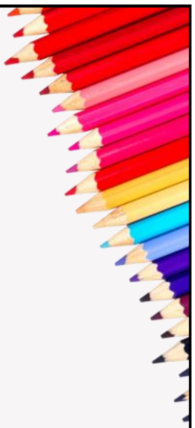
Chapter 5 Loops



Motivations

- Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a hundred times:

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

```
            ┌ System.out.println("Welcome to Java!");
100 times  <  System.out.println("Welcome to Java!");
            │ ...
            └ System.out.println("Welcome to Java!");
```

- So, how do you solve this problem?
- *A loop can be used to tell a program to execute statements repeatedly.*

Objectives (1)

- To write programs for executing statements repeatedly using a **while** loop (§5.2).
- To follow the loop design strategy to develop loops (§§5.3).
- To control a loop with a sentinel value (§5.4).
- To obtain large input from a file using input redirection rather than typing from the keyboard (§5.5).
- To write loops using **do-while** statements (§5.6).
- To write loops using **for** statements (§5.7).
- To discover the similarities and differences of three types of loop statements (§5.8).

Objectives (2)

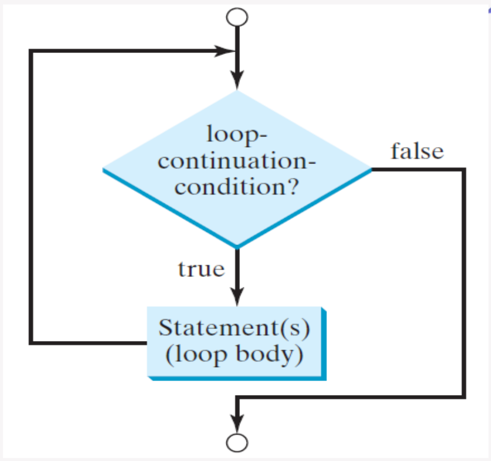
- To write nested loops (§5.9).
- To learn the techniques for minimizing numerical errors (§5.10).
- To learn loops from a variety of examples (**GCD**, **FutureTuition**, **Dec2Hex**) (§5.11).
- To implement program control with **break** and **continue** (§5.12).
- To process characters in a string using a loop in a case study for checking palindrome (§5.13).
- To write a program that displays prime numbers (§5.14).

5.2 The while Loop

- *A while loop executes statements repeatedly while the condition is true.*
- The syntax for the **while loop is:**

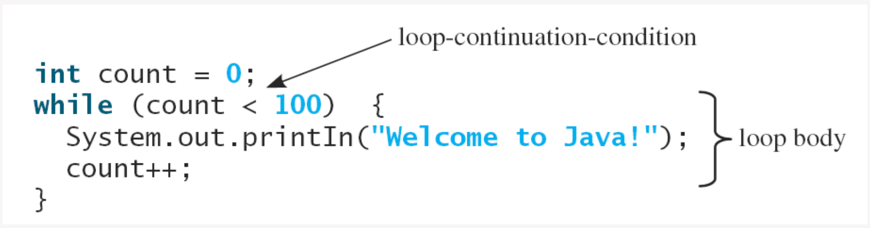
```
while (loop-continuation-condition) {
  // Loop body
  Statement(s);
}
```

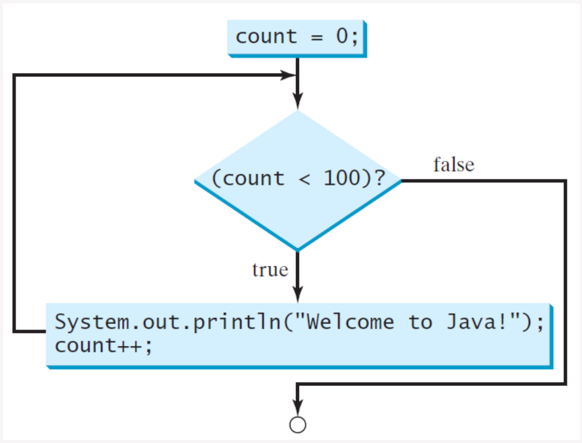
- while Loop Flow Chart



while Loop

```
int count = 0;
while (count < 100)  {
  System.out.printIn("Welcome to Java!");
  count++;
}
```

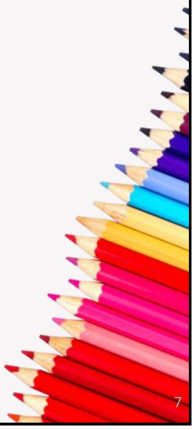




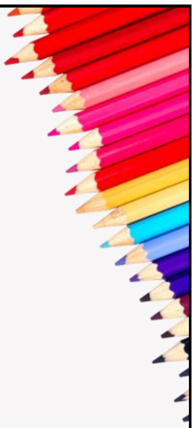
Problem: Repeat Addition Until Correct

- Recall that Listing 3.1 AdditionQuiz.java gives a program that prompts the user to enter an answer for a question on addition of two single digits.
- Using a loop, you can now rewrite the program to let the user enter a new answer until it is correct.

RepeatAdditionQuiz Run



Case Study: Guessing Numbers



- Write a program that randomly generates an integer between 0 and 100, inclusive. 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.
- Here is a sample run:

GuessNumberOneTime Run

GuessNumber Run

5.4 Loop Design Strategies

- *The key to designing a loop is to identify the code that needs to be repeated and write a condition for terminating the loop.*
- Step 1: Identify the statements that need to be repeated.
- Step 2: Wrap these statements in a loop as follows:

```
while (true) {Statements;}
```

- Step 3: Code the **loop-continuation-condition** and add appropriate statements for controlling the loop.

```
while (loop-continuation-condition) {
    Statements;
    Additional statements for controlling the loop;
}
```

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.

SubtractionQuizLoop   Run

10

5.5 Controlling a Loop with User Confirmation or 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*.
- A loop that uses a sentinel value to control its execution is called a *sentinel-controlled loop.*
- Write a program that reads and calculates the sum of an unspecified number of integers. The input 0 signifies the end of the input.

SentinelValue Run

Caution

- Don’t use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results.
- Consider the following code for computing 1 + 0.9 + 0.8 + ... + 0.1:

```
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
  sum += item;
  item -= 0.1;
}
System.out.println(sum);
```

Input and Output Redirections

- If you have a large number of data to enter, it would be cumbersome to type from the keyboard. You can store the data separated by whitespaces in a text file, say **input.txt,** and run the program using the following command:

  java SentinelValue < input.txt

- This command is called ***input redirection***. The program takes the input from the file input.txt rather than having the user type the data from the keyboard at runtime.
- Similarly, there is ***output redirection****, which sends the output to a file rather than displaying* it on the console. The command for output redirection is:

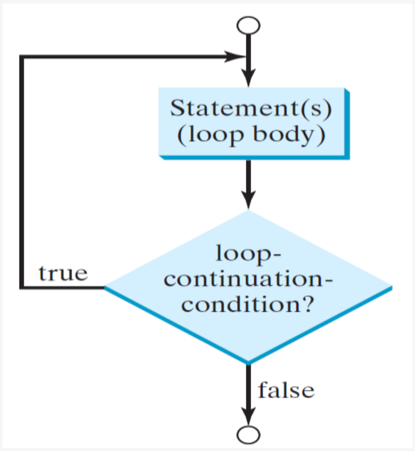
  java ClassName > output.txt

5.6 The do-while Loop

- *A **do-while** loop is the same as a while loop except that it executes the loop body first and then checks the loop continuation condition.*

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

TestDoWhile   Run

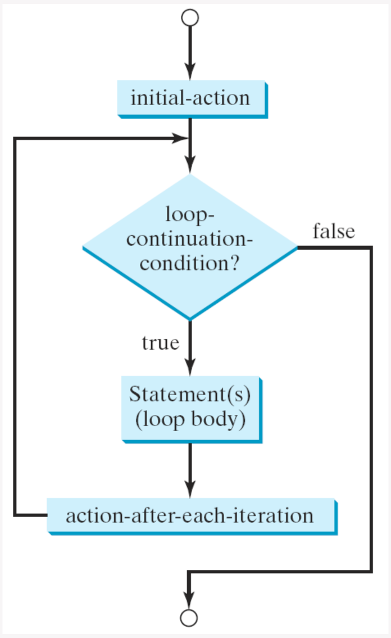


14

5.7 The for Loop

- *A **for loop** has a concise syntax for writing loops.*
- In general, the syntax of a **for** loop is:

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



15

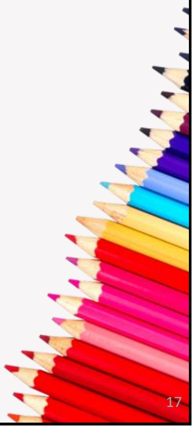
5.8 Which Loop to Use?

- You can use a for loop, a while loop, or a do-while loop, whichever is convenient.
- The while loop and for loop are called *pretest loops* because the continuation condition is checked before the loop body is executed.
- The do-while loop is called a *posttest* loop because the condition is checked after the loop body is executed.
- The three forms of loop statements—while, do-while, and for—are expressively equivalent; that is, you can write a loop in any of these three forms.

16

Recommendations

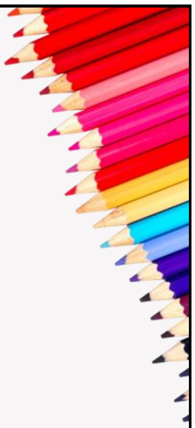
- Use the one that is most intuitive and comfortable for you.
  - In general, a for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
  - A while loop may be used if the number of repetitions is not known, as in the case of reading the 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.

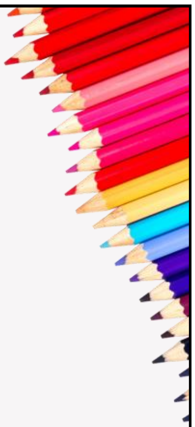


5.9 Nested Loops

- A loop can be nested inside another loop.
- Nested loops consist of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered, and started anew.
- Problem: Write a program that uses nested for loops to print a multiplication table.

MultiplicationTable Run



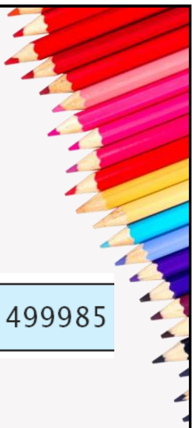


5.10 Minimizing Numeric Errors

- Numeric errors involving floating-point numbers are inevitable. This section discusses how to minimize such errors through an example.
- Here is an example that sums a series that starts with 0.01 and ends with 1.0. The numbers in the series will increment by 0.01, as follows: 0.01 + 0.02 + 0.03 and so on.

TestSum

Run



Minimizing Numeric Errors

```
float sum = 0;

// Add 0.01, 0.02, ..., 0.99, 1 to sum
for (float i = 0.01f; i <= 1.0f; i = i + 0.01f)
  sum += i;
```

```
The sum is 50.499985
```

```
// Initialize sum
double sum = 0;

// Add 0.01, 0.02, ..., 0.99, 1 to sum
for (double i = 0.01; i <= 1.0; i = i + 0.01)
  sum += i;
```

```
49.50000000000003
```

Minimizing Numeric Errors

```
double currentValue = 0.01;
for (int count = 0; count < 100; count++) {
   sum += currentValue;
   currentValue += 0.01;
}
```

```
sum is 50.50000000000003
```

```
double currentValue = 1.0;

for (int count = 0; count < 100; count++) {
   sum += currentValue;
   currentValue -= 0.01;
}
```

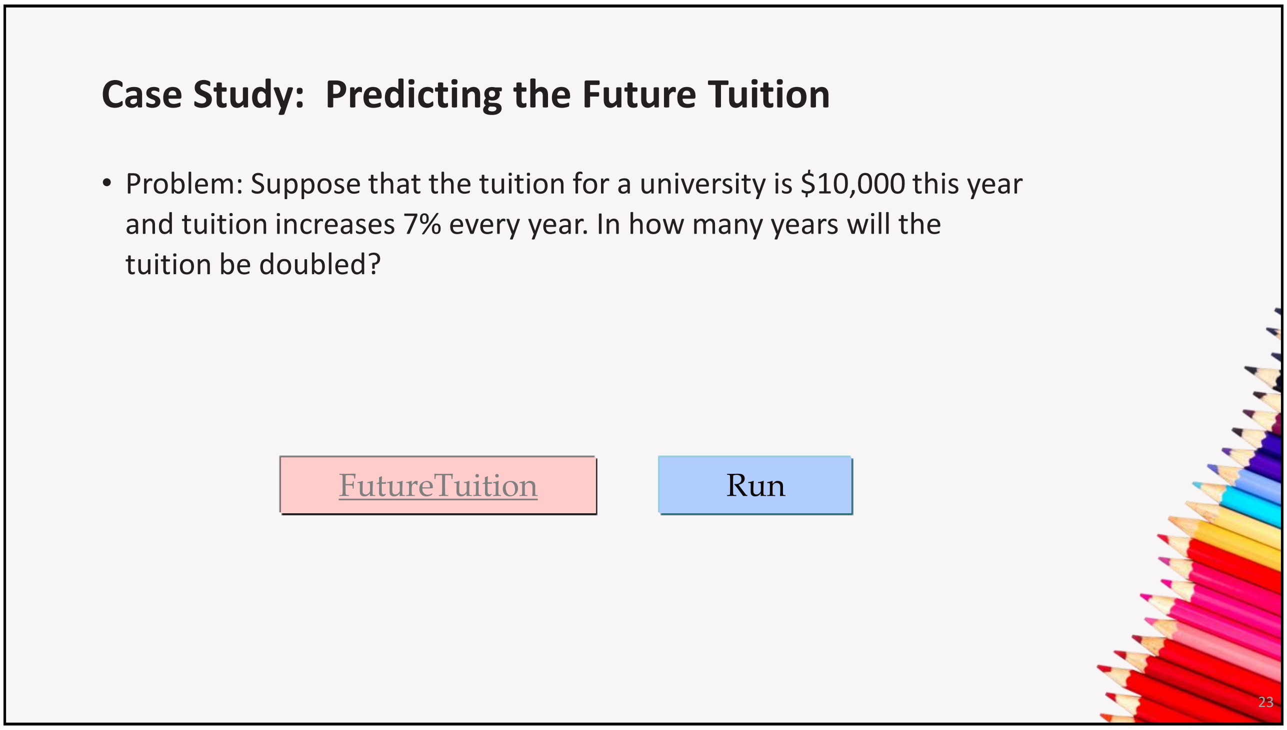
```
sum is 50.49999999999995
```

Case Study: Finding the Greatest Common Divisor

- Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.
- Solution: Let the two input integers be n1 and n2. You know number 1 is a common divisor, but it may not be the greatest commons divisor. So you can check whether k (for k = 2, 3, 4, and so on) is a common divisor for n1 and n2, until k is greater than n1 or n2.

GreatestCommonDivisor   Run

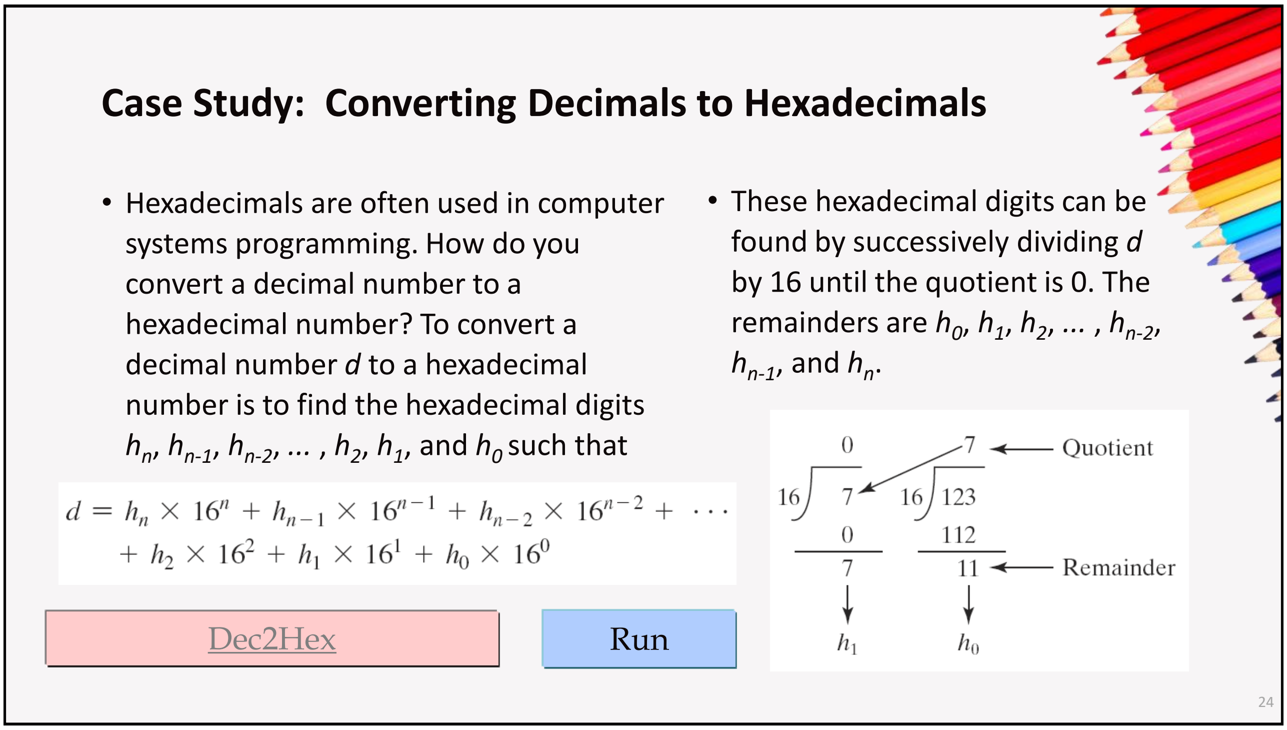
22



Case Study: Predicting the Future Tuition

- Problem: Suppose that the tuition for a university is $10,000 this year and tuition increases 7% every year. In how many years will the tuition be doubled?

FutureTuition Run



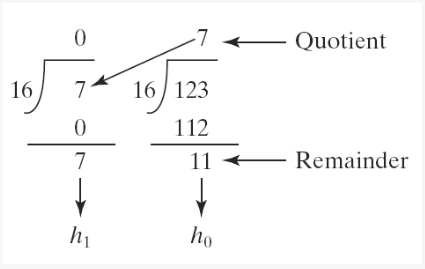
Case Study: Converting Decimals to Hexadecimals

- Hexadecimals are often used in computer systems programming. How do you convert a decimal number to a hexadecimal number? To convert a decimal number *d* to a hexadecimal number is to find the hexadecimal digits $h_n, h_{n-1}, h_{n-2}, \ldots, h_2, h_1$, and h_0 such that

$$d = h_n \times 16^n + h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + \cdots + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0$$

Dec2Hex Run

- These hexadecimal digits can be found by successively dividing *d* by 16 until the quotient is 0. The remainders are $h_0, h_1, h_2, \ldots, h_{n-2}, h_{n-1}$, and h_n.



5.12 Keywords break and continue

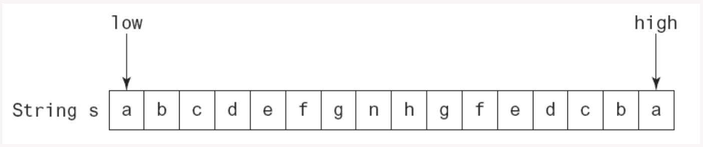
- The break and continue keywords provide additional controls in a loop.
- You can also use **break** in a loop to immediately terminate the loop.
- You can also use the **continue** keyword in a loop. When it is encountered, it ends the current iteration and program control goes to the end of the loop body.
- **break *[Identifier] ;***
  - *JLS8(14.15)*
- **continue *[Identifier] ;***
  - *JLS8(14.16)*

TestBreak Run

TestContinue Run

Case Study: Checking Palindrome

- A string is a palindrome if it reads the same forward and backward.
- The problem is to write a program that prompts the user to enter a string and reports whether the string is a palindrome.
- One solution is to check whether the first character in the string is the same as the last character. If so, check whether the second character is the same as the second-to-last character. This process continues until a mismatch is found or all the characters in the string are checked, except for the middle character if the string has an odd number of characters.

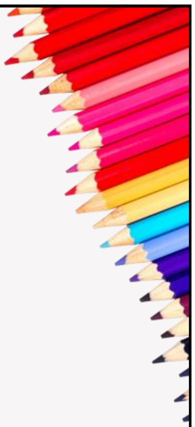


Palindrome

Run

26

Case Study: Displaying Prime Numbers

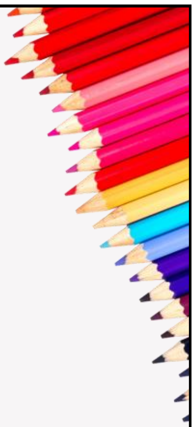


- Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself.
- Solution: The problem can be broken into the following tasks:
  - For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
  - Determine whether a given number is prime.
  - Count the prime numbers.
  - Print each prime number, and print 10 numbers per line.

PrimeNumber

Run

Chapter Summary



Chapter Summary

- There are three types of repetition statements: the **while loop,** the **do-while loop,** and the **for loop.**
- The part of the loop that contains the statements to be repeated is called the *loop body.*
- A one-time execution of a loop body is referred to as an *iteration of the loop.*
- In designing loops, you need to consider both the *loop control structure and the loop* body.
- Two keywords, **break** and **continue,** can be used in a loop.

Programming Exercises

1, 4, 7, 9, 12, 16, 17, 19,

20, 22, 24, 25, 27, 29,

30, 33, 34, 47, 48