Building Java Programs

Chapter 4
Lecture 4-2: Advanced if/else; cumulative algorithms; printf

reading: 4.1 - 4.5

Misuse of if

What's wrong with the following code?

```java
Scanner console = new Scanner(System.in);
System.out.print("What percentage did you earn? ");
int percent = console.nextInt();
if (percent >= 90) {
    System.out.println("You got an A!");
}
if (percent >= 80) {
    System.out.println("You got a B!");
}
if (percent >= 70) {
    System.out.println("You got a C!");
}
if (percent >= 60) {
    System.out.println("You got a D!");
}
if (percent < 60) {
    System.out.println("You got an F!");
}
...
Nested if/else

Chooses between outcomes using many tests

if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else {
    statement(s);
}

Example:
if (x > 0) {
    System.out.println("Positive");
} else if (x < 0) {
    System.out.println("Negative");
} else {
    System.out.println("Zero");
}

Nested if/else/else

- If it ends with else, exactly one path must be taken.
- If it ends with if, the code might not execute any path.

if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else if (test) {
    statement(s);
}

Example:
if (place == 1) {
    System.out.println("Gold medal!");
} else if (place == 2) {
    System.out.println("Silver medal!");
} else if (place == 3) {
    System.out.println("Bronze medal.");
}
Nested if structures

- exactly 1 path (mutually exclusive)
  if (test) {
    statement(s);
  } else if (test) {
    statement(s);
  } else {
    statement(s);
  }

- 0 or 1 path (mutually exclusive)
  if (test) {
    statement(s);
  } else if (test) {
    statement(s);
  } else if (test) {
    statement(s);
  }

- 0, 1, or many paths (independent tests; not exclusive)
  if (test) {
    statement(s);
  }
  if (test) {
    statement(s);
  }
  if (test) {
    statement(s);
  }

Which nested if/else?

- (1) if/if/if  (2) nested if/else  (3) nested if/else/if

- Whether a user is lower, middle, or upper-class based on income.
  - (2) nested if / else if / else

- Whether you made the dean's list (GPA ≥ 3.8) or honor roll (3.5-3.8).
  - (3) nested if / else if

- Whether a number is divisible by 2, 3, and/or 5.
  - (1) sequential if / if / if

- Computing a grade of A, B, C, D, or F based on a percentage.
  - (2) nested if / else if / else if / else if / else
if/else with return

- The following two versions of a max method don't compile:

```java
public static int max(int a, int b) {
    if (a > b) {
        return a;
    }
    // Error: not all paths return a value
}

public static int max(int a, int b) {
    if (a > b) {
        return a;
    } else if (b >= a) {
        return b;
    }
}
```

- The compiler thinks if/else/if code might skip all paths, even though mathematically it must choose one or the other.

All paths must return

- This version of max does compile and works:

```java
// Returns the larger of the two given integers.
public static int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

- Methods can return different values using if/else
  - Whichever path the code enters, it will return that value.
  - Returning a value causes a method to immediately exit.
  - All paths through the code must reach a return statement.
Cumulative algorithms

reading: 4.2

Adding many numbers

• How would you find the sum of all integers from 1-1000?

    // This may require a lot of typing
    int sum = 1 + 2 + 3 + 4 + ... + 999 + 1000;
    System.out.println("The sum is " + sum);

• What if we want the sum from 1 - 1,000,000? Or the sum up to any maximum?
  • How can we generalize the above code?
A failed attempt

- An incorrect solution for summing 1-1000:
  
  ```java
  for (int i = 1; i <= 1000; i++) {
    int sum = 0;
    sum = sum + i;
  }
  // error: sum is undefined here
  System.out.println("The sum is " + sum);
  ```

- sum's scope is in the for loop, so the code does not compile.

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The sum above is an incorrect attempt at a cumulative sum.

Corrected cumulative sum

```java
int sum = 0;
for (int i = 1; i <= 1000; i++) {
  sum = sum + i;
}
System.out.println("The sum is " + sum);
```

- Cumulative sum variables must be declared outside the loops that update them, so that they will still exist after the loop.
Cumulative product

- This cumulative idea can be used with other operators:

```java
int product = 1;
for (int i = 1; i <= 20; i++) {
    product = product * 2;
}
System.out.println("2 ^ 20 = " + product);
```

- How would we make the base and exponent adjustable?

Cumulative sum question

- Modify the Receipt program from Ch 2 (tax 8%, tip 15%).
  - Prompt for how many people, and each person's dinner cost.
  - Use static methods to structure the solution.

- Example log of execution:

```
How many people ate? 4
Person #1: How much did your dinner cost? 20.00
Person #2: How much did your dinner cost? 15
Person #3: How much did your dinner cost? 30.0
Person #4: How much did your dinner cost? 10.00

Subtotal: $ 75.00
Tax: $ 6.00
Tip: $ 11.25
Total: $ 92.25
```
Formatting text with printf

System.out.printf("format string", parameters);

- A format string can contain placeholders to insert parameters:
  - `%d` integer
  - `%f` real number
  - `%s` string
    - these placeholders are used instead of + concatenation

- Example:
  ```java
  int x = 3;
  int y = -17;
  System.out.printf("x is %d and y is %d\n", x, y);
  // x is 3 and y is -17!
  ```

  - printf does not drop to the next line unless you write `\n`

printf width

- `%Wd` integer, W characters wide, right-aligned
- `%Wd` integer, W characters wide, left-aligned
- `%Wf` real number, W characters wide, right-aligned
- ...

```java
for (int i = 1; i <= 3; i++) {
  for (int j = 1; j <= 10; j++) {
    System.out.printf("%4d", (i * j));
  }
  System.out.println(); // to end the line
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
```
printf precision

- \%Df \hspace{1em} real number, rounded to D digits after decimal
- \%W.Df \hspace{1em} real number, W chars wide, D digits after decimal
- \%-W.Df \hspace{1em} real number, W wide (left-align), D after decimal

\begin{verbatim}
double gpa = 3.253764;
System.out.printf("your GPA is %.1f\n", gpa);
System.out.printf("more precisely: %8.3f\n", gpa);
\end{verbatim}

Output:
your GPA is 3.3
more precisely: 3.254

Cumulative sum answer

// This program enhances our Receipt program using a cumulative sum.
import java.util.*;
public class Receipt2 {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        double subtotal = meals(console);
        results(subtotal);
    }

    // Prompts for number of people and returns total meal subtotal.
    public static double meals(Scanner console) {
        System.out.print("How many people ate? ");
        int people = console.nextInt();
        double subtotal = 0.0; // cumulative sum
        for (int i = 1; i <= people; i++) {
            System.out.print("Person #\" + i + ": How much did your dinner cost? ");
            double personCost = console.nextDouble();
            subtotal = subtotal + personCost; // add to sum
        }
        return subtotal;
    }
    ...
}
printf answer (partial)

... 

// Calculates total owed, assuming 8% tax and 15% tip 
public static void results(double subtotal) { 
    double tax = subtotal * .08; 
    double tip = subtotal * .15; 
    double total = subtotal + tax + tip; 
    System.out.printf("Subtotal: $%6.2f\n", subtotal); 
    System.out.printf("Tax: $%6.2f\n", tax); 
    System.out.printf("Tip: $%6.2f\n", tip); 
    System.out.printf("Total: $%6.2f\n", total); 
} 