

Introduction to Programming

with Java, for Beginners

Control Structures

Sequential Control Flow

- Consists of just a list of commands to be done in order

Welcome to Dr. Java

```
> int num = 2;
> int sqNum;
> sqNum = num * num;
> System.out.println(sqNum); //Or just type:sqNum
2
```

ESE112

1/17

What are control Structures ?

- Limitations of sequential control flow
 - Cannot choose whether or not to perform a command/instruction
 - Cannot perform the same command more than once
- Such programs are extremely limited!
- **Control structures** allow a program to base its behavior on certain conditions

ESE112

2/17

Recap: Boolean

- **Boolean** is one of the eight primitive types
 - Only 2 value: true, or false
 - Booleans are used to make **yes** or **no** decisions
 - All control structures use Booleans
- The following expression each give a Boolean result:
(25 > 24) && (12 == 13) //results to **false**
(25 > 24) || (12 == 13) //results to **true**
- Thus based on certain conditions we can alter the outcome or flow of the program

ESE112

3/17

Conditionals ("if" statements)

- An "if" statement is a *flow control* statement
- It is also called a *conditional*, or a branch
- We'll see several "flavors"
 - An "if" all by itself
 - An "if" with an "else" part
 - An "if" with an "else if" part

ESE112

4/17

"if" statement

```
if (condition){  
    statement(s)  
}
```

If the condition is true, then the statement(s) (i.e. instructions) will be executed. Otherwise, it/they won't.

```
//Assume x is an integer  
if((x % 2) == 0){  
    System.out.println(x + " is even");  
}
```

ESE112

5/17

"if-else" statement

```
if (condition){  
    statement(s)  
}  
else {  
    statement(s)  
}
```

```
//Assume x is an integer  
if((x % 2) == 0){  
    System.out.println(x + " is even");  
}  
else {  
    System.out.println(x + " is odd");  
}
```

ESE112

6/17

Example: "if-else" statement

Example	Syntax & Explanation
<pre>//Determine smaller of two integers numbers & put it variable "min" int min; int x = 5; int y = 10; if (x <= y){ min = x; } else { min = y; }</pre>	<pre>if (condition){ statement(s) } else { statements(s) }</pre> <p><i>If the condition is true, then the statement(s) in the "if block" are executed.</i></p> <p><i>Otherwise, if there is an "else" part, the statement(s) in it are executed.</i></p>

ESE112

if with else-if statement

```

if (condition){
    statement(s)
}
else if (condition){
    statement(s)
}
..
...
else {
}
    
```

- No limit on “else if” statements
- The “else” condition is usually a default condition
 - For completeness

ESE112

8/17

Cascading “if-else”

Example

```

char userChoice;
// Ask user for input and store it in userChoice
if (userChoice == 'q')
    System.out.println("quitting.");
else if (userChoice == 'a')
    System.out.println("adding.");
else if (userChoice == 's')
    System.out.println("saving.");
else
    System.out.println("unrecognized choice.");

//Note: You can avoid the curly brace after
condition if only one statement is to be performed
    
```

ESE112

9/17

Nested if-statements

An if within an if	Truth Table												
<pre> if (condition1){ if (condition2){ statement(s) A } else{ statement(s) B } } else { statements(s) C } </pre>	<p>What values must the conditions have in order for block A to run? B? C?</p> <table border="1"> <thead> <tr> <th></th> <th>A</th> <th>B</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>condition1</td> <td>T</td> <td></td> <td></td> </tr> <tr> <td>condition2</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		A	B	C	condition1	T			condition2			
	A	B	C										
condition1	T												
condition2													

ESE112

10/17

The infamous “dangling else”

Code	Notes
<pre> if (x > y) if (y < z) statementA; else statementB; </pre>	<p>When is statementB executed?</p> <p>In other words, which if is the else paired with?</p>

- An **else** is paired with the last **else-less if**, regardless of spacing, unless { } dictate otherwise.

```

if (x > y){
    if (y < z){
        statementA;
    }
}
else{
    statementB;
}
    
```

ESE112

11/17

A "Loop"

- A simple but powerful mechanism for "making lots of things happen!"
- Performs a statement (or block) over & over
- Usually set up to repeat an action until some condition is satisfied
- Computing Scenarios Examples
 - Run an application until user hits "quit" button
 - Read characters until end of file reached
 - Deal card hands until game over

ESE112

12/17

Syntax of the *while* statement

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

- *condition* is a true/false (boolean) expression
- If *condition* is initially false, the statement is never executed
- If *condition* is true, *statement* is executed and *condition* is re-evaluated
- The *statement* should eventually make the loop stop

ESE112

13/17

A *while* Loop to Print Numbers

```
// Print the numbers 1 thru 10  
int x = 1;  
while (x <= 10){  
    System.out.println(x);  
    x = x + 1;  
}
```

- What happens if you forget the statement $x = x + 1$?
 - We print value 1 forever
 - Known as *infinite* loop

ESE112

14/17

More *Infinite* Loops

```
// Some infinite loops are intentional  
while (true){  
    statement(s)  
}  
  
// Others are not  
int x = 5;  
while (x < 10){  
    statement(s) which don't change x  
}
```

ESE112

15/17

Compute Square of first 10 numbers

```
//In Square.java
int num = 1;
int sqNum = 0;
while (num <= 10) {
    sqNum = num * num;
    System.out.println(num + " " + sqNum);
    num = num + 1;
}
```

ESE112

16/17

For Loop

```
for (init; end-test; re-init){
    statement
}
```

- Executes loop body as long as *end-test* evaluates to TRUE
- Initialization and re-initialization code included in loop statement
- Note: Test is evaluated **before** executing loop body

ESE112

17/17

While vs. For

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

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

Note: *For* loops are used generally for bounded iteration

ESE112

18/17

Summary of Loops

Type of Loop	Syntax
while	<pre>while (condition){ statement(s) }</pre>
for	<pre>for (expr1; condition; expr3){ statement(s) }</pre>

ESE112

19/17