CS620c Structured Programming Lesson 10

Joe Duffin

Email: Joseph.Duffin@nuim.ie

A few reminders:

- Truth Tables
- AND
  - True AND True gives True
  - True AND False gives False
  - False AND True gives False
  - False AND False gives False
- OR
  - True OR True gives True
  - True OR False gives True
  - False OR True gives True
  - False OR False gives False

A few reminders:

- NOT FALSE means True
- NOT TRUE means False

- Operator Precedence
  - Generally speaking you read the expression from left to right
  - If everything has the same order of precedence (e.g. divide, modulus and multiply) you evaluate from left to right – that is the first operator from the left is performed first, followed by the second operator from the left …
  - Unless brackets are used. Brackets take precedence

Default Values

- Always give your variables a default value
  - **Data Type** **Default Value**
  - int 0
  - float 0.0f
  - double 0.0
  - char '\u0000'
  - boolean false

Memory Allocation

- What happens when I declare and initialise in memory?

Class Test

- Let’s have a look at some class test questions..

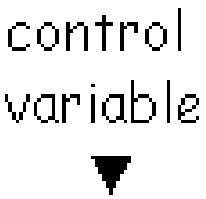
Loops (Iteration)

- Loops allow your program to do the same thing again and again and again and again. . .
- There are 3 looping statements in JAVA:-
  - while
  - for
  - do…while
- A loop repeatedly executes the same set of instructions until a finishing condition is met

Example

```
int count = 1; // start count at one
while ( count <= 3 )  // loop while count is <= 3
{
    System.out.println("Count is:" + count );
    count = count + 1; // add one to count
}
```

Let's take it apart..




```
int count = 1; // start count at one
while ( count <= 3 )// loop while count is <= 3
{
    System.out.println("Count is:" + count );
    count = count + 1; // add one to count
}
```

Here's how it works..

- The variable count is initially assigned a value of 1.
- The condition (count <= 3) is evaluated as true. Because the condition is true, the block statement following the while is executed.
  - The current value of count is written out:

    count is 1
  - count is incremented by one, to 2.

Cont…

- The condition (count <= 3) is re-evaluated and is still true so the block statement following the while is executed.
  - The current value of count is written out

  count is 2
  - count is incremented by one, to 3.

- The condition (count <= 3) is evaluated as true so the block statement following the while is executed.
  - The current value of count is written out.

  count is 3
  - count is incremented by one, to 4.

Cont..

- The condition (count <= 3) is evaluated as FALSE. Because the condition is FALSE, the block statement following the while is SKIPPED.

Important!

- There are three important parts to a loop:
  - The initial value (control variable etc)
  - The condition
  - Update
- PracticeLoops.java

What happens?

- Let's make just a change to the program
  - change the initialisation of count to:

```
int count = 1;
```

- What does the program print out?

Another change?

- How would I change the program so it prints downwards in two's from 20 to 0?

Another example

- Look at the following program fragment:

```
int count = 13;
int increment = 1;
while ( count >= 0 )
{
    System.out.println("Count is: " + count );
    count = count + increment;
}
System.out.println("Count was "+count + "when the condition become
    false");
```

- Look over each of the three parts of the counting loop

Answer

- Here's what's displayed…

```
count is 13
count is 14
count is 15
count is 16
count is 17
 .   .   .   .
```

and so on without end!

- It's an infinite loop…