Chapter 4
BJP 3rd Ed

Name: Lemsbow

per: _____

Chapter Summary

An `if` statement lets you write code that will execute only if a certain condition is met. An `if/else` statement lets you execute one piece of code if a condition is met, and another if the condition is not met. Conditions are Boolean expressions and can be written using relational operators such as <, >= , and != . You can test multiple conditions using the `&&` and `||` operators.

You can nest `if/else` statements to test a series of conditions and execute the appropriate block of code on the basis of whichever condition is true.

The == operator that tests primitive data for equality doesn't behave the way we would expect with objects, so we test objects for equality by calling their `equals` method instead.

Common code that appears in every branch of an `if/else` statement should be factored out so that it is not replicated multiple times in the code.

Cumulative algorithms compute values incrementally. A cumulative sum loop declares a sum variable and incrementally adds to that variable's value inside the loop.

Since the `double` type does not store all values exactly, small roundoff errors can occur when the computer performs calculations on real numbers. Avoid these errors by providing a small amount of tolerance in your code for values near the values that you expect.

The `char` type represents individual characters of text. Each letter of a `String` is stored internally as a `char` value, and you can use the `String`'s `charAt` method to access these characters with an index.

The `System.out.printf` method prints formatted text. You can specify complex format strings to control the width, alignment, and precision by which values are printed.

You can "throw" (generate) exceptions in your own code. This technique can be useful if your code ever reaches an unrecoverable error condition, such as the passing of an invalid argument value to a method.

Self-Check Problems

Section 4.1: `if/else` Statements

1. Translate each of the following English statements into logical tests that could be used in an `if/else` statement. Write the appropriate `if` statement with your logical test. Assume that three `int` variables, `x`, `y`, and `z`, have been declared.

a. z is odd. `if (z % 2 != 0)`

b. z is not greater than y's square root. `Math.sqrt(y) >= z`

c. y is positive. `y > 0`

d. Either x or y is even, and the other is odd. `Math.max(x%2, y%2) != 0 && Math.min(x%2, y%2) == 0`

e. y is a multiple of z. `(double) y) % z == 0`

f. z is not zero. `z != 0`

g. y is greater in magnitude than z. `Math.abs(y) > Math.abs(z)`

h. x and z are of opposite signs. `Math.min(x, z) < 0 && Math.max(x, z) > 0`

i. y is a nonnegative one-digit number. `y >= 0 && y < 10`

j. z is nonnegative.
k. x is even.
l. x is closer in value to y than z is.

2. Given the variable declarations

```
int x = 4;
int y = -3;
int z = 4;
```

what are the results of the following relational expressions?

a. `x == 4`
b. `x == y`
c. `x == z`
d. `y == z`
e. `x + y > 0`
f. `x - z != 0`
g. `y * y <= z`
h. `y / y == 1`
i. `x * (y + 2) > y - (y + z) * 2`

3. Which of the following `if` statement headers uses the correct syntax?

a. `if x = 10 then {`
b. `if [x == 10] {`
c. `if (x => y) {`
d. `if (x equals 42) {`
e. `if (x == y) {`

4. The following program contains 7 mistakes! What are they?

```
1  public class Oops4 {
2      public static void main(String[] args) {
3          int a = 7, b = 42;
4          minimum(a, b);
5          if (smaller = a) {
6              System.out.println("a is the smallest!");
7          }
8      }
9
10     public static void minimum(int a, int b) {
11         if (a < b) {
12             int smaller = a;
13         } else (a => b) {
14             int smaller = b;
15         }
16         return int smaller;
17     }
18 }
```

5. Consider the following method:

```
public static void ifElseMystery1(int x, int y) {
    int z = 4;
    if (z <= x) {
        z = x + 1;
    } else {
        z = z + 9;
    }
    if (z <= y) {
        y++;
    }
    System.out.println(z + " " + y);
}
```

What output is produced for each of the following calls?

a. `ifElseMystery1(3, 20);` // 13 21
b. `ifElseMystery1(4, 5);` // 5 6
c. `ifElseMystery1(5, 5);` // 6 5
d. `ifElseMystery1(6, 10);` // 7 11

6. Consider the following method:

```
public static void ifElseMystery2(int a, int b) {
    if (a * 2 < b) {
        a = a * 3;
    } else if (a > b) {
        b = b + 3;
    }
    if (b < a) {
        b++;
    } else {
        a--;
    }
    System.out.println(a + " " + b);
}
```

What output is produced for each of the following calls?

a. `ifElseMystery2(10, 2);` // 10 6
b. `ifElseMystery2(3, 8);` // 8 8
c. `ifElseMystery2(4, 4);` // 3 4
d. `ifElseMystery2(10, 30);` // 29 30

7. Write Java code to read an integer from the user, then print even if that number is an even number or odd otherwise. You may assume that the user types a valid integer.

```
System.out.println(new Scanner(System.in).nextInt() % 2 == 0 ? "even" : "odd");
```

8. The following code contains a logic error:

```
Scanner console = new Scanner(System.in);
System.out.print("Type a number: ");
int number = console.nextInt();
if (number % 2 == 0) {
    if (number % 3 == 0) {
        System.out.println("Divisible by 6.");
    } else {
        System.out.println("Odd.");
    }
}
```

Examine the code and describe a case in which the code would print something that is untrue about the number that was entered. Explain why. Then correct the logic error in the code.

9. Describe a problem with the following code:

```
Scanner console = new Scanner(System.in);
System.out.print("What is your favorite color?");
String name = console.next();
if (name == "blue") {
    System.out.println("Mine, too!");
}
```

10. Factor out redundant code from the following example by moving it out of the `if/else` statement, preserving the same output.

```
if (x < 30) {
    a = 2;
    x++;
    System.out.println("Java is awesome! " + x);
} else {
    a = 2;
    System.out.println("Java is awesome! " + x);
}
```

11. The following code is poorly structured:

```
int sum = 1000;
Scanner console = new Scanner(System.in);
System.out.print("Is your money multiplied 1 or 2 times? ");
int times = console.nextInt();
if (times == 1) {
    System.out.print("And how much are you contributing? ");
    int donation = console.nextInt();
    sum = sum + donation;
    count1++;
    total = total + donation;
}
```

```
if (times == 2) {
    System.out.print("And how much are you contributing? ");
    int donation = console.nextInt();
    sum = sum + 2 * donation;
    count2++;
    total = total + donation;
}
```

Rewrite it so that it has a better structure and avoids redundancy. To simplify things, you may assume that the user always types 1 or 2. (How would the code need to be modified to handle any number that the user might type?)

12. The following code is poorly structured.

```
Scanner console = new Scanner(System.in);
System.out.print("How much will John be spending? ");
double amount = console.nextDouble();
System.out.println();
int numBills1 = (int) (amount / 20.0);
if (numBills1 * 20.0 < amount) {
    numBills1++;
}
System.out.print("How much will Jane be spending? ");
amount = console.nextDouble();
System.out.println();
int numBills2 = (int) (amount / 20.0);
if (numBills2 * 20.0 < amount) {
    numBills2++;
}
System.out.println("John needs " + numBills1 + " bills");
System.out.println("Jane needs " + numBills2 + " bills");
```

Rewrite it so that it has a better structure and avoids redundancy. You may wish to introduce a method to help capture redundant code.

13. Write a piece of code that reads a shorthand text description of a color and prints the longer equivalent. Acceptable color names are `B` for `Blue`, `G` for `Green`, and `R` for `Red`. If the user types something other than `B`, `G`, or `R`, the program should print an error message. Make your program case-insensitive so that the user can type an uppercase or lowercase letter. Here are some example executions:

```
What color do you want? B
You have chosen Blue.

What color do you want? g
You have chosen Green.

What color do you want? Bork
Unknown color: Bork
```

14. Write a piece of code that reads a shorthand text description of a playing card and prints the longhand equivalent. The shorthand description is the card's rank (`2` through `10`, `J`, `Q`, `K`, or `A`) followed by its suit (`C`, `D`, `H`, or `S`). You should expand the shorthand into the form "<Rank> of <Suit>". You may assume that the user types valid input. Here are two sample executions:

```
Enter a card: 9 S
Nine of Spades

Enter a card: K C
King of Clubs
```

Section 4.2: Cumulative Algorithms

15. What is wrong with the following code, which attempts to add all numbers from 1 to a given maximum? Describe how to fix the code.

```
public static int sumTo(int n) {
    for (int i = 1; i <= n; i++) {
        int sum = 0;
        sum += i;
    }
    return sum;
}
```

16. What is wrong with the following code, which attempts to return the number of factors of a given integer *n*? Describe how to fix the code.

```
public static int countFactors(int n) {
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {  // factor
            return i;
        }
    }
}
```

17. Write code to produce a cumulative product by multiplying together many numbers that are read from the console.

18. The following expression should equal `6.8`, but in Java it does not. Why not?

```
0.2 + 1.2 + 2.2 + 3.2
```

19. The following code was intended to print a message, but it actually produces no output. Describe how to fix the code to print the expected message.

```
double gpa = 3.2;
if (gpa * 3 == 9.6) {
    System.out.println("You earned enough credits.");
}
```

Self-Check Problems

Section 4.3: Text Processing

20. What output is produced by the following program?

```
1 public class CharMystery {
2     public static void printRange(char startLetter, char endLetter) {
3         for (char letter = startLetter; letter <= endLetter; letter++) {
4             System.out.print(letter);
5         }
6         System.out.println();
7     }
8
9     public static void main(String[] args) {
10        printRange('e', 'g'); // efg
11        printRange('n', 's'); // nopqrs
12        printRange('z', 'a'); //
13        printRange('q', 'r'); // qr
14    }
15 }
```

21. Write an `if` statement that tests to see whether a `String` begins with a capital letter.

22. What is wrong with the following code, which attempts to count the number occurrences of the letter `'e'` in a `String`, case-insensitively?

```
int count = 0;
for (int i = 0; i < s.length(); i++) {
    if (s.charAt(i).toLowerCase() == 'e') {
        count++;
    }
}
```

23. Consider a `String` stored in a variable called `name` that stores a person's first and last name (e.g., "Marla Singer"). Write the expression that would produce the last name followed by the first initial (e.g., "Singer, M.").

24. Write code to examine a `String` and determine how many of its letters come from the second half of the alphabet (that is, have values of `'n'` or subsequent letters). Compare case-insensitively, such that values of `'N'` through `'Z'` also count. Assume that every character in the `String` is a letter.

Section 4.4: Methods with Conditional Execution

25. Consider a method `printTriangleType` that accepts three integer arguments representing the lengths of the sides of a triangle and prints the type of triangle that these sides form. The three types are equilateral, isosceles, and scalene. An equilateral triangle has three sides of the same length, an isosceles triangle has two sides that are the same length, and a scalene triangle has three sides of different lengths.

However, certain integer values (or combinations of values) would be illegal and could not represent the sides of an actual triangle. What are these values? How would you describe the precondition(s) of the `printTriangleType` method?

26. Consider a method `getGrade` that accepts an integer representing a student's grade percentage in a course and returns that student's numerical course grade. The grade can be between `0.0` (failing) and `4.0` (perfect). What are the preconditions of such a method?

27. The following method attempts to return the median (middle) of three integer values, but it contains logic errors. In what cases does the method return an incorrect result? How can the code be fixed?

```
public static int medianOf3(int n1, int n2, int n3) {
    if (n1 < n2) {
        if (n2 < n3) {
            return n2;
        } else {
            return n3;
        }
    } else {
        if (n1 < n3) {
            return n1;
        } else {
            return n3;
        }
    }
}
```

28. One of the exercises in Chapter 3 asked you to write a method that would find the roots of a quadratic equation of the form $ax^2 + bx + c = 0$. The quadratic method was passed `a`, `b`, and `c` and then applied the following quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Under what conditions would this formula fail? Modify the quadratic method so that it will reject invalid values of `a`, `b`, or `c` by throwing an exception. (If you did not complete the exercise in the previous chapter, just write the method's header and the exception-throwing code.)

29. Consider the following Java method, which is written incorrectly:

```
// This method should return how many of its three
// arguments are odd numbers.
public static void printNumOdd(int n1, int n2, int n3) {
    int count = 0;
    if (n1 % 2 != 0) {
        count++;
    } else if (n2 % 2 != 0) {
        count++;
```

```
    } else if (n3 % 2 != 0) {
        count++;
    }
    System.out.println(count + " of the 3 numbers are odd.");
}
```

Under what cases will the method print the correct answer, and when will it print an incorrect answer? What should be changed to fix the code? Can you think of a way to write the code correctly without any `if/else` statements?

It is correct if none or 1, but can return wrong answers. no

Exercises

1. Write a method called `fractionSum` that accepts an integer parameter n and returns as a `double` the sum of the first n terms of the sequence

$$\sum_{i=1}^{n} \frac{1}{i}$$

In other words, the method should generate the following sequence:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \ldots$$

You may assume that the parameter n is nonnegative.

2. Write a method called `repl` that accepts a `String` and a number of repetitions as parameters and returns the `String` concatenated that many times. For example, the call `repl("hello", 3)` should return `"hellohellohello"`. If the number of repetitions is zero or less, the method should return an empty string.

3. Write a method called `season` that takes as parameters two integers representing a month and day and returns a `String` indicating the season for that month and day. Assume that the month is specified as an integer between 1 and 12 (1 for January, 2 for February, and so on) and that the day of the month is a number between 1 and 31. If the date falls between 12/16 and 3/15, the method should return `"winter"`. If the date falls between 3/16 and 6/15, the method should return `"spring"`. If the date falls between 6/16 and 9/15, the method should return `"summer"`. And if the date falls between 9/16 and 12/15, the method should return `"fall"`.

4. Write a method called `daysInMonth` that takes a month (an integer between 1 and 12) as a parameter and returns the number of days in that month in this year. For example, the call `daysInMonth(9)` would return `30` because September has 30 days. Assume that the code is not being run during a leap year (that February always has 28 days). The following table lists the number of days in each month:

Month	1 Jan	2 Feb	3 Mar	4 Apr	5 May	6 Jun	7 Jul	8 Aug	9 Sep	10 Oct	11 Nov	12 Dec
Days	31	28	31	30	31	30	31	31	30	31	30	31

5. Write a method called `pow` that accepts a base and an exponent as parameters and returns the base raised to the given power. For example, the call `pow(3, 4)` should return `3 * 3 * 3 * 3`, or `81`. Assume that the base and exponent are nonnegative.