

Lecture 14
CSE11 Fall 2013
For loops, Do While, Break, Continue

General Loops in Java

- Look at other loop constructions
- Very common while loop:
 - do a loop a fixed number of times (MAX in the example)

```
int count = 0;
while (count < MAX) {
    // processing statements
    count ++;
}
```

for loop

```
int count;  
for (count=0; count < MAX; count++)  
{  
    // processing statements  
}
```

- The loop parameters are upfront
 - A succinct way to “say” do the following “MAX times”
 - You don't have to just increment by 1
- DiagonalLoopGray2.java

for loop – general form

```
int count;  
for (initial statement; test condition; update  
statement)  
{  
    // processing statements  
}
```

- The loop parameters are upfront
 - Don't have to read to the bottom of the loop to figure out is a standard “counting loop”
 - single digit letters are commonly used for loop indices.
- DiagonalLoopGray2.java

How is for loop “translated” into a while loop?

```
for (initial statement; test condition; update
statement)
{
    // processing statements
}
```

This is Translated to:

```
initial statement;
while ( test condition)
{
    // processing statements
    update statement;
}
```

For loop steps

1) Initial statement executed

2) termination statement evaluated

3) if !terminated then

- execute statements
- execute update statement
- Go back to step 2)

The Loop Index

```
for (count=0; count < MAX; count++)  
{  
    // processing statements  
}
```

- `count` is the “loop index”, takes on values
- 0, 1, 2, 3, 4, ,MAX - 1

You don't have to just count up you can count Down

```
for (count=MAX; count > 0; count--)  
{  
    // processing statements  
}
```

- `count` (the “loop index”), takes on values
 - MAX, MAX -1, MAX -2 , ..., 1
- NOTE: the difference between this and previous
 - Both perform MAX loops
 - loop index is “off by one”
 - Always need to check starting and ending index for “errors” (important for Arrays in next lecture)
 - This is a common “bug” in loops

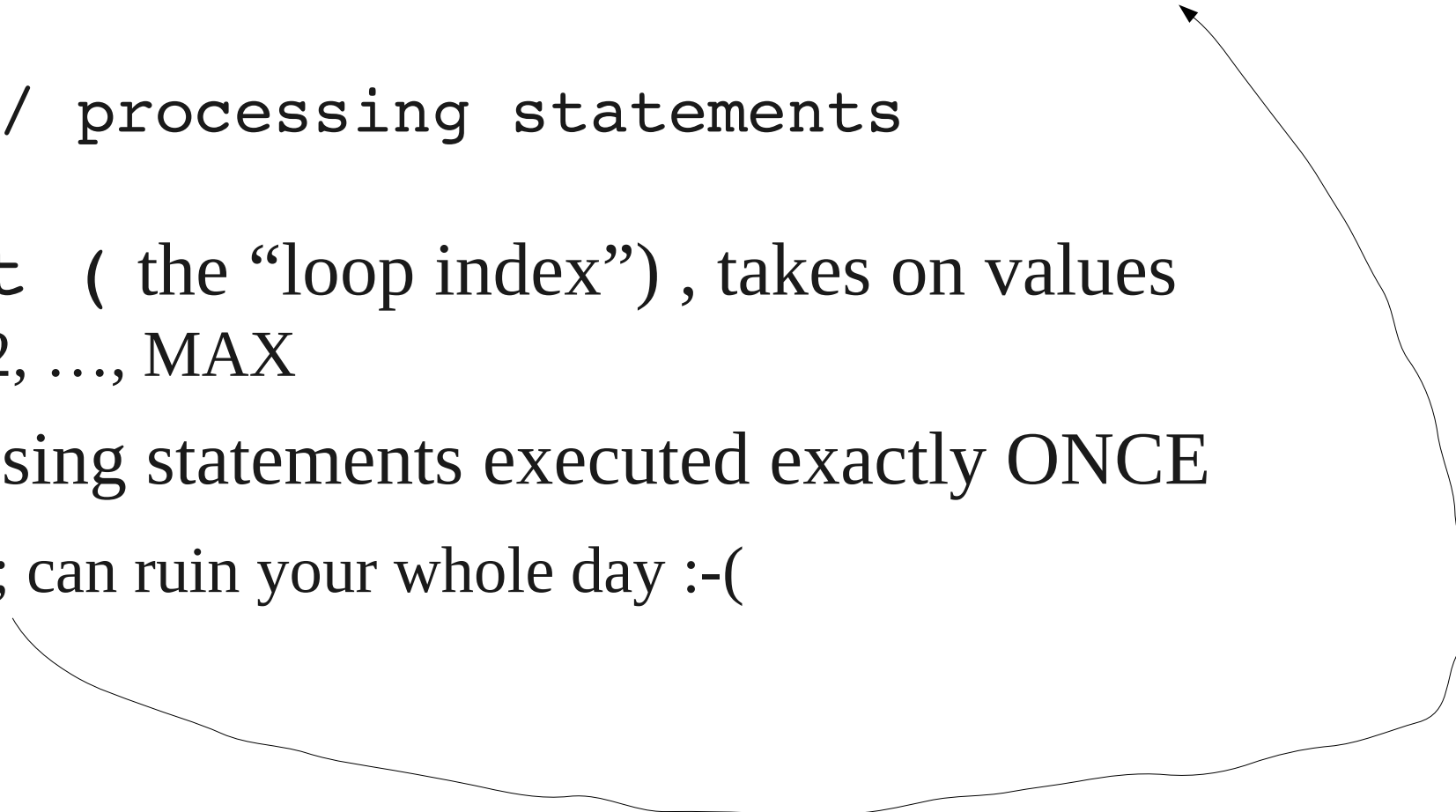
MAX + 1 iterations

```
for (count=0; count <= MAX; count++)  
{  
    // processing statements  
}
```

- `count` (the “loop index”), takes on values
- 0,1,2, ..., MAX
- NOTE: “MAX + 1” iterations
- get used to counting **starting from 0**, very common in Java and C

Common Error

```
for (count=0; count <= MAX; count++);  
{  
    // processing statements  
}
```

- `count` (the “loop index”), takes on values
 - 0,1,2, ..., MAX
 - Processing statements executed exactly ONCE
 - one ; can ruin your whole day :-)
- 

Subversive but legal

```
public class Subversive {  
    private static final int MAX=20;  
    public static void main(String [] args)  
    {  
        int count, j = 0;  
        for (count=0; count++ <= MAX; j++)  
        {  
            System.out.format("%d %d\n", count, j);  
        }  
    }  
}
```

- Confusing – takes longer to figure out what it does
- Can you guess the output?
- This might be a little too clever :-)

Loops with non-integer indexes

- Legal, but you may not get exactly what you expect

```
for (double count=0; count < MAX; count += 0.33333)
{
    // processing statements
}
```

- `count` is the “loop index”, takes on values
 - 0.0, 0.33333, 0.66666, 0.99999, 1.33332, 1.66665,
 - May not be exactly what you intended because doubles are only approximations of real numbers
 - This is called round-off error
 - When using non-integer loop indices, never check for exact equality as the termination condition. e.g. `count != .999999`)

Do...While

```
do
{
    // processing statements
}
while (condition);
```

- Do loop guaranteed to be executed *at least once*
- DoWhileEx.java

NestedLoops

- Just like nested while loops
 - You can nest a for loop inside of while loop and vice-versa.
 - When we do two-dimensional arrays, nested loops are quite common
 - Use this on your homework due friday
 - Nest can go arbitrarily deep, but seldom do you see more than three levels.
- Knitting.java example from book

break and continue

- `break` and `continue` are special keywords used inside of loops
 - `break` – immediately stop processing the loop, go to first statement that follows the entire loop.
 - `continue` – stop processing this iteration of the loop. If a for loop, execute the “update statement”, then go to the top of the loop. While/do While, go to the top of the loop
- Why write code with `break` and/or `continue`?
 - special handling of particular cases become apparent

A common use of break

- Code is searching for the **first** occurrence of a particular condition, but will only search for so long.

```
for (int i = 0; i < MAX; i++)
```

```
{
```

```
    if (Function(i) < 0)
```

```
        break;
```

```
}
```

```
if (i < MAX)
```

```
    System.out.println(i);
```

```
else
```

```
    System.out.println("Function is  $\geq 0$  ")
```


Common Use of Continue

- You only want to process occurrences that have (or have not) met a condition

```
Student pupil;
Course myClass;
while ((pupil = myClass.nextStudent()) != null)
{
    if (!pupil.tookMidterm())
        continue;
    // only proces if midterm was taken.
    grade = pupil.computeGrade();
    myClass.record(pupil.getName(), grade);
    pupil.emailGrade(grade);
}
```

Formatted Printing

- `System.out.format("format string", value1, value2, ... , valueN)`
- Method supports a variable number of arguments
- format string has *replaceable* format elements
 - The first replaceable element is assigned value1, the second value2, and so on.
 - %<char> is a placeholder for the type for format to perform
 - %d – format as an integer
 - %f – format as floating point number
 - %s – format as a string
 -
 -

Special Characters in the Format string

- `\n` – newline
- `\t` – tab
- `\r` – return (no new line)
- Example strings
- “`%d : %f \n`” – print first arg as an int, second as a float, print a newline.
 - `System.out.format(“%d : %f \n”, j, vX);`
- Can specify fixed width, too
- “`%5d`” would format an integer using at least 5 spaces
- a good “cheat-sheet” of format codes
 - <http://alvinalexander.com/programming/printf-format-cheat-sheet>
- Example: `FormatMe.java`