4

Control Statements: Part 1

Objectives

- To understand basic problem-solving techniques.
- To be able to develop algorithms through the process of top-down, stepwise refinement.
- To be able to use the ifand if…elseselection statements to choose among alternative actions.
- To be able to use the whilerepetition statement to execute statements in a program repeatedly.
- To understand counter-controlled repetition and sentinel-controlled repetition.
- To be able to use the assignment, increment and decrement operators.

SELF-REVIEW EXERCISES

4.1 Fill in the blanks in each of the following statements:

a) All programs can be written in terms of three types of control structures: __________ , __________ and __________ .

ANS: sequence, selection, repetition

b) The __________ statement is used to execute one action when a condition is true and an- other action when that condition is false.
ANS: if…else

c) Repeating a set of instructions a specific number of times is called repetition.
ANS: counter-controlled (or definite)

d) When it is not known in advance how many times a set of statements will be repeated, a(n) __________ value can be used to terminate the repetition.
ANS: sentinel, signal, flag or dummy

4.2 Write four different Java statements that each add 1 to integer variable x.

ANS: x = x + 1;

```
x += 1;
++x;
x++;
```

4.3 Write Java statements to accomplish each of the following tasks:

a) Assign the sum of xand yto z, and increment xby 1 after the calculation. Use only one statement.

ANS: z = x++ + y;

b) Test whether variable countis greater than 10. If it is, print "Countis greaterthan 10".

ANS:
```
if ( count > 10 )
   System.out.println( "Count is greater than 10" );
```

c) Decrement the variable xby 1, then subtract it from the variable total. Use only one statement.

ANS: total -= --x;

d) Calculate the remainder after qis divided by divisor, and assign the result to q. Write this statement in two different ways.

ANS:
```
q %= divisor;
q = q % divisor;
```

4.4 Write a Java statement to accomplish each of the following tasks:

a) Declare variables sumand xto be of type int.

ANS: int sum, x;

b) Assign 1to variable x.

ANS: x = 1;

c) Assign 0to variable sum.
ANS: sum = 0;

d) Add variable xto variable sum, and assign the result to variable sum.
ANS: sum += x;or sum = sum + x;

e) Print "The sum is: ", followed by the value of variable sum.
ANS: System.out.println("The sum is: " + sum);

4.5 Combine the statements that you wrote in Exercise 4.4 into a Java application that calculates and prints the sum of the integers from 1 to 10. Use a while statement to loop through the calculation and increment statements. The loop should terminate when the value of xbecomes 11.
ANS:

```
// Calculate the sum of the integers from 1 to 10
public class Calculate {

    public static void main( String args[] )
    {
        int sum, x;

        x = 1;
        sum = 0;

        while ( x <= 10 ) {
            sum += x;
            ++x;
        }

        System.out.println( "The sum is: " + sum );

    } // end main

} // end class Calculate
```

4.6 Determine the value of each of the following variables after the calculation is performed. As- sume that when each statement begins executing, all variables are type intand have the value 5.
a) product *= x++;
ANS: product= 25, x= 6

b) quotient /= ++x;
ANS: quotient = 0, x = 6

4.7 Identify and correct the errors in each of the following sets of code:

a)
```
while ( c <= 5 ) {
    product *= c;
    ++c;
```

ANS: Error: The closing right brace of the whilestatement’s body is missing.

Correction: Add a closing right brace after the statement ++c;.

b)
```
if ( gender == 1 )
      System.out.println( "Woman"
      );
   else;
      System.out.println( "Man" );
```
ANS: Error: Semicolon after elseresults in a logic error. The second output statement will always be executed.
Correction: Remove the semicolon
after else.

4.8 What is wrong with the following whilestatement?

```
while ( z >= 0 )
   sum += z;
```

ANS: The value of the variable zis never changed in the while statement. Therefore, if the loop-continuation condition (z>=0)is true, an infinite loop is created. To prevent an infi- nite loop from occurring, zmust be decremented so that it eventually becomes less than 0.

EXERCISES

4.9 Identify and correct the errors in each of the following pieces of code. [*Note*: There may be more than one error in each piece of code.]

a)
```
if ( age >= 65 );
      System.out.println( "Age greater than or equal to 65" );
   else
      System.out.println( "Age is less than 65 )";
```
ANS: Semicolon at the end of the ifcondition should be removed. The closing double quote of the second System.out.printlnshould be inside of the closing parenthesis.

b)
```
int x = 1, total;
   while ( x <= 10 ) {
      total += x;
      ++x;
   }
```
ANS: The variable total should be
initialized to zero.

c) While (x <= 100
)

```
    total += x;
    ++x;
```

ANS: The W in While should be lowercase. The two statements should be enclosed in curly braces to properly group them into the body of the while; otherwise the loop will be an infinite loop.

d) while (y > 0) {

```
    System.out.println( y );
    ++y;
```

ANS: The ++operator should be changed to --. The closing curly brace for the while loop is missing.

4.10 What does the following program print?

```
public class Mystery {

    public static void main( String args[] )
    {
        int y, x = 1, total = 0;

        while ( x <= 10 ) {
            y = x * x;
            System.out.println( y );
            total += y;
            ++x;
        }

        System.out.println( "Total is " + total );

    } // end main

} // end class Mystery
```

ANS:

```
1
4
9
16
25
36
49
64
81
100
Total is 385
```

For Exercise 4.11 through Exercise 4.14, perform each of the following steps:

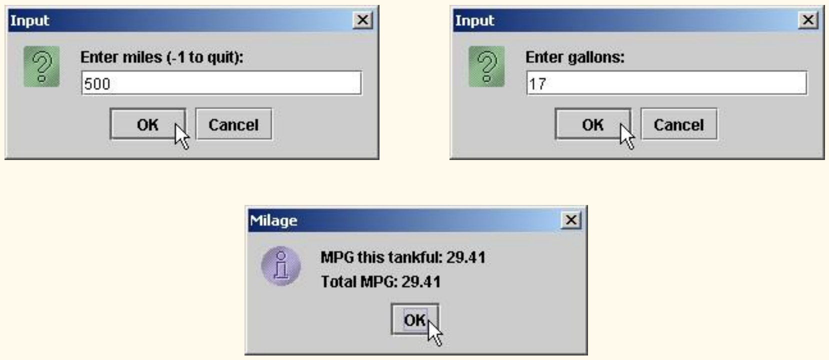
a) Read the problem statement.
b) Formulate the algorithm using pseudocode and top-down, stepwise refinement. c) Write a Java program.
d) Test, debug and execute the Java program. e) Process three complete sets of data.

4.11 Drivers are concerned with the mileage their automobiles get. One driver has kept track of several tankfuls of gasoline by recording miles driven and gallons used for each tankful. Develop a Java application that will input the miles driven and gallons used (both as integers) for each tankful. The program should calculate and display the miles per gallon obtained for each tankful and print the combined miles per gallon obtained for all tankfuls up to this point. All averaging calculations should produce floating-point results. Use input dialogs to obtain the data from the user.

ANS:

```
// Exercise 4.11 Solution: Gas.java
// Program calculates average mpg
import java.text.DecimalFormat;
import javax.swing.JOptionPane;

public class Gas {

   public static void main( String args[] )
   {
      int miles, gallons, totalMiles = 0, totalGallons = 0;
      float milesPerGallon, totalMilesPerGallon;
      String inputMiles, inputGallons, result = "";

      // read first number from user as a string
      inputMiles = JOptionPane.showInputDialog(
         "Enter miles (-1 to quit):" );

      // convert miles from type String to type int
      miles = Integer.parseInt( inputMiles );

      // exit if the input is -1 otherwise, proceed with the program
      while ( miles != -1 ) {
         // read second number from user as String
         inputGallons = JOptionPane.showInputDialog( "Enter gallons:"
);

         // convert gallons from type String to type int
         gallons = Integer.parseInt( inputGallons );

         totalMiles += miles;
         totalGallons += gallons;

         DecimalFormat twoDigits = new DecimalFormat( "0.00" );

         if ( gallons != 0 ) {
            milesPerGallon = (float) miles / gallons;
            result = "MPG this tankful: " +
               twoDigits.format( milesPerGallon ) + "\n";
         }

         totalMilesPerGallon = (float) totalMiles / totalGallons;
         result += "Total MPG: " +
            twoDigits.format( totalMilesPerGallon ) + "\n";

         JOptionPane.showMessageDialog( null, result, "Milage",
            JOptionPane.INFORMATION_MESSAGE );

         // input new value for miles and convert from String to int
         inputMiles = JOptionPane.showInputDialog(
            "Enter miles (-1 to quit):" );
         miles = Integer.parseInt( inputMiles );

      }  // end while loop
```

```
         System.exit( 0 );

      }  // end method main

   }  // end class Gas
```


Input

Enter miles (-1 to quit):

500

Input

Enter gallons:

17

Milage

MPG this tankful: 29.41

Total MPG: 29.41

4.12 Develop a Java application that will determine whether a department-store customer has ex- ceeded the credit limit on a charge account. For each customer, the following facts are available:

a) account number,

b) balance at the beginning of the month,

c) total of all items charged by the customer this month,

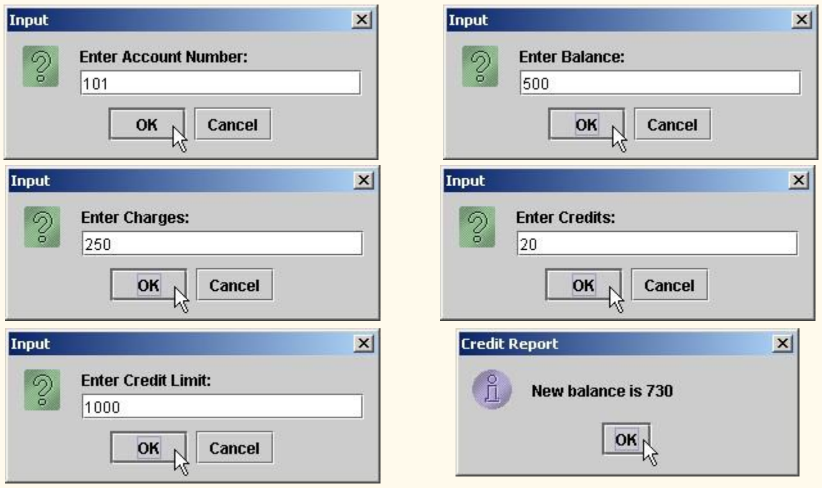
d) total of all credits applied to the customer's account this month and e) allowed credit limit.

The program should input each of these facts from input dialogs as integers, calculate the new bal- ance (= *beginning balance* + *charges* – *credits*), display the new balance and determine whether the new balance exceeds the customer's credit limit. For those customers whose credit limit is exceeded, the program should display the message "Credit limit exceeded."

ANS:

```
// Exercise 4.12 Solution: Credit.java
// Program monitors accounts.
import java.awt.*;
import javax.swing.JOptionPane;

public class Credit {

   public static void main( String args[] )
   {
      String inputString,           // user input
             resultsString,         // result String
             creditStatusString;    // credit status
      int account,                  // account number
          oldBalance,               // starting balance
          charges,                  // total charges
          credits,                  // total credits
```

```
                creditLimit,          // allowed credit limit
                newBalance;           // new balance

        inputString = JOptionPane.showInputDialog(
            "Enter Account Number:" );
        account = Integer.parseInt( inputString );

        inputString = JOptionPane.showInputDialog( "Enter Balance: " );
        oldBalance = Integer.parseInt( inputString );

        inputString = JOptionPane.showInputDialog( "Enter Charges: " );
        charges = Integer.parseInt( inputString );

        inputString = JOptionPane.showInputDialog( "Enter Credits: " );
        credits = Integer.parseInt( inputString );

        inputString = JOptionPane.showInputDialog( "Enter Credit Limit: " );
        creditLimit = Integer.parseInt( inputString );

        newBalance = oldBalance + charges - credits;

        resultsString = "New balance is " + newBalance;

        if ( newBalance > creditLimit )
            creditStatusString = "CREDIT LIMIT EXCEEDED";
        else
            creditStatusString = "Credit Report";

        JOptionPane.showMessageDialog( null, resultsString,
            creditStatusString, JOptionPane.INFORMATION_MESSAGE );

        System.exit( 0 );

    }  // end method main

}  // end class Credit
```



4.13 A large company pays its salespeople on a commission basis. The salespeople receive $200 per week, plus 9% of their gross sales for that week. For example, a salesperson who sells $5000 worth of merchandise in a week receives $200 plus 9% of $5000, or a total of $650. You have been supplied with a list of items sold by each salesperson. The values of these items are as follows:

Item	Value
1	239.99
2	129.75
3	99.95
4	350.89

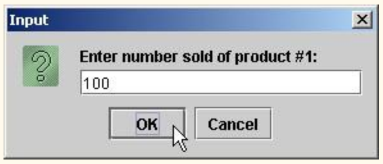
Develop a Java application that inputs one salesperson's items sold for last week and calculates and displays that salesperson's earnings. There is no limit to the number of items that can be sold by a salesperson.

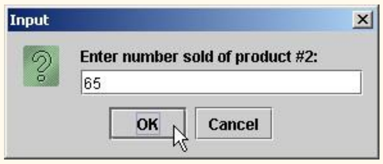
ANS:

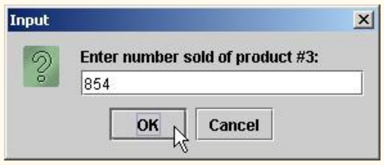
```java
// Exercise 4.13 Solution: Sales.java
// Program calculates commissions based on sales.
import java.text.DecimalFormat;
import javax.swing.JOptionPane;

public class Sales {

   public static void main( String args[] )
   {
      double gross = 0.0, earnings;
      int product = 0, numberSold;
      String input;

      while ( product < 4 ) {
         product++;

         // read number from user as String
         input = JOptionPane.showInputDialog(
            "Enter number sold of product #" + product + ":" );

         // convert numbers from type String to type int
         numberSold = Integer.parseInt( input );

         // determine gross of each individual product and add to total
         if ( product == 1 )
            gross += numberSold * 239.99;

         else if ( product == 2 )
            gross += numberSold * 129.75;

         else if ( product == 3 )
            gross += numberSold * 99.95;

         else if ( product == 4 )
            gross += numberSold * 350.89;

```

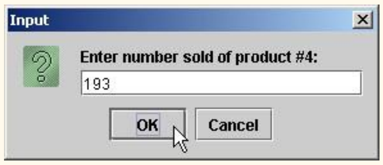
```
        }   // end while

```

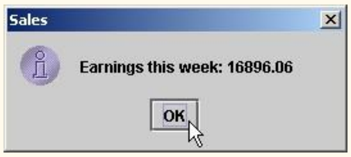
```
      DecimalFormat twoDigits = new DecimalFormat( "0.00" );

      earnings = 0.09 * gross + 200;
      String result = "Earnings this week: " +
         twoDigits.format( earnings );

      JOptionPane.showMessageDialog( null, result, "Sales",
         JOptionPane.INFORMATION_MESSAGE );

      System.exit( 0 );

   }  // end method main

}  // end class Sales
```


Input
Enter number sold of product #1:
100


Input
Enter number sold of product #2:
65


Input
Enter number sold of product #3:
854

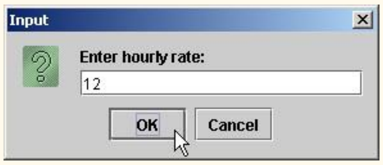

Input
Enter number sold of product #4:
193

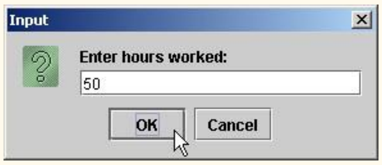

Sales
Earnings this week: 16896.06

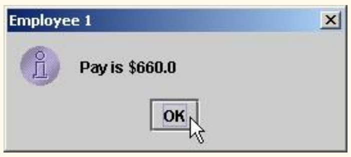
4.14 Develop a Java application that will determine the gross pay for each of three employees. The company pays “straight time” for the first 40 hours worked by each employee and pays “time and a half” for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the em- ployee’s gross pay. Use input dialogs to input the data.

ANS:

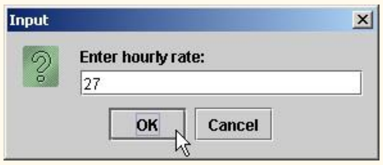
```
// Exercise 4.14 Solution: Wages.java
// Program calculates wages.
import java.awt.*;
import javax.swing.JOptionPane;

public class Wages {

   public static void main( String args[] )
   {
          String inputString,   // user input
                 resultsString;  // result String
       double pay;              // gross pay
       int hours,               // hours worked
           rate,                // hourly rate
           count = 1;           // number of

          // repeat calculation for 3 employees
          while ( count <= 3 ) {

             inputString = JOptionPane.showInputDialog(
                "Enter hourly rate: " );
             rate = Integer.parseInt( inputString );

             inputString = JOptionPane.showInputDialog(
                "Enter hours worked: " );
             hours = Integer.parseInt( inputString );

             // with overtime
             if ( hours > 40 )
                pay = ( 40.0 * rate ) + ( hours - 40 ) * ( rate * 1.5 );

             else // straight time
                pay = hours * rate;

             resultsString = "Pay is $" + pay;

             JOptionPane.showMessageDialog( null, resultsString,
                "Employee " + count,
JOptionPane.INFORMATION_MESSAGE );

             count++;
          }

          System.exit( 0 );

       }  // end method main

    }  // end class Wages
```

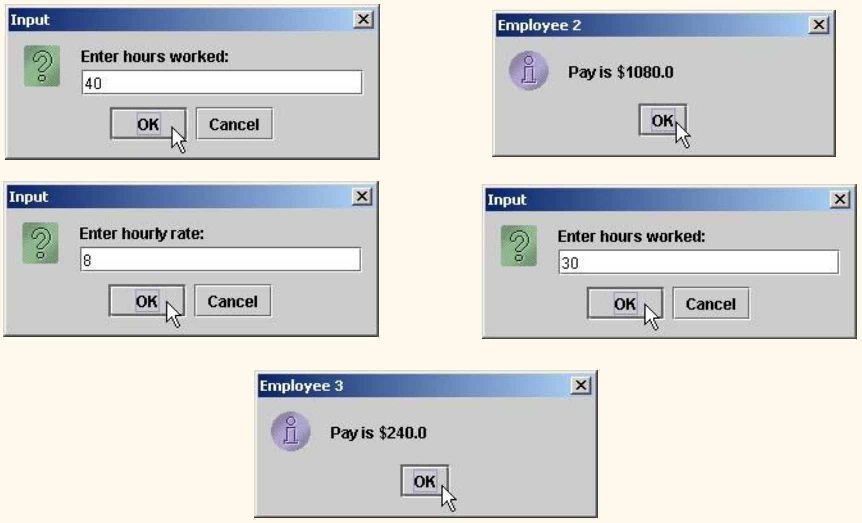






Pay is $660.0




Input
Enter hours worked:
40
OK Cancel

Employee 2
Pay is $1080.0
OK

Input
Enter hourly rate:
8
OK Cancel

Input
Enter hours worked:
30
OK Cancel

Employee 3
Pay is $240.0
OK

4.15 The process of finding the largest value (i.e., the maximum of a group of values) is used fre- quently in computer applications. For example, a program that determines the winner of a sales con- test would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a pseudocode program and then a Java application that inputs a series of
10 single-digit numbers as characters and determines and prints the largest of the numbers. Your pro- gram should use at least the following three variables:

a) counter: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed);

b) number: The current digit input to the program;

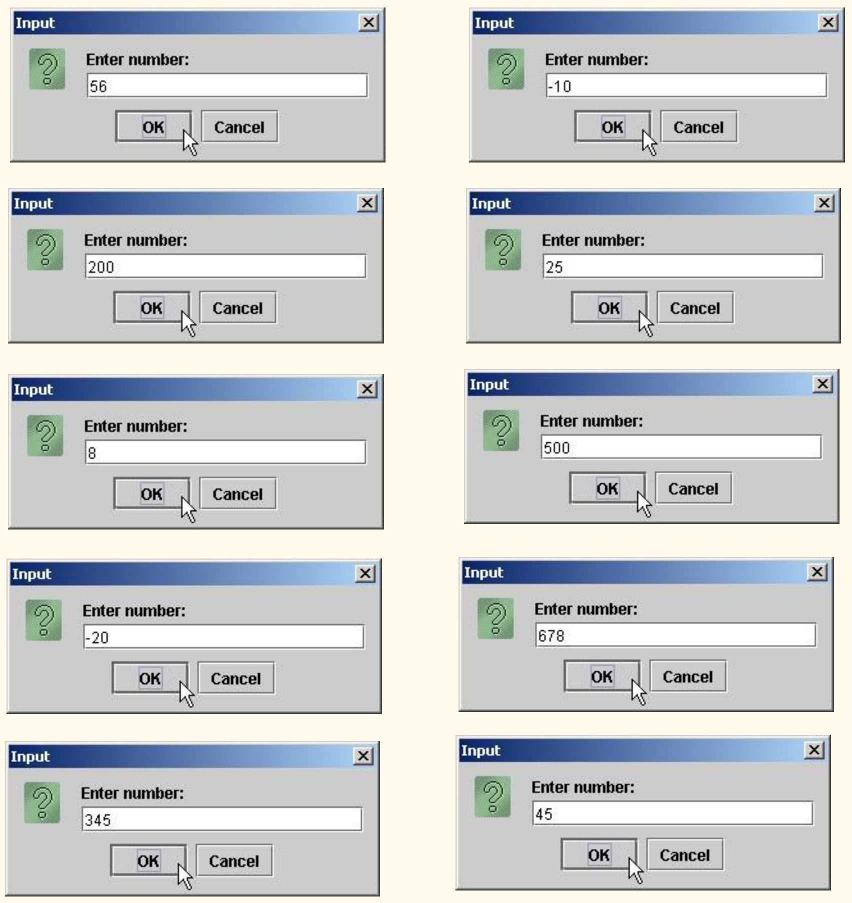
c) largest: The largest number found so far.

ANS: Pseudocode: Input the first number directly into the variable largest.

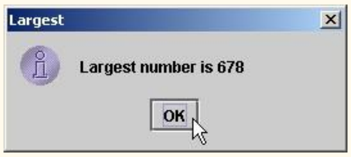
```
// Exercise 4.15 Solution: Largest.java
// Program determines and prints the largest of ten numbers.
import java.awt.*;
import javax.swing.*;

public class Largest {

   public static void main( String args[] )
   {
      int largest.     // largest number
```

```
          number,     // user input
          counter;    // number of inputted values

      // get first number and assign it to variable largest
      largest = Integer.parseInt(
            JOptionPane.showInputDialog( "Enter number: " ) );

      counter = 2;
```

```
        // get rest numbers and find the largest
        while ( counter <= 10 ) {
            number = Integer.parseInt(
                JOptionPane.showInputDialog( "Enter number: " ) );

            if ( number > largest )
                largest = number;

            counter++;
        }

        JOptionPane.showMessageDialog( null, "Largest number is " +
largest,
            "Largest", JOptionPane.INFORMATION_MESSAGE );

        System.exit( 0 );

    }  // end method main

}  // end class Largest
```



```
Enter number: 56
Enter number: -10
Enter number: 200
Enter number: 25
Enter number: 8
Enter number: 500
Enter number: -20
Enter number: 678
Enter number: 345
Enter number: 45
```



Largest number is 678

4.16 Write a Java application that uses looping to print the following table of values:

```
N        10*      100*     1000*N
          N       N
1        10       100      1000
2        20       200      2000
3        30       300      3000
4        40       400      4000
5        50       500      5000
```

ANS:

```java
// Exercise 4.16 Solution: Table.java
// Program prints a table of values using a while loop.

public class Table {

    public static void main( String args[] )
    {
        int n = 1;

        System.out.println( "N\t10*N\t100*N\t1000*N\n" );

        while ( n <= 5 ) {
            System.out.println( n + "\t" + ( 10 * n ) +
                "\t" + ( 100 * n ) + "\t" + ( 1000 * n ) );
            n++;
        }
    }

} // end class Table
```

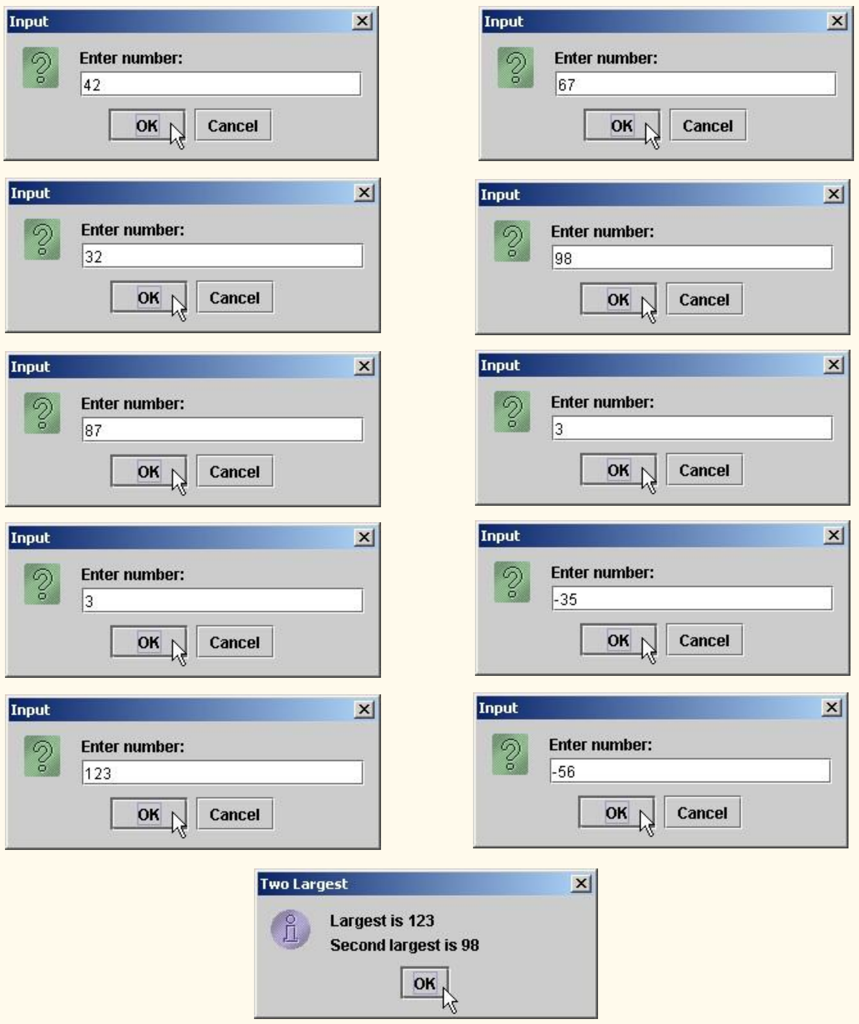
```
N      10*N    100*N   1000*N

1      10      100     1000
2      20      200     2000
3      30      300     3000
4      40      400     4000
5      50      500     5000
```

4.17 Using an approach similar to that for Exercise 4.15, find the *two* largest values of the 10 digits entered. [*Note*: You may input each number only once.]

ANS:

```
// Exercise 4.17 Solution: TwoLargest.java
// Program determines and prints the two largest of ten numbers.
import java.awt.*;
import javax.swing.*;

public class TwoLargest {

    public static void main( String args[] )
    {
        int largest,        // largest number
            nextLargest,    // second largest number
            number,         // user input
            counter;        // number of values inputted
        String resultsString;   // result String

        // get first number and assign it to variable largest
        largest = Integer.parseInt(
            JOptionPane.showInputDialog( "Enter number: " ) );

        // get second number and compare it with first number
        number = Integer.parseInt(
            JOptionPane.showInputDialog( "Enter number: " ) );

        if ( number > largest ) {
            nextLargest = largest;
            largest = number;
        }
        else
            nextLargest = number;

        counter = 3;

        // get rest numbers and find the largest and nextLargest
        while ( counter <= 10 ) {
            number = Integer.parseInt(
                JOptionPane.showInputDialog( "Enter number: " ) );

            if ( number > largest ) {
                nextLargest = largest;
                largest = number;
            }

            else if ( number > nextLargest )
                nextLargest = number;

            counter++;
        }

        resultsString = "Largest is " + largest +
            "\nSecond largest is " + nextLargest;

        JOptionPane.showMessageDialog( null, resultsString,
            "Two Largest", JOptionPane.INFORMATION_MESSAGE );
```

```

         System.exit( 0 );
      }  // end method main

   }  // end class TwoLargest
```



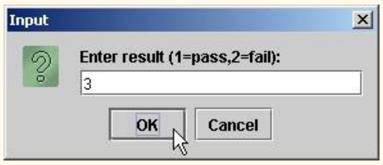
```
Input
Enter number:
42
OK  Cancel

Input
Enter number:
67
OK  Cancel

Input
Enter number:
32
OK  Cancel

Input
Enter number:
98
OK  Cancel

Input
Enter number:
87
OK  Cancel

Input
Enter number:
3
OK  Cancel

Input
Enter number:
3
OK  Cancel

Input
Enter number:
-35
OK  Cancel

Input
Enter number:
123
OK  Cancel

Input
Enter number:
-56
OK  Cancel

Two Largest
Largest is 123
Second largest is 98
OK
```

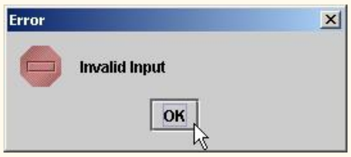
4.18 Modify the program in Fig. 4.11 to validate its inputs. For any input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

ANS:

```
// Exercise 4.18 Solution: Analysis.java
// Program performs analysis of examination results.
```

```
import javax.swing.JOptionPane;

public class Analysis
          {

   public static void main( String args[] )
   {
      // initializing variables in declarations
      int passes = 0, failures = 0, student = 1, result;
      String input, output;

      // process 10 students; counter-controlled loop
      while ( student <= 10 ) {
         input = JOptionPane.showInputDialog(
            "Enter result (1=pass,2=fail): " );

         result = Integer.parseInt( input );

         if ( result == 1 ) {      // if...else nested in while
            passes++;
            student++;
         }

         else if ( result == 2 ) {
            failures++;
            student++;
         }

         else
            JOptionPane.showMessageDialog( null, "Invalid Input",
               "Error", JOptionPane.ERROR_MESSAGE );
      }

      output = "Passed: " + passes + "\nFailed: " + failures;

      if ( passes > 8 )
         output += "\nRaise tuition ";

      JOptionPane.showMessageDialog( null, output,
         "Results", JOptionPane.INFORMATION_MESSAGE );

      System.exit( 0 );

   }  // end method main

}  // end class Analysis
```





4.19 What does the following program print?

```
public class Mystery2 {

   public static void main( String args[] )
   {
      int count = 1;

      while ( count <= 10 ) {
         System.out.println( count % 2 == 1 ? "****" : "++++++++" );

          ++count;
      }

    } // end main

 } // end class Mystery2
```

ANS:

```
****
++++++++
****
++++++++
****
++++++++
****
++++++++
****
++++++++
```

4.20 What does the following program print?

```
public class Mystery3 {

   public static void main( String args[] )
   {
      int row = 10, column;

      while ( row >= 1 ) {
         column = 1;

          while ( column <= 10 ) {
             System.out.print( row % 2 == 1 ? "<" : ">" );
             ++column;
          }

          --row;
          System.out.println();
      }
```

```
    } // end main

} // end class Mystery3
```

ANS:

```
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
```

4.21 *(Dangling-else Problem)* Determine the output for each of the given sets of code when xis 9and yis 11and when xis 11and yis 9. Note that the compiler ignores the indentation in a Java program. Also, the Java compiler always associates an else with the immediately preceding ifun- less told to do otherwise by the placement of braces ({}). On first glance, the programmer may not be sure which ifan elsematches; this situation is referred to as the “dangling-elseproblem.” We have eliminated the indentation from the following code to make the problem more challenging. [*Hint*: Apply indentation conventions you have learned.]

a) `if ( x < 10 ) if ( y > 10 )`

```
When:  x = 9, y = 11
*****
$$$$$

When:  x = 11, y = 9
```

b)
```
if ( x < 10 ) { if ( y > 10 )
    System.out.println( "*****" );
    }
    else {
    System.out.println( "#####" );
    System.out.println( "$$$$$" );
    }
```

ANS:

```
When:  x = 9, y = 11
*****

When:  x = 11, y = 9
#####
$$$$$
```

4.22 *(Another Dangling-else Problem)* Modify the given code to produce the output shown in each part of the problem. Use proper indentation techniques. Make no changes other than inserting braces and changing the indentation of the code. The compiler ignores indentation in a Java program. We have eliminated the indentation from the given code to make the problem more challenging. [*Note*: It is possible that no modification is necessary for some of the parts.]

```
if ( y == 8 ) if ( x == 5 )
System.out.println(
"@@@@@" ); else
System.out.println( "#####" );
System.out.println( "$$$$$" );
System.out.println(
"&&&&&" );
```

a) Assuming that x=5and y=8, the following output is produced:

```
@@@@@
$$$$$
&&&&&
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
      System.out.println( "@@@@@" );
   else
      System.out.println( "#####" );
System.out.println( "$$$$$" );
System.out.println( "&&&&&" );
```

b) Assuming that x=5and y=8, the following output is produced:

```
@@@@@
```

ANS:

```
if ( y == 8 )
   if ( x == 5 )
       System.out.println(
       "@@@@@" );
   else {
       System.out.println( "#####" );
       System.out.println( "$$$$$" );
       System.out.println(
       "&&&&&" );
   }
```

c) Assuming that x=5and y=8, the following output is produced:

```
@@@@@
&&&&&
```

ANS:

```
    if ( y == 8 )
       if ( x == 5 )
           System.out.println( "@@@@@" );
       else {
           System.out.println( "#####" );
           System.out.println( "$$$$$" );
       }
   System.out.println( "&&&&&" );
```

d) Assuming that x=5and y=7, the following output is produced. [*Note*: The last three output statements after the elseare all part of a block.]

```
#####
$$$$$
&&&&&
```

ANS:

```
   if ( y == 8 ) {
       if ( x == 5 )
           System.out.println( "@@@@@" );
   }
   else
       System.out.println( "#####" );

   System.out.println( "$$$$$" );
   System.out.println(
   "&&&&&" );
```

4.23 Write an applet that reads in the size of the side of a square and displays a hollow square of that size out of asterisks, by using the drawString method inside your applet's paintmethod. Use an input dialog to read the size from the user. Your program should work for squares of all side lengths between 1 and 20.

ANS:

```java
// Exercise 4.23 Solution: Hollow.java
// Program prints a hollow square.
import java.awt.Graphics;
import javax.swing.*;

public class Hollow extends JApplet {
        int stars;

        // initializes applet by obtaining value from user
         public void init()
         {
                String input;    // String entered by user

```