Chapter 5

Conditionals and Logic

5.1 Exercises

Exercise 5.1 Here is a table of wavelength ranges associated with the colors of visible light.

Wavelength λ in nm	**Color**
$0 < \lambda < 380$	Ultraviolet
$380 \leq \lambda < 450$	Violet
$450 \leq \lambda < 485$	Blue
$485 \leq \lambda < 500$	Cyan
$500 \leq \lambda < 565$	Green
$565 \leq \lambda < 590$	Yellow
$590 \leq \lambda < 625$	Orange
$625 \leq \lambda < 700$	Red
$700 \leq \lambda \leq 1000$	Infrared

Write a program that prompts the user for the wavelength of light and prints the corresponding color. If the value is less than or equal to zero or greater

than 1000, print an error that tells what the correct range should be. Using an if-else chain is probably the best way to approach this exercise.

Exercise 5.2 Write a program that prompts the user for what percent of their maximum heart rate they achieved during an exercise session. The program will then describe the level of exercise they had:

% p of maximum heart rate	**Description**
$0 < p < 50$	Extremely light
$50 \leq p < 60$	Very light
$60 \leq p < 70$	Light
$70 \leq p < 80$	Moderate
$80 \leq p < 90$	Hard
$90 \leq p \leq 100$	Maximum

If the percentage entered is less than or equal to zero or more than 100, print an appropriate error message. Here is what the program output might look like for several runs of the program:

```
What percent of your max heart rate did you reach? 72.8
Your exercise session was: Moderate

What percent of your max heart rate did you reach? 92
Your exercise session was: Maximum

What percent of your max heart rate did you reach? 102.6
Percent must be from 0 to 100.
```

Exercise 5.3 Here is a table that gives ticket prices for a movie theater. The price depends on the day of week, time of day, and age of the customer.

Day	Time	Age	Price
Mon–Fri	Before 2 p.m.	All	$5.00
	After 2 p.m.	Teen/Adult	$6.00
		Child/Senior	$4.50
Sat–Sun	Before 1 p.m.	Teen/Adult	$6.50
		Child/Senior	$5.25
	After 1 p.m.	Teen/Adult	$8.00
		Child/Senior	$5.50

Write a program that asks the user for:

- The day of the week, where 1 stands for Monday and 7 stands for Sunday
- The time of day as an hour from 0 (midnight) to 23 (11:00 p.m.)
- The customer's age. If a customer's age is from 13 to 64 (inclusive), they are a "Teen/Adult". Everyone else is a "Child/Senior".

The program then prints the ticket price for the customer. *Hint*: To make your code easier to read, write a method named `isTeenAdult` that takes the customer's age as its parameter and returns a `boolean` `true` if the age is from 13 to 64 (inclusive), `false` otherwise.

Do not allow the user to enter a day outside the range 1-7, an hour outside the range 0-23, or an age less than zero.

Using nested if-else statements is probably your best approach to this exercise.

Exercise 5.4 Write a program that prompts the user for a number in the range 1-12 and returns the number of days of that month. If the month is 2 (February) print that the month has 28 days in a normal year and 29 in a leap year. Be sure to validate the input. Here is what the output from running the program several times might look like:

```
Enter a month number (1-12): 3
Month number 3 has 31 days.

Enter a month number (1-12): 14
Invalid month; must be from 1 to 12.

Enter a month number (1-12): 2
Month number 2 has 28 days in a normal year
and 29 days in a leap year.
```

Hint: Use a `switch` statement to set a variable with the number of days in the month. Then, use an `if` statement to determine whether the month is February or not and print the appropriate output. Do *not* put the `System.out.println` inside the `switch` statement—that would lead to unnecessarily duplicated code.

Exercise 5.5 One phone messaging plan has a base cost of $9.50 per month that includes 60 messages. You pay 4.5 cents for each message over 60 messages sent. The other messaging plan has a base cost of $8.75 per month that includes 50 messages. You pay 5.2 cents for each message over 50 messages sent.

Write a program that will:

1. Ask the user for number of messages sent (integer).
2. Calculate and display the cost for the first plan.
3. Calculate and display the cost for the second plan.
4. Tell which plan is cheaper. The amounts for the two plans will never be exactly equal when the units are integers, although they may appear to be when rounded to two decimal places.

Your program will use a method named `calculateCost` that returns the cost of a plan based on four parameters:

- The number of messages sent
- The base cost per month
- The number of messages included in the base cost (the "base limit")

- The cost per message when you go over the base limit

Here are the results of running the program several times. Your output does not have to look exactly like this, but it must reflect the same information. All monetary amounts must be displayed with two digits after the decimal point.

```
Enter number of messages: 40
Cost for plan one: $9.50
Cost for plan two: $8.75
Plan two is cheaper.

Enter number of messages: 80
Cost for plan one: $10.40
Cost for plan two: $10.31
Plan two is cheaper.

Enter number of messages: 150
Cost for plan one: $13.55
Cost for plan two: $13.95
Plan one is cheaper.
```

Exercise 5.6 This code fragment contains unnecessary condition tests:

```
double bonus = 3.50;
if (hours < 10) {
    bonus = 0.50;
} else if (hours >= 10 && hours < 20) {
    bonus = 1.25;
} else if (hours >= 20 && hours < 40) {
    bonus = 2.25;
}
```

It can be rewritten like this:

```
double bonus = 3.50;
if (hours < 10) {
    bonus = 0.50;
} else if (hours < 20) {
    bonus = 1.25;
} else if (hours < 40) {
    bonus = 2.25;
}
```

Why is it possible to eliminate the first condition before the `&&`?

Now consider this code:

```
double bonus = 3.50;
if (hours < 10) {
    bonus = 0.50;
}
if (hours >= 10 && hours < 20) {
    bonus = 1.25;
}
if (hours >= 20 && hours < 40) {
    bonus = 2.25;
}
```

Is it possible to eliminate the condition before the `&&` in this code? If so, why? If not, why not?

Exercise 5.7 Here is a program that asks a user for their age in years, tells the user their approximate age in days, and gives them a discount based on how old they are:

```
import java.util.Scanner;

public class DuplicatedCode {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter your age in years: ");
        if (in.hasNextInt()) {
            int years = in.nextInt();
            int days = years * 365;
            double discount = 0.0;
            if (years > 0 && years < 18) {
                System.out.printf("That is about %d days.\n",
                    days);
                discount = years * 0.10;
                System.out.printf("Your discount is $%.2f\n",
                    discount);
            } else if (years < 21) {
                System.out.printf("That is about %d days.\n",
                    days);
                discount = years * 0.15;
                System.out.printf("Your discount is $%.2f\n",
                    discount);
            } else if (years < 65) {
                System.out.printf("That is about %d days.\n",
                    days);
                discount = years * 0.25;
                System.out.printf("Your discount is $%.2f\n",
                    discount);
            } else {
                System.out.printf("That is about %d days.\n",
                    days);
                discount = years * 0.40;
                System.out.printf("Your discount is $%.2f\n",
                    discount);
            }
        } else {
            System.out.println("Age must be numeric.");
        }
    }
}
```

The code works, but there is a lot of code that is repeated unnecessarily. Rewrite the code so that the code that prints the days and the code that prints the discount each appear only once in the program. Can you explain a general rule that tells when and how this sort of duplication can be eliminated?

Exercise 5.8 In the dice game called "'craps," the first phase of the game begins with a player rolling two dice. If the total is 7 or 11, the player wins. If the total is 2, 3, or 12, the player loses. If the total is another number (4, 5, 6, 8, 9, or 10), that number is called the player's "point," and the game enters its second phase, which we will not discuss here.

Write a program that simulates the first phase of the game by randomly rolling two dice and displaying the result.

Here is what the output from running the programs several times might look like:

```
You rolled 3 and 5 - your point is 8.

You rolled 1 and 1 - you lose, sorry.

You rolled 3 and 4 - you win!
```

One difficulty of testing a program like this is that you have to wait for all combinations of dice to come up, but, since this is a random process, you could be waiting for a very long time. In order to test the program more easily, add code that lets you input the numbers for the dice directly:

```
int die1 = // your java code to generate random number
int die2 = // your java code to generate random number
/* --- TESTING --- */
System.out.print("TEST - Enter values for dice: ");
die1 = in.nextInt(); // where "in" is your Scanner
die2 = in.nextInt();
in.nextLine(); // clear buffer in advance of String input
/* --- TESTING --- */
```

Your program output now looks like this when the program is run twice:

```
TEST- Enter values for dice: 3 7
You rolled 3 and 7 - your point is 10.

TEST - Enter values for dice: 6 5
You rolled 6 and 5 - you win!
```

Remember to remove your testing code when you finish!

Exercise 5.9 In this exercise, you will modify the program from the preceding exercise to describe the results of the dice roll in words, using this table (adapted from https://en.wikipedia.org/wiki/Craps):

	1	2	3	4	5	6
1	Snake Eyes					
2	Ace Deuce	Hard Four				
3	Easy Four	Fever Five	Hard Six			
4	Fever Five	Easy Six	Natural	Hard Eight		
5	Easy Six	Natural	Easy Eight	Nina	Hard Ten	
6	Natural	Easy Eight	Nina	Easy Ten	Yo-leven	Boxcars

Now the output from the program might look like this:

```
You rolled 3 and 5: Easy Eight - your point is 8.

You rolled 1 and 1: Snake Eyes - you lose, sorry.

You rolled 3 and 4: Natural - you win!
```

Hints: For the "Easy" versus "Hard" combinations, if the dice values are equal, it's "hard" (in the sense of being more difficult to get that combination); all the others are considered "easy."

Do **not** take the preceding program and blindly add code to it. Instead, carefully think through the new cases you must handle and the best way to handle them. (A 36-way `if` statement is definitely not your best choice.) Take time to plan it out! You might find that you have to re-structure your code to

some degree to add this new feature. That's OK—that happens all the time when programming.