CS133: Developing Programming Principles

Lecture 3

Programming Style, Boolean Expressions, If Statements

Documentation and style

- Make sure you use **meaningful** variable names.
  - classes, variables: noun phrases
  - methods: verb phrases
- Add comments to make your code easier to understand (i.e., use English)
- Add comments to your program to identify who wrote it, what it does, etc.

Commented example

```
import java.util.*;
/** Hello program
 *  @author Student McStudentson
 *  @version 2
 *  Get the name of the user and output it along with a greeting.
 */
public class Example
{
  public static void main (String[] args)
  { // Set up keyboard input
    Scanner myScanner = new Scanner(System.in);

    // Input name
    System.out.println("Enter your first name: ");
    String name = myScanner.nextLine();

    System.out.println("Hello, " + name + "!");
  }
}
```


Programming style tips

- { and } should line up in the same column
  - DrJava does this for you! (<tab> key)
- Place one **instruction** per line.
- Use whitespace to separate code blocks
  - Briefly comment each code block
- Comment and format code as you program

`if-else` statements

- **Formal definition:** A statement which allows a program to choose an action depending on the value of a boolean expression.
- Example in plain English:

  If the temperature is greater than zero it will rain. Otherwise, it will snow.

`if-else` statements continued

- Syntax:

```
if (Boolean_Expression)
{
   Statement_List_1;
}
else
{
   Statement_List_2;
}
```

if-else statement example

- Example:

```
if (temperature > 0)
{
  System.out.println("It's raining.");
}
else
{
  System.out.println("It's snowing.");
}
```


Compound statements

- Put **{}** around multiple statements.
  - Around single statements too
- The result is a **compound statement** treated as a single statement.
- Most commonly used in **if-else** statements, as we had already seen, and loops.

Importance of { }

- Consider:

```
if (instruction.equals("QUIT"))
  System.out.println("Game over");
  System.out.println("The winner is " +
                winner);
```

- What is printed when instruction is **"QUIT"**? When it is **"CONTINUE"**?

Importance of { } continued

- Consider:

```
if (numGuesses < maxGuesses)
  if (guess == actualNum)
    System.out.println("Correct");
else
  System.out.println("Out of guesses.");
```

- What is displayed when guess is correct? When it is wrong? Guesses remaining?

Importance of { } continued

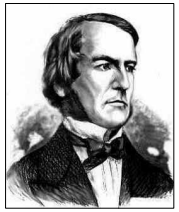
- The outcome of code can be significantly different than intended depending on the absence or presence of braces.
- Be sure to trace code as written, **not** as intended.
- **General Rule**: use braces in all **`if`** statements, even if only one statement within body of **`if`** statement

Boolean expressions

- Must give a true or false value when evaluated.

  Recall a boolean variable can be either **true** or **false**.

- Named after George Boole (1815-1864), inventor of Boolean algebra (using AND, NOT, OR with true/false values).



Boolean expressions

- either **`true`** or **`false`**.

Math notation	Name	Java notation	Java example
=	*Equal to*	==	`balance == 0`
≠	*Not equal to*	!=	`income != tax`
>	*Greater than*	>	`expenses > income`
≥	*Greater than or equal to*	>=	`points >= 60`
<	*Less than*	<	`pressure < max`
≤	*Less than or equal to*	<=	`expenses <= income`

From p. 135 of text

Logical operators

- Used to combine boolean expressions

Name	Java notation	Java example
AND	`&&`	`stoveOn == true && stoveHot == true`
OR	`||`	`height < 10 || height > 20`
NOT	`!`	`!(myString.length() <= 5)`

Using AND and OR

- **English example:**

  If the difference between the number of apples and oranges is at most 3, output, "Close". Otherwise, output, "Far".

Using AND and OR continued

```
int numApples, numOranges;
if ((numApples - numOranges <= 3
    && numApples - numOranges >= 0)
   ||
    (numOranges - numApples <= 3
    && numOranges - numApples >= 0))
{
  System.out.println("Close");
} else {
  System.out.println("Far");
}
```

Testing for equality

- When comparing primitive values, use ==.
- When comparing objects, use the **`equals`** method (usually).
- **`String`**s also have an **`equalsIgnoreCase`** method
- Generally, do not use == when comparing objects (like **`String`**s).
  - The meaning of == is different for objects.

Comparing **`String`**s

- Compare **`String`**s with **`equals(otherString)`** and **`equalsIgnoreCase(otherString)`**.
- Example:

```
String first = myScanner.nextLine();
String second = myScanner.nextLine();
if (first.equals(second))
{
  System.out.println("The two strings are identical.");
}
else
{
  System.out.println("The two strings are different.");
}
```

Other forms of **`if-else`** statements

- **No `else` part**

  Use if you want some code to be executed only if a condition is true. Same as having an empty **`else`** statement.

  Example:

```
if (temperature > 32)
{
  System.out.println("It's hot");
  System.out.println("I'm melting!");
}
```


Nested statements

- We can put statements inside each other.
- There is no limit to how many times this can be done.

Nested statements continued

```
if (numGuesses < maxGuesses) {
  if (guess == actualNum)
   {
     System.out.println("Correct");
   } else {
     System.out.println("Wrong");
   }
} else {
  System.out.println("Out of guesses.");
}
```


Multibranch statements

- We can have more than two choices in an **`if`**-**`else`** statement.
- Exactly one branch will be chosen.

Multibranch statements continued

- Example:

```
if (grade == 'A') {
    System.out.println("Excellent!");
} else if (grade == 'B') {
    System.out.println("Good");
} else if (grade == 'C') {
    System.out.println("Passed");
} else if (grade == 'D') {
   System.out.println("Passed");
} else if (grade == 'F') {
   System.out.println("Failed");
}
```


Multibranch statements continued

- Example, rewritten:

```
if (grade == 'A') {
    System.out.println("Excellent!");
} else if (grade == 'B') {
    System.out.println("Good");
} else if (grade == 'F') {
    System.out.println("Failed");
} else {
   System.out.println("Passed");
}
```

ERROR: undefined
OFFENDING COMMAND:

STACK: