Chapter 4 Exercise Solutions

EX 4.1. What happens in the `MinOfThree` program if two or more of the values are equal? If exactly two of the values are equal, does it matter whether the equal values are lower or higher than the third?

If two or more values are equal, the program still prints the lowest value. Because only less than comparisons are made, the comparison of two equal values produces a false result. If two values are equal, and lower than the third value, then one of the two lower but equal values is printed. If all three values are equal, then this value is printed. Which "version" of the equal value is irrelevant.

If only two values are equal, it does not matter whether the third is lower or higher. The correct result is determined in either case. If the two equal values are lower than the third, then one of the two lower but equal values is printed. If the two equal values are higher than the third, then the third value is printed.

EX 4.2. What is wrong with the following code fragment? Rewrite it so that it produces correct output.

```
if (total == MAX)
   if (total < sum)
      System.out.println("total == MAX and < sum.");
else
   System.out.println("total is not equal to MAX");
```

Despite the indentation, the else clause is associated with the immediately preceding if rather than the first if. The program will produce the correct output if it is rewritten as:

```
if (total == MAX)
{
   if (total < sum)
      System.out.println("total == MAX and < sum.");
}
else
   System.out.println("total is not equal to MAX");
```

EX 4.3. What is wrong with the following code fragment? Will this code compile if it is part of an otherwise valid program? Explain.

```
if (length = MIN_LENGTH)
   System.out.println("The length is minimal.");
```

The assignment operator (=) is used erroneously in place of the equality operator (==). Hence, it will not compile in an otherwise valid program.

EX 4.4. What output is produced by the following code fragment?

```
int num = 87, max = 25;
if (num >= max*2)
   System.out.println("apple");
   System.out.println("orange");
System.out.println("pear");
```

The second println statement is improperly indented, so the output produced is:

```
apple
orange
pear
```

EX 4.5. What output is produced by the following code fragment?

```
int limit = 100, num1 = 15, num2 = 40;
if (limit <= limit)
{
   if (num1 == num2)
      System.out.println("lemon");
   System.out.println("lime");
}
System.out.println("grape");
```

The output is:

```
lime
grape
```

EX 4.6. Put the following list of strings in lexicographic order as if determined by the `compareTo` method of the `String` class. Consult the Unicode chart in Appendix C.

```
"fred"
"Ethel"
"?-?-?-?"
"{([])}"
"Lucy"
"ricky"
"book"
"******"
"12345"
"       "
"HEPHALUMP"
"bookkeeper"
"6789"
";+<?"
"^^^^^^^^^^"
"hephalump"
```

The strings in lexicographic order:

```
"       "
"******"
"12345"
"6789"
";+<?"
"?-?-?-?"
"Ethel"
"HEPHALUMP"
```

```
"Lucy"
"^^^^^^^^^^"
"book"
"bookkeeper"
"fred"
"hephalump"
"ricky"
"{([])}"
```

EX 4.7. What output is produced by the following code fragment?

```
int num = 1, max = 20;
while (num < max)
{
   System.out.println(num);
   num += 4;
}
```

The output produced is:

```
1
5
9
13
17
```

EX 4.8. What output is produced by the following code fragment?

```
int num = 1, max = 20;
while (num < max)
{
   if (num%2 == 0)
      System.out.println(num);
   num++;
}
```

The output produced is:

```
2
4
6
8
10
12
14
16
18
```

EX 4.9. What output is produced by the following code fragment?

```
for (int num = 0; num <= 200; num += 2)
   System.out.println(num);
```

The output produced is the even numbers from 0 to 200:

```
0
2
4
and so on until…
198
200
```

EX 4.10. What output is produced by the following code fragment?

```
for (int val = 200; val >= 0; val -= 1)
   if (val % 4 != 0)
      System.out.println(val);
```

The output produced is all values from 200 down to 0, except those that are evenly divisible by 4:

```
199
198
197
195
and so on until…
5
3
2
1
```

EX 4.11. Transform the following `while` loop into an equivalent `do` loop (make sure it produces the same output).

```
int num = 1;
while (num < 20)
{
   num++;
   System.out.println(num);
}
```

This code can be written using a do loop as follows:

```
int num = 1;
do
{
   num++;
   System.out.println(num);
}
while (num < 20);
```

EX 4.12. Transform the `while` loop from exercise 4.11 into an equivalent `for` loop (make sure it produces the same output).

```
for (int num = 2; num <=20; num ++)
   System.out.println(num);
```

EX 4.13. What is wrong with the following code fragment? What are three distinct ways it could be changed to remove the flaw?

```
count = 50;
while (count >= 0)
{
   System.out.println(count);
   count = count + 1;
}
```

The loop is infinite because count initially is greater than zero, and continues to increase in value. The flaw can be removed by (1) decrementing rather than incrementing count, (2) initializing count to 0 and using, as the condition of the while loop, `count <= 50`, and (3) picking an upper limit and using, as the condition of the while loop, `count <= upperLimit`.

EX 4.14. Write a `while` loop that verifies that the user enters a positive integer value.

```
Scanner scan = new Scanner(System.in);
System.out.print("Enter a positive integer: ");
number = scan.nextInt();
while (number <= 0)
{
   System.out.print("That number was not positive.");
   System.out.print("Enter a positive integer: ");
   number = scan.nextInt();
}
```

EX 4.15. Write a `do` loop that verifies that the user enters an even integer value.

```
Scanner scan = new Scanner(System.in);
do
{
   System.out.print("Enter an even integer: ");
   number = scan.nextInt();
}
while {number%2 != 0);
```

EX 4.16. Write a code fragment that reads and prints integer values entered by a user until a particular sentinel value (stored in `SENTINEL`) is entered. Do not print the sentinel value.

```
Scanner scan = new Scanner(System.in);
System.out.print("Enter some integers (" + SENTINEL +
   " to quit): ");
number = scan.nextInt();
while (number != SENTINEL)
{
   System.out.println(number);
   number = scan.nextInt();
}
```

EX 4.17. Write a `for` loop to print the odd numbers from 1 to 99 (inclusive).

```
for (int value = 1; value <= 99; value +=2)
   System.out.println(value);
```

EX 4.18. Write a `for` loop to print the multiples of 3 from 300 down to 3.

```
for (int value = 300; value >= 3, value -= 3)
   System.out.println(value);
```

EX 4.19. Write a code fragment that reads 10 integer values from the user and prints the highest value entered.

```
Scanner scan = new Scanner(System.in);
int max, number;
System.out.print("Enter an integer: ");
max = scan.nextInt();
for (int count = 2; count <= 10; count++)
{
   System.out.print("Enter another integer: ");
   number = scan.nextInt();
   if (number > max)
      max = number;
}
System.out.println("The highest value is :" + max);
```

EX 4.20. Write a code fragment that computes the sum of the integers from 20 to 70, inclusive, then prints the results.

```
int sum = 0;
for (int count = 20; count <= 70; count++)
   sum += count;
System.out.println("The sum is " + sum);
```

EX 4.21. Write a code fragment that determines and prints the number of times the character `'z'` appears in a `String` object called `name`.

```
int count = 0;
for (int index = 0; index < name.length(); index++)
   if (name.charAt(index) == 'z')
      count++;
System.out.println("The character \'z\' appears "
   + count + " time(s)");
```

EX 4.22. Write a code fragment that prints the characters stored in a `String` object called `str` backward.

```
for (int index = str.length()-1; index >= 0; index--)
   System.out.print(str.charAt(index));
System.out.println();
```

EX 4.23. Write a code fragment that prints every other character in a `String` object called `word` starting with the first character.

```
for (int index = 0; index < word.length(); index +=2)
   System.out.println(word.charAt(index));
```