NUM + " and " +
        MAX_NUM + ". Please try again: ");
   answer = Integer.parseInt(stringAnswer);
}
```

Thank You!