Outline

- Flowcharts
- If/Else Statements
- Boolean Expressions
Flowcharts

- Schematic representation of an algorithm
import java.util.*;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;
        Scanner keyboard = new Scanner(System.in);
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.println("The sum of those two numbers is");
        System.out.println(n1 + n2);
    }
}
Branching

- Add conditionals to control flow of program

- Example: going to school in the morning
Flowchart to Pseudocode

1. Leave home
2. Check weather
3. If it is sunny
   1. Walk to school
4. Otherwise
   1. Take the bus
5. Reach school
Flowchart to Pseudocode

1. Leave home
2. Check weather
3. If (sunny)
   1. Walk to school
4. Else
   1. Take the bus
5. Reach school
Branching Statement if/else

- Syntax
  - if (Boolean_Expression)
    Statement_1
  else
    Statement_2

- If value of Boolean_Expression is true, then perform Statement_1, else perform Statement_2
Branching Statement if/else

- Can execute multiple statements per case
  - if (Boolean_Expression)
    
    ```
    { 
      Statement_1a
      Statement_1b
      ...
    }
    ```
  
  else
  
  ```
  { 
    Statement_2a
    Statement_2b
    ...
  }
  ```

- Recommend using block syntax, even if just using one statement
Branching Statement if/else

- Else statement is optional
  - if (Boolean_Expression)
    Statement_1

- Only if value of Boolean_Expression is true, then perform Statement_1
Boolean Expressions

- An expression that evaluates to either true or false

Examples

- It is sunny today (true)
- 10 is larger than 5 (true)
- Today is Saturday (false)

- This sentence is false. (????)
  - Paradox!
Primitive Type: boolean

- Boolean expressions evaluate to a boolean type value
  - Can be either true or false

Example

- boolean sunny = true;
  boolean cloudy = false;
Java Comparison Operators

- When programming, Boolean expressions are often used for comparisons

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Is equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Is not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is strictly greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Is greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Is strictly less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Is less than or equal to</td>
</tr>
</tbody>
</table>

- Examples:
  - myInt <= 6
  - myDouble > 4.0
  - myChar == 'A'
import java.util.*;

public class FlowChart {
    public static void main(String[] args) {
        System.out.println("Give me an integer:");
        Scanner keyboard = new Scanner(System.in);
        int inputInt = keyboard.nextInt();

        if (inputInt > 10) {
            System.out.println("big number");
        } else {
            System.out.println("small number");
        }
    }
}

Prompt user for integer input

Is input greater than 10?

Yes

Print "big number"

No

Print "small number"
Gotchas with ==

- = VS. ==
  - var1 = var2  (assignment statement)
    - Compiler Error!
  - var1 == var2  (equality comparison)

- String Equality
  - Do not use == to compare strings (or any other class type)
    - string1 == string2  // BAD

  - Use the equals method instead
    - string1.equals(string2)  // GOOD
Boolean Operator: && (AND)

- True if **all** expressions are true

- Example
  - Only walk to school if it is between 50 and 75 degrees
    ```
    if ((temperature > 50) && (temperature < 75))
    {
        // walk to school
    }
    ```
Boolean Operator: || (OR)

- True if at least one expression is true

Example

- Walk to school if it is sunny or cloudy
  - if (isSunny || isCloudy)
    
    // walk to school
  

Boolean Operator: ! (NOT)

- Flips the state of a Boolean expression
  - !true is false
  - !false is true

- Example
  - Walk to school if it is not raining
    - if (!isRaining)
      {
        // walk to school
      }
Boolean Operator: ! (NOT)

- Flips the sense of a comparison

<table>
<thead>
<tr>
<th>Equivalent Expressions Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputInt1 == inputInt2</td>
</tr>
<tr>
<td>inputInt1 != inputInt2</td>
</tr>
<tr>
<td>inputInt1 &lt; inputInt2</td>
</tr>
<tr>
<td>inputInt1 &lt;= inputInt2</td>
</tr>
<tr>
<td>inputInt1 &gt; inputInt2</td>
</tr>
<tr>
<td>inputInt1 &gt;= inputInt2</td>
</tr>
</tbody>
</table>

- Works for more than just ints
Order of Operations

- From highest to lowest precedence
  - Unary: +, -, ++, --, !
  - Unary: (Type)
  - Binary: *, /, %
  - Binary: +, -
  - Binary: <, >, <=, >=
  - Binary: ==, !=
  - Binary: &&
  - Binary: ||
  - Binary: All assignment operators (=, +=, ...)
Prompt user for chairSize

chairSize > 10
Print “too big”

chairSize < 5
Print “too small”

(chairSize >= 5) && (chairSize <= 10)
Print “just right”
if (chairSize < 5) {
    System.out.println("too small");
}
else if (chairSize > 10) {
    System.out.println("too large");
}
else {
    System.out.println("just right");
}
Nested ifs

```java
if (chairSize < 5)
{
    System.out.println("too small");
}
else if (chairSize > 10)
{
    System.out.println("too large");
    if (chairSize > 100)
    {
        System.out.println("way too large");
    }
}
else
{
    System.out.println("just right");
}
```
Questions?
Logistics

Next:
- Lab 2: String Manipulation

Tomorrow:
- More Branching Statements
- Loops