Flow of Control - Branching

Cheng, Wei  COMP110-001  May 19, 2014
Today’s Contents

• If ... else ....
• Boolean expressions
Assignment 1

• Don’t forget to include the required program header (where you make pledge)

• Review the honor code and understand the consequences of violating honor code!

• Program plagiarism will be suspected if an assignment that calls for individual work results in two or more programs so similar that one can be converted to another by purely mechanical transformations.
  (changing variable name / comments / simple output messages does not make you look smart)

• Examples of Cheating (not exhaustive)
  – Copying all or part of another person's program and submitting it as your own work.
  – Lending your program so that it may be copied.
Flow of Control

- *Flow of control* is the order in which a program performs actions.
- Up to this point, the order has been sequential.
Flow of control

• There are multiple types of control:

  – continuation (unconditional)
  – executing a set of statements only if some condition is met (choice)
  – executing a set of statements zero or more times, until some condition is met (loop)

  – ...


Flow of control

• continuation (unconditional)

• Perform actions sequentially

• A single path in the flow chart
Flow of control

• Branching

• Choose one path in the flow chart by checking conditions
Flow of control

• Branching

• Choose one path in the flow chart by checking conditions

Leave Home → Check Time

- Before 7am?
  - Yes: Take Bus → Reach School
  - No: Take Subway
Another Example

• In assignment 1

Compute #days

Print #days

#days==1

Yes

Print “day”

No

Print “days”

Compute #hours
The **if-else** Statement

- A branching statement that chooses between two possible actions.
- Syntax

  ```java
  if (Boolean_Expression)
      Statement_1
  else
      Statement_2
  ```

  Example:

  ```java
  System.out.print( days );
  if (days==1)
      System.out.print(" day");
  else
      System.out.print(" days");
  ```
The **if-else** Statement

- What if you have multiple statements for each branch?

```java
if (Boolean_Expression) {
    Statement_1.1
    Statement_1.2
    ...
} else {
    Statement_2.1
    Statement_2.2
    ...
}
```

*My suggestion: always enclose branches with curly brackets*
import java.util.*;

public class ControlFlowExample
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        int inputInt = keyboard.nextInt();

        if (inputInt > 5)
        {
            System.out.println("Big number");
        } else {
            System.out.println("Small number");
        }
    }
}
### Java Comparison Operators for Primitive types

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>Equal to</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal to</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>
Boolean Expressions

• True of False

• “atomic” expressions
  
  \[ 5 == 3 \] // always false
  \[ myInt <= 6 \] // depends on the value of myInt
  \[ myInt != anotherInt \] // depends on the value of myInt, anotherInt

• Composite expressions:
  
  \( (myInt <= 6) \&\& (myInt!=anotherInt) \)
Review of all boolean operations

<table>
<thead>
<tr>
<th>Value of $A$</th>
<th>Value of $B$</th>
<th>Value of $A &amp;&amp; B$</th>
<th>Value of $A \mid\mid B$</th>
<th>Value of $! (A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Example in Assignment 1

• Check if #days is not 1:
  
  \[
  \text{days} \neq 1
  \]
  
  \[
  \text{days} < 1 \text{ || days} > 1
  \]
  
  \[
  \neg (\text{days} == 1)
  \]
  
  \[
  \neg (\text{days} \geq 1 \text{ && days} \leq 1)
  \]

  What about:

  \[
  \text{days} < 1 \text{ && days} > 1
  \]
  
  \[
  \text{days} <= 1 \text{ && days} >= 1
  \]
  
  \[
  \text{days} <= 1 \text{ || days} > 1
  \]
Comparison for Objects (including String)

- Call `.equals()` method
- Don’t use `==`

```java
String name1 = “COMP110”;
String name2 = new String(“COMP110”); // tell Java to create a new instance

if (name1 == name2) {
    System.out.println(“Same”);
} else {
    System.out.println(“Different”); // you will always get this
}
```
Comparison for Objects (including String)

- Call `Object.equals()` method
- Don’t use `==`

```java
String name1 = “COMP110”;
String name2 = new String(“COMP110”); // tell Java to create a new instance

if (name1.equals(name2) ) {
    System.out.println(“Same”); // Now this branch is selected
} else {
    System.out.println(“Different”);
}
```
If Without Else

- You can use just an if statement

```java
if (Boolean_Expression) {
    Statement_1.1
    Statement_1.2
    ...
}
the rest of your code
```

Wake up

Check Time

Before 7am?

Yes

Have Breakfast

No

Leave Home
A different approach to solve Assignment 1

- Print 
  
  - #days == 1
    - Yes: Print “day”
    - No: Print “days”

- DayText = “days”
  - Yes: Print DayText
  - No: #days == 1
    - Yes: DayText = “day”
    - No: DayText = “days”

A different approach to solve Assignment 1

```java
System.out.print( days );
String dayText = " days";
if (days==1){
    DayText = " day";
}
System.out.print( dayText );
```

Yes

No

Print dayText
Next Class

• Nested if...else...
• More Java examples