Limitations of sequential programming

- Cannot choose whether or not to perform a command/instruction
- Cannot perform the same command more than once
- Such programs are extremely limited!

Control Structures

- Allow a program to base its behavior on certain conditions
- Two kinds:
  - Conditional (If) Statements
  - Loop Structures

Recap: Boolean

- **Boolean** is one of the primitive types
  - Only 2 values: true, or false
  - Booleans are used to make yes or no decisions
  - All control structures use Booleans

- The following expression each give a Boolean result:
  - \(25 > 24\) && \((12 == 13)\) //results to false
  - \(25 > 24\) || \((12 == 13)\) //results to true

- Thus based on certain conditions we can alter the outcome or flow of the program
Conditionals ("if" statements)

- An "if" statement is a *flow control* statement
- It is also called a *conditional*, or a branch
- We’ll see several “flavors”
  - An “if” all by itself
  - An “if” with an “else” part
  - An “if” with an “else if” part

```
//Assume x is an integer
if(x > 10) {
    x = x * 2;
    System.out.println("x = " + x);
}
```

"if" statement

```
if (condition){
    statement(s)
}
```

*If the condition is true, then the statement(s) (i.e. instructions) will be executed. Otherwise, it/they won’t.*

```
//Assume x is an integer
if(x > 0) {
    System.out.println(x + " is positive");
}
```

If statement (contd..)

- {} indicates the block of code that will get executed given the condition is true
- You can avoid the curly brace after condition if only one statement is to be performed
- If using Dr Java Interaction pane, best to use {}

```
//Assume x is an integer
if(x > 0) {
    System.out.println(x + " is positive");
}
```

"if-else" statement

```
if (condition){
    statement(s)
}
else {
    statement(s)
}
```

```
//Assume x is an integer
if(x > 0) {
    System.out.println(x + " is positive");
} else {
    System.out.println(x + " is negative");
}
```
If-else Flow chart

- If condition is true, go to Statement(1)
- If condition is false, go to Statement(s)

Style Rule: Indentation and Spacing

- Recommended indentation is from 2 to 4 spaces, but must be consistent throughout the program
- In Dr Java, you can set the indent level:
  - Edit > Preferences > Miscellaneous
- Single space around every binary operator, including comparisons and assignment (=)
  ```java
  if (x < 10) {
    x = x + 1;
  } else {
    x = x - 1;
  }
  ```

Cascading “if-else”

```java
//Assume variable score is entered by user
if (score > 90) {
  System.out.println("Grade A");
} else if (score > 80) {
  System.out.println("Grade B");
} else if (score > 65) {
  System.out.println("Grade C");
} else {
  System.out.println("F");
}
//Note: You can avoid the curly brace after condition if only one statement is to be performed
```

Nested if-statements

<table>
<thead>
<tr>
<th>An if within an if</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (condition1) {</td>
<td>What values must the conditions have in order for block A to run? B? C?</td>
</tr>
<tr>
<td>if (condition2) {</td>
<td></td>
</tr>
<tr>
<td>statement(s) A</td>
<td>A</td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>statement(s) B</td>
<td></td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>statement(s) C</td>
<td></td>
</tr>
<tr>
<td>} else {</td>
<td></td>
</tr>
</tbody>
</table>

ESE112 8 9 10 11
The infamous “dangling else”

<table>
<thead>
<tr>
<th>Code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (x &gt; y)</td>
<td>When is statementB executed?</td>
</tr>
<tr>
<td>if (y &lt; z)</td>
<td>In other words, which if is the else paired</td>
</tr>
<tr>
<td>statementA;</td>
<td>with?</td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>statementB;</td>
<td></td>
</tr>
</tbody>
</table>

- An else is paired with the last else-less if, regardless of spacing, unless {} dictate otherwise.

```c
if (x > y) {
    if (y < z) {
        statementA;
    }
}
else {
    statementB;
}
```