"There are only two kinds of languages: the ones people complain about and the ones nobody uses."

— Bjarne Stroustrup, creator of C++
Input and \texttt{System.in}

\begin{itemize}
  \item \textbf{interactive program}: Reads input from the console.
    \begin{itemize}
      \item While the program runs, it asks the user to type input.
      \item The input typed by the user is stored in variables in the code.
      \item Can be tricky; users are unpredictable and misbehave.
      \item But interactive programs have more interesting behavior.
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item \textbf{Scanner}: An object that can read input from many sources.
    \begin{itemize}
      \item Communicates with \texttt{System.in}
      \item Can also read from files (Ch. 6), web sites, databases, ...
    \end{itemize}
\end{itemize}
The `Scanner` class is found in the `java.util` package.

```java
import java.util.Scanner;
```

Constructing a `Scanner` object to read console input:

```java
Scanner name = new Scanner(System.in);
```

- Example:
  ```java
  Scanner console = new Scanner(System.in);
  ```
### Scanner methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nextInt()</td>
<td>reads an int from the user and returns it</td>
</tr>
<tr>
<td>nextDouble()</td>
<td>reads a double from the user</td>
</tr>
<tr>
<td>nextLine()</td>
<td>reads a one-line String from the user</td>
</tr>
<tr>
<td>next()</td>
<td>reads a one-word String from the user. Avoid when Scanner connected to System.in</td>
</tr>
</tbody>
</table>

– Each method waits until the user presses Enter.
– The value typed by the user is returned.

– **prompt**: A message telling the user what input to type.

```java
System.out.print("How old are you? "); // prompt
int age = console.nextInt();
System.out.println("You typed "+ age);
```
import java.util.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!
The Scanner can read multiple values from one line.

```java
import java.util.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
Clicker 1 - 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"
```

A. 2        B. 6        C. 7
D. 8        E. 9
When a token is the wrong type, the program crashes. (runtime error)

```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)
...
```
The if/else statement

reading: 4.1
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.");
}
```
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.");
}
```
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>
Logical operators

- Tests can be combined using *logical operators*:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
<td>(2 == 3) &amp;&amp; (-1 &lt; 5)</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td>!</td>
<td>not</td>
<td>!(2 == 3)</td>
<td>true</td>
</tr>
</tbody>
</table>

- "Truth tables" for each, used with logical values $p$ and $q$:

| p   | q   | p && q | p || q |
|-----|-----|--------|--------|
| true| true| true   | true   |
| true| false| false  | true   |
| false| true| false  | true   |
| false| false| false  | false  |

<table>
<thead>
<tr>
<th>p</th>
<th>!p</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Nested if/else

Chooses between outcomes using many tests

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

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

- Write a method that prints out if it is good weather to go for a bike ride. The weather is good if the temperature is between 40 degrees and 100 degrees inclusive unless it is raining, in which case the temperature must be between 70 degrees and 110 degrees inclusive.

- Write a method that returns the largest of three numbers using if statements.

- Write a method that determines if one day is before another day (given month and day).
Exercise

- Prompt the user to enter two people's heights in inches.
  - Each person should be classified as one of the following:
    - short (under 5'3'')
    - medium (5'3'' to 5'11'')
    - tall (6' or over)
  
  - The program should end by printing which person is taller.

Height in feet and inches: 5 7
You are medium.

Height in feet and inches: 6 1
You are tall.

Person #2 is taller than person #1.
Exercises

- Write a method that simulates rolling 2 six-sided dice a given number of times and returns the number of times a given value is the sum of the two dice when rolled.

- Write a method that determines if a number is a perfect number. A perfect number equals the sum of its integer divisors, excluding itself:
  
  6 = 1 + 2 + 3, perfect
  8 > 1 + 2 + 4, deficient
  12 < 1 + 2 + 3 + 4 + 6, excessive
Exercises

- Write a method that determines if we have time to go out for lunch. Inputs are distance to restaurant, average walking speed, time required to finish meal, time available, expected cost of meal, and money available.
  - Times are expressed as whole number of minutes.
  - Money is expressed as a double.