Building Java Programs

Chapter 4
Lecture 4-1: Scanner; if/else

reading: 3.3 – 3.4, 4.1, 4.5
A Guide to Understanding Flow Charts
Presented in Flow Chart Form

START

DO YOU UNDERSTAND FLOW CHARTS?

NO

OKAY. YOU SEE THE LINE LABELED "YES"?

NO

BUT YOU SEE THE ONES LABELED "NO".

NO

LISTEN.

I HATE YOU.

YES

GOOD

YES

LET'S GO DRINK.

6 DRINKS

HEY, I SHOULD TRY INSTALLING FreeBSD!

NON

...AND YOU CAN SEE THE ONES LABELED "NO"?

NO

SCREW IT.

(THAT WASN'T A QUESTION.)

YES

WAIT, WHAT?

NO

BUT YOU JUST FOLLOWED THEM TWICE!

YES

(THAT WASN'T A QUESTION.)
Interactive Programs with Scanner

reading: 3.3 - 3.4
Interactive programs

**interactive program**: Reads input from the console.

- While the program runs, it asks the user to type input.
- The input typed by the user is stored in variables in the code.

- Can be tricky; users are unpredictable and misbehave.
- But interactive programs have more interesting behavior.
Exercise

• In physics, the *displacement* of a moving body represents its change in position over time while accelerating.
  • Given initial velocity $v_0$ in m/s, acceleration $a$ in m/s$^2$, and elapsed time $t$ in s, the displacement of the body is:
    • Displacement = $v_0 t + \frac{1}{2} a t^2$

• Write a method `displacement` that accepts $v_0$, $a$, and $t$ and computes and returns the change in position.
  • example: `displacement(3.0, 4.0, 5.0)` returns 65.0
**Scanner**

- **Scanner**: An object that can read input from many sources.
  - Communicates with `System.in`
  - Can also read from files (Ch. 6), web sites, databases, ...

- **The Scanner class is found in the java.util package.**
  ```java
class Scanner {
    // scanner methods...
}
```

- **Constructing a Scanner object to read console input:**
  ```java
  Scanner name = new Scanner(System.in);
  ```

- **Example:**
  ```java
  Scanner console = new Scanner(System.in);
  ```
Each method waits until the user presses Enter. The value typed by the user is returned.

- **nextInt()** reads an int from the user and returns it
- **nextDouble()** reads a double from the user
- **next()** reads a one-word String from the user
- **nextLine()** reads a one-line String from the user

```
System.out.print("How old are you? "); // prompt
int age = console.nextInt();
System.out.println("You typed " + age);
```

- **prompt**: A message telling the user what input to type.
Scanner example

import java.util.*;  // so that I can use Scanner

public class UserInputExample {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("How old are you? ");
        int age = console.nextInt();

        int years = 65 - age;
        System.out.println(years + " years until retirement!");
    }
}

• Console (user input underlined):
  How old are you? 29
  36 years until retirement!
Input tokens

- **token**: A unit of user input, as read by the `Scanner`.
  - Tokens are separated by *whitespace* (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?
    23  John Smith  42.0  "Hello world"  $2.50  "  19"

- When a token is not the type you ask for, it crashes.

```java
System.out.print("What is your age? ");
int age = console.nextInt();
```

**Output:**

*What is your age? Timmy*

```java
java.util.InputMismatchException
  at java.util.Scanner.next(Unknown Source)
  at java.util.Scanner.nextInt(Unknown Source)
  ...
Scanner example 2

```java
import java.util.*;    // so that I can use Scanner

public class ScannerMultiply {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();

        int product = num1 * num2;
        System.out.println(\"The product is \" + product);
    }
}
```

- Output (user input underlined):

  Please type two numbers: 8 6
  The product is 48

- The Scanner can read multiple values from one line.
Scanners as parameters

- If many methods need to read input, declare a `Scanner` in `main` and pass it to the other methods as a parameter.

```java
public static void main(String[] args) {
    Scanner console = new Scanner(System.in);
    int sum = readSum3(console);
    System.out.println("The sum is " + sum);
}

// Prompts for 3 numbers and returns their sum.
public static int readSum3(Scanner console) {
    System.out.print("Type 3 numbers: ");
    int num1 = console.nextInt();
    int num2 = console.nextInt();
    int num3 = console.nextInt();
    return num1 + num2 + num3;
}
```
The if/else statement

reading: 4.1, 4.5
The if statement

Executes a block of statements only if a test is true

```java
if (test) {
    statement;
    ...
    statement;
}
```

- Example:

```java
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Application accepted.");
}
```
Relational expressions

- if statements and for loops both use logical tests.

```java
for (int i = 1; i <= 10; i++) { ... 
if (i <= 10) { ... 
```

- These are boolean expressions, seen in Ch. 5.

- Tests use relational operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equals</td>
<td>1 + 1 == 2</td>
<td>true</td>
</tr>
<tr>
<td>!=</td>
<td>does not equal</td>
<td>3.2 != 2.5</td>
<td>true</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>10 &lt; 5</td>
<td>false</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>10 &gt; 5</td>
<td>true</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>126 &lt;= 100</td>
<td>false</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>5.0 &gt;= 5.0</td>
<td>true</td>
</tr>
</tbody>
</table>
The *if/else* statement

*Executes one block if a test is true, another if false*

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

- **Example:**
  ```java
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Welcome to Mars University!");
} else {
    System.out.println("Application denied.");
}
```
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 \texttt{if/else}

\textit{Chooses between outcomes using many tests}

\begin{verbatim}
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else {
    statement(s);
}
\end{verbatim}

- Example:

\begin{verbatim}
if (x > 0) {
    System.out.println("Positive");
} else if (x < 0) {
    System.out.println("Negative");
} else {
    System.out.println("Zero");
}
\end{verbatim}
Nested if/else if

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

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

- Example:

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

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

- 0, 1, or many paths  (independent tests; not exclusive)
  
  ```java
  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