

Introduction to Programming

with Java, for Beginners

Do-while
Pseudo code
Continue
Break
Nested Loops

While vs. For

Code	Explanation
<pre>int x = 1; while (x <= 10){ System.out.println(x); x = x + 1; }</pre>	<i>An example of a while loop that has this pattern</i>

<pre>for (int x = 1; x <= 10; x = x + 1){ System.out.println(x); }</pre>	<i>A for loop that does the same thing</i>
---	--

Note: In a for loop a “missing” end-test or re-initialization condition evaluates to true (infinite loop)

ESE112

1

Another loop: Do-While

```
do {
    statement(s)
} while (condition);
```

- Do the statement/block at least once
- Evaluate the *condition*. If it is
 - true: re-execute *statement(s)*; repeat step 2
 - false: we're done with the loop

```
int x = 0;
do {
    x = x + 1;
}while (x < 3);
```

ESE112

2

Common semantic errors w/ loops

- Condition starting out to be false initially
- Wrong start/end condition
 - Start condition:
 - End condition e.g. < vs. <=
- Wrong Re-initialization condition
- No terminating condition
 - For preferred over while due the nature of syntax

ESE112

3

Formulating a Solution

- First Think *Algorithmically*
 - Well defined step by step procedure
 - Use *pseudocode* to write out your steps
 - English like code
 - It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax
- Then *Translate* the solution into programming language
 - Put together the components we have so far
 - declarations, assignments, control structures

ESE112

4

Example: Fibonacci sequences

- A Fibonacci sequence is an infinite list of integers
- The first two numbers are given
 - Usually (but not necessarily) these are 1 and 1
- Each subsequent number is the sum of the two preceding numbers:
1 1 2 3 5 8 13 21 34 55 89 144 ...
- Let's write a program to compute number is less than 1000

ESE112

5

Starting the Fibonacci sequence

- We need to initialize two numbers in the sequence
 - Set *first* to 1
 - Set *second* to 1
- We need to print these out:
 - Print *first* and *second*
- We need to compute and print the next number:
 - Set *next* to sum of *first* & *second*;
 - print *next*

ESE112

6

Taking the next step

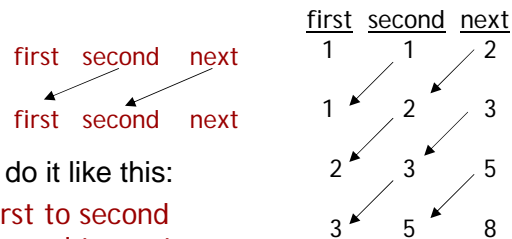
- Now what?
 - Need to add *second* and *next*
 - $nextnext = second + next$
 - What if the sequence is to long
 - I do want to make 100s of storage to hold each item
- The sequence so far is: *first* *second* *next*
 - Do I see a pattern emerging ?

ESE112

7

Preparing to make many steps

- We need to make these moves:



- We can do it like this:

Set first to second
Set second to next

- We can put these statements in a loop and do them as many times as we please

ESE112

8

Complete Psuedocode

Set *first* to 1

Set *second* to 1

Print first and second

while next number < 1000

Set next to sum of first & second;

print next

Set first to second

Set second to first

ESE112

9

Psuedocode Rules

- Can use words such as while, if else-if
 - E.g. for 1 to n
- Do not specify data declarations or types
- Use Words that specify an action such as set, reset, increment, compute, calculate, add, sum, multiply, print, getinput
- Use indentation for block of code i.e. {}

ESE112

10

Translate it to Programming Language

```
int first = 1;
int second = 1;
int next = 0;

System.out.print(first + " ");
System.out.print(second + " ");
while (next < 1000) {
    next = first + second;
    System.out.print(next + " ");
    first = second;
    second = next;
}
```

ESE112

11

Break and Continue Statements

- *break* and *continue* are Java statements
- Are also “flow control” statements
 - if, while, do-while, for, return
- A break “breaks you out” of the closest enclosing loop
- A continue is a shortcut to the next iteration of the loop
- A loop may have
 - Zero or more break statements
 - Zero or more continue statements

ESE112

12

while-loop with break, continue

```
while (condition1){  
    . . .  
    if (condition2)  
        continue; // go up and re-evaluate condition1  
    if (condition3)  
        break; // exit the loop  
    . . .  
}
```

// after a break statement, execution resumes here

ESE112

13

for-loop with break, continue

```
for (expr1; condition1; expr2){  
    . . .  
    if (condition2)  
        continue; // evaluate expr2, then condition1  
    if (condition3)  
        break; // exit the loop  
    . . .  
}
```

// after a break statement, execution resumes here

ESE112

14

Nested Loops

- A *nested loop* is a loop within a loop, an inner loop within the body of an outer one
 - Just like nested if-statements

ESE112

15

Example: Multiplication Table

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

ESE112

16

Example: Multiplication Table

```
int product = 0;
for (int i = 1; i < 11; i = i + 1) {
    for (int j = 1; j < 11; j = j + 1) {
        product = i * j;
        if (product < 10)
            System.out.print(" " + product);
        else
            System.out.print(" " + product);
    }
    System.out.println();
}
```

Some things to try out:

- For every "i" i.e. 1 to 10, how does the inner loop perform.
- change j to be j < 8
- Try to change the spacing in print commands to see what happens

ESE112

17