Intro to if conditional construct

How to introduce control of flow in code

Topics

- Simple use of the most common Decision Construct – The if statement
  - A means of controlling flow of program
- Operators
  - Purpose of each
  - Precedence

Up to this point ...

- Programs have been simple
  - Variables of fixed value
  - Calculations the same
  - No real decision making process
- Need to introduce some means of controlling flow of program
  - By the end of Course, with knowledge of methods, classes etc., more control will be known
  - But must start at the basic point ... if construct

Cotrolling flow of Program

- Some means of decision making, for example
  - Voting machine. If age < 18 then don't allow them to vote
  - Climate Control. If temp > 30, turn on air conditioning

Operators

- For those who have programmed before ...
- Operators same as Other languages:
  - Arithmetic
    - `int i = i+1; i++; i--; i *= 2;`
    - `+, -, *, /, %,`
  - Relational and Logical
    - `<, >, <=, >=, ==, !=`
    - `&&, ||, &, |, !`
- **WE WILL EXAMINE EACH IN DETAIL!!!**

Operators II - Relational

Relational

- **< Less than**
- **> Greater than**
- **<= Less than or equal**
- **>= Greater than or equal**
- **== Equal to!**
  - **DO NOT CONFUSE WITH = ASSIGN OPERATOR!**
- **!= Not Equal to!**

Operators III - Logical

Logical – The first 4 are used when testing two or more tests

e.g. If age > 18 and Irish Citizen in a voting context

Short Circuit Operators

&& =>AND

|| => OR

& => AND

| => OR

Will return to significance of Short Circuit Operators later in course, but basically used if one test more crucial than second e.g. if person is < 18, no point checking if Irish citizen

! => FALSE … Used with boolean VARIABLES ONLY!

Conditional Constructs

Syntax same as in most other languages:

- **if (TEST) {CODE TO BE RUN IF TEST = TRUE }**
  - **Simple if**
- **if ( ) { } else { }**
  - **If with an else to grab all other scenarios**
- **if ( ) { } else if ( ){ }**
  - **If with a secondary if condition i.e. two test conditions**

```
if ( ) { } else if ( ){ } else {}
```

Important Parts

- Use of word *if* (A reserved word)
- ( )'s to enclose condition what we are testing
- { }'s to enclose what code is to be executed if condition is true
  - The {}'s can be **excluded** but must be careful
  - If {}'s are excluded, then the if controls whether the next line of code will be executed or not **only**
  - For simplicity, it is better to include { }'s

Use of ;

- Ordinarily ; would denote an end of a line of code
- For if's and indeed many other constructs, they are not needed for the first line
  - The for loop is another example where the ; does not need to be stated

if ... else

- The if ... else is used when you need to test for one condition but might want to do something else if that condition is not met
  - e..g Voting Age. You may wish only to print out an error message if < 18, but if this is not true, then you may wish to run the rest of the program

If ... else II

```
public static void main(String[] args) {

    int user = 17;

    if (user <= 18) {
        System.out.println("User is 18 or younger");
    }
    else {
        System.out.println("User is older than 18");
    }
}
```

Controlling the flow of the Program

- As stated at the start of the presentation, if's are used to control the flow of the program
- IT IS VITAL THAT YOU MAKE SURE THAT THE PROGRAM
  - Works
  - Does the right thing!
- This is especially true when using Conditional Constructs
  - Need to ensure that they govern only the blocks of code that are needed

Program Flow

- For an if statement, this is governed by the positioning of the { }'s, especially the } at the end of the block of code controlled by the if
- The positioning of the { }'s is VITAL!!!
  - It is not simply a case of having the required amount of {}'s

Booleans and if statements

- Boolean variables essentially represent a true or false state which can be then utilised for on/off, yes/no scenarios
- By default, unless changed, default state will be FALSE
- Once the required code is performed, the state of the boolean can then be changed to TRUE
  - For voting, if over 18 isEligible will be set to TRUE

Boolean and if Statements II

- isEligible = true // Settting isEligible to true
- However care should be taken when testing within confines of an if
  - if (isEligble = = true) ==> CORRECT!!!
  - if (isEligible = true) ==> INCORRECT BUT WILL STILL BE COMPILED!! How?
    - isEligible is assigned true and is then tested. So even it was false before the if, it will convert it to being true, thus making the if redundant

Booleans and if Statements III

- Shorter means of evaluating ...
  - if (isEligible) ==> Checks if isEligible is true
  - if(!isEligible) ==> Checks if isEligible is false
- They are both the same as their longer equivalents (if(isEligible ==true)), but are used more commonly

Comments … An aside

- Ensure that comments are used to mark code of importance
- Get into the habit of marking code for
  - Yourself
  - Others

Links

- If ... else & if ... else if
  - http://www.homeandlearn.co.uk/java/java_if_else_statements.html
- Coding Conventions
  - http://java.sun.com/docs/codeconv/CodeConventions.pdf

Links II

- Site for overall Java basics
  - http://www.homeandlearn.co.uk/java/java.html