

CSE11 Spring 2015

Lecture 3

Making Choices, Indentation

A program is a sequence of instructions

compute the expression: $(x + 2) * 7$

1. Enter a number at the keyboard
2. Add 2 to that number
3. multiply the result by 7
4. print out the final calculation

A small twist

if x is an even number, compute : $(x + 2) * 7$

if x is an odd number, compute : $(x + 3) * 9$

1. Enter a number (X) at the keyboard
2. if X is even
 1. Add 2 to that number
 2. multiply the result by 7
3. if X is odd
 1. Add 3 to that number
 2. multiply the result by 9
4. print out the final calculation

Twist.java

```
import java.util.Scanner;
public class Twist
{
    public static void main(String[] args)
    {
        Scanner scnr = new Scanner(System.in);
        System.out.println("Enter your X value:");
        int xval, result;
        xval=scnr.nextInt();
        if ( (xval % 2) == 0 )
            result = (xval + 2) * 7;
        else
            result = (xval + 3) * 9;
        System.out.println("result is " + result);
    }
}
```

What is a main method?

How is the input read from the user?

How is it determined if the input is even?
Will that work if xval is of type float?

How is data output?

Where will it be printed?

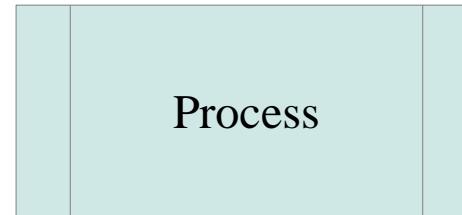
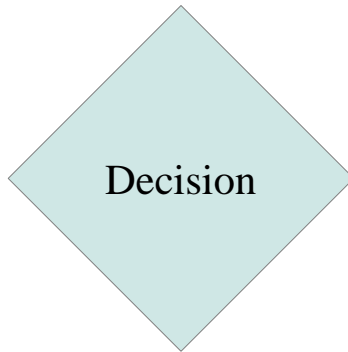
Making Choices

- Computer programs have to respond to “conditionals”
- `if` (the sky is blue) `then` play outside
- `if` (I am hungry) `then` eat dinner
- Must also be able to say what happens if the conditional is not true
- `else` (what you do when the condition is not met)

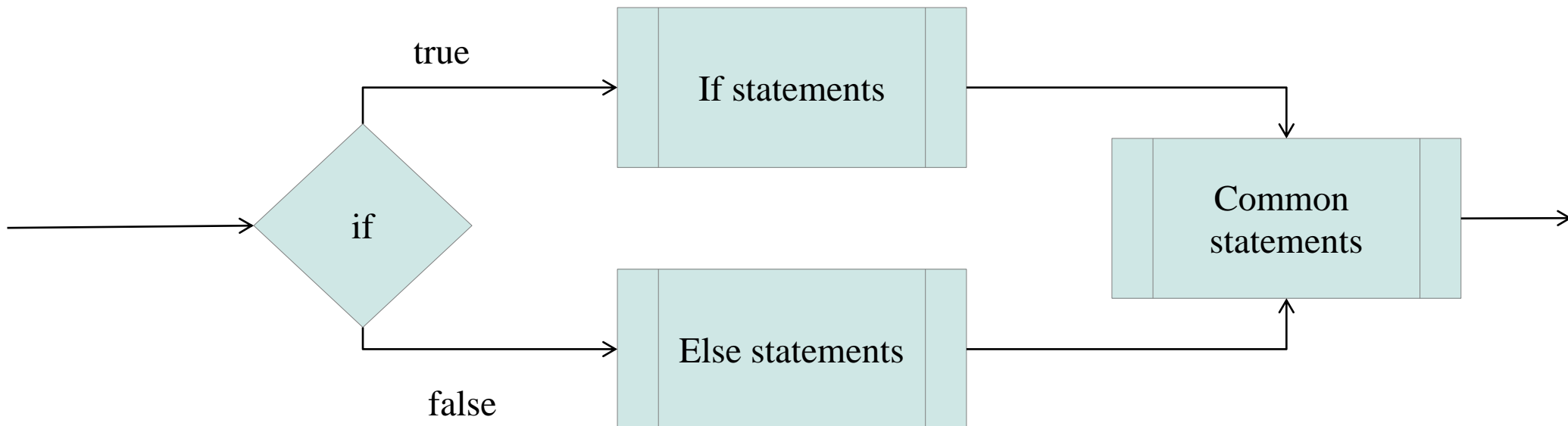
The if statement

```
if ( condition ) {  
    if-part; // when condition is true  
}  
else {  
    else-part // when condition is not true  
}
```

Flow-chart 101



This is called **Branching**
Code follows one branch or the other



(condition)

- The condition is a logical expression.
- To create a general logical expression, we need
 - Comparison operators
 - E.g. less than, greater than, equal to, not equal to
 - Logical operators
 - OR, AND, ...

Java Comparison Operators

- $A < B$ is A *less than* B?
- $A > B$ is A *greater than* B?
- $A \leq B$ is A *less than or Equal* to B?
- $A \geq B$ is A *greater than or Equal* to B?
- $A == B$ is A *Equal* to B?
- $A != B$ is A *not Equal* to B?



“=” is the assignment operator

“==” is the equality comparison operator

It's very easy to confuse/misread these two

Each conditional operator evaluates to true or false

- There is no “maybe” in conditional operators
- The type is called “Boolean” (named after the 19th century Mathematician, George Boole)
- The boolean data type has only two possible values
 - true
 - false
- One can also declare a variable to be of type `boolean`

Equivalent code using a Boolean variable


```
boolean theSame;  
int A;  
int B;  
  
if (A == B) {  
    System.out.println("Equals")  
}
```

```
boolean theSame;  
int A;  
int B;  
  
theSame = A == B;  
if (theSame) {  
    System.out.println("Equals")  
}
```

Why use Boolean variables?

- Sometimes the conditions you want to test for are “complicated”.
- A suitably named Boolean variable will *describe* in English the condition you want to meet
- Populating the condition (boolean) variable becomes a separate thought from using it.
 - Sometimes you use the same condition several times in your code for different branches. Evaluate the condition once, then use it in several `if` statements.

Logical Operators

operand1  operand2

Operator	Function	result with at least one float operand
&&	logical and	true if both operands are true
	logical or	true if either (or both) operands are true
^	exclusive or	true if and only if one operand is true
! <operand>	unary negation	true if operand is false

Truth Tables

		B	
A	OR	T	F
	T	T	T
	F	T	F

A || B

		B	
A	AND	T	F
	T	T	F
	F	F	F

A && B

		B	
A	XOR	T	F
	T	F	T
	F	T	F

A ^ B

Logical Statement Equivalences – DeMorgan's Laws

- $\neg (A \ \&\& \ B) \iff \neg A \ || \ \neg B$
 - I am not (sick and tired) \rightarrow I am either not sick OR not tired
- $\neg (A \ || \ B) \iff \neg A \ \&\& \ \neg B$
 - I am not (sick or tired) \rightarrow I am not sick AND I am not tired

		B	
		T	F
A	T	F	T
	F	T	T
		$\neg A \ \ \neg B$	

		B	
		T	F
A	T	F	T
	F	T	T
		$\neg (A \ \&\& \ B)$	

Java Expressions

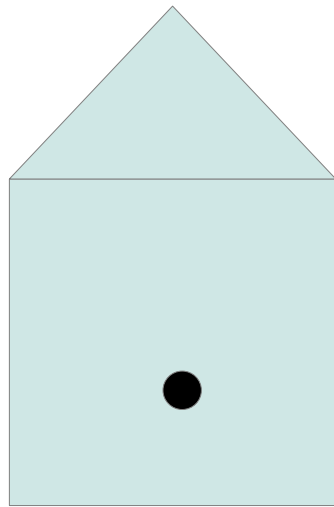
- A java expression is a chunk of code the can be evaluated to be a single object (haven't formally introduced objects, yet)
- Many common expressions evaluate to numbers or boolean values
- Think of an expression as a 'function' that when evaluated “returns” are particular value

$x > y \parallel z < y \parallel x == z$



Expression (evaluates to a boolean)

Using a Boolean Variable



triangle

square

house

Point is “in the house” if it is either in the square and/or in the triangle

```
Location point;  
boolean inside;  
  
inside = triangle.contains(point) || square.contains(point);  
  
if (inside) {  
    System.out.println("We are warm inside!");  
}  
else {  
    System.out.println("It's cold out here!");  
}
```

Intuition: Objects and Methods

From previous slide:

```
triangle.contains(point) || square.contains(point);
```

What is “triangle”? What is “square”? What is “point”?

What do you think `triangle.contains(point)` does?

What *type* does the `contains` method return?

Short-Circuit Evaluation

- Java scans conditional expressions from left to right.
- If it can determine the correct boolean value before reaching the end of the expression, it stops evaluating
- This is called “Short-Circuit”
- `&&` and `||` are the only operators that can be short-circuited

Short-Circuit Examples

- If (conditionA && conditionB)
 - IF ConditionA is FALSE, then the && must be false. ConditionB is never evaluated
- If (conditionA || conditionB)
 - IF ConditionA is TRUE, then the || must be true. ConditionB is never evaluated
- Why important? Sometimes you will see program assignment inside of boolean expressions
 - `A = (C || (i=j) != 10)`
 - IF C is TRUE, then the i=j assignment is never performed

Java Statements/ Statement Blocks

- A java statement has a semicolon “;” at the end of it

```
A=25;  
Box.moveTo(30,80);
```

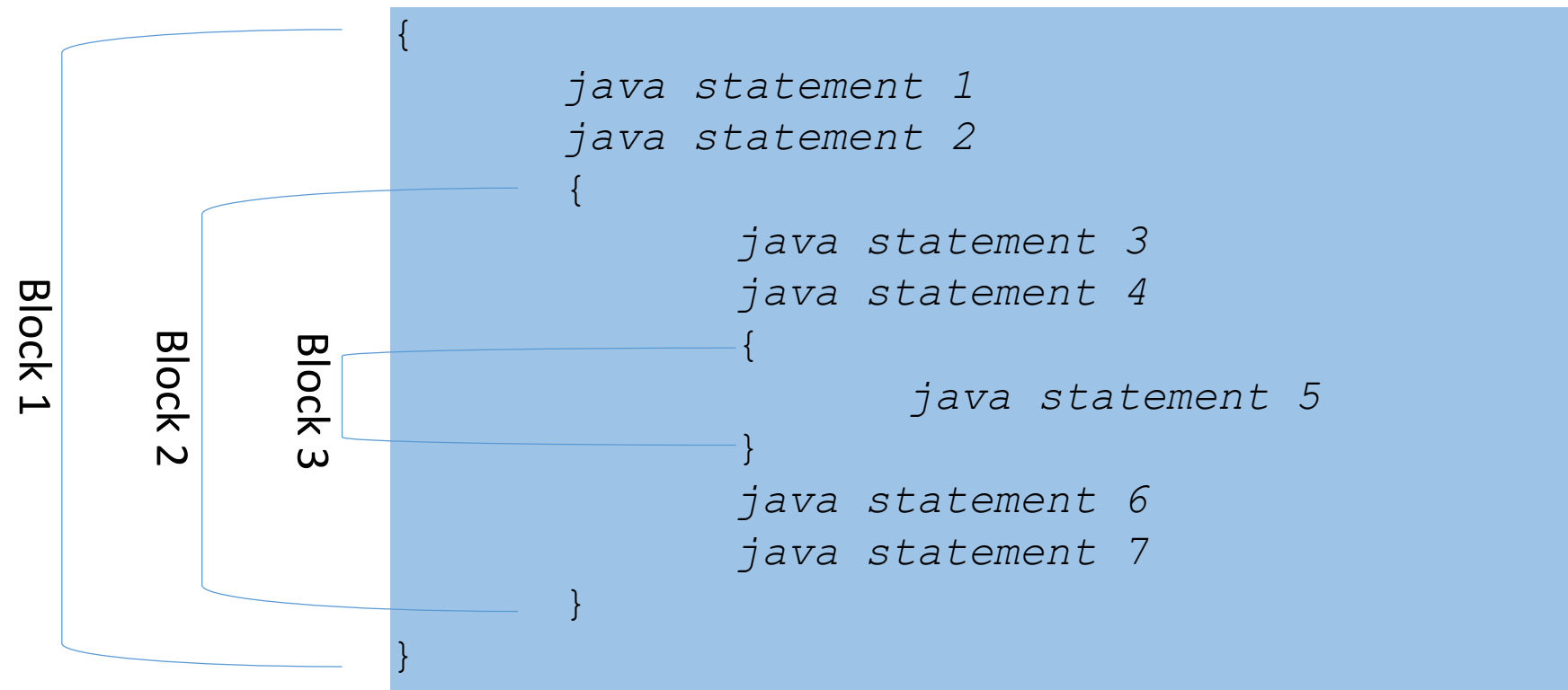
- A java statement block has an opening '{' and closing '}' with zero or more statements in it

```
{  
    A=25;  
    Box.moveTo(30,80);  
}  
{ /* empty block */ }
```

{ . . . }

The “{” and the “}” indicate the beginning and end of a statement block.

- Zero or more java statements are contained within a statement block
- Statement blocks can be nested



The “if” construction syntax revisited

if (boolean expression) *statement*

OR

if (boolean expression) *statement-block*

In English: This is the *syntax* of an if “statement”.

The **keyword** is **if**. Followed by parenthesis that must contain a boolean expression. Followed by a java statement OR a java statement block.

(Note, this doesn't include optional **else** or **else if ...**)

if-else if-else construction syntax

```
if ( boolean expression )  
    statement-block  
else if (boolean expression)  
    statement-block  
....  
else if (boolean expression)  
    statement-block  
else  
    statement-block
```



Be very careful with this construction. It can be confusing which “else” block is “bound” to which if statement.

Java String Type

- Not a primitive type. => Object-type
- String Literals
 - “this is a string”, “so is this”, “and”, “t”, “his”
- Numerous common operations on Strings
 - How long is it? (how many characters in it)
 - Is it empty (length is zero)
 - Put two strings together to make a new string (concatenation)
 - Remove one or more characters from the string.
 - Compare if two strings are identical.
 -
- <http://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

Indentation

Indentation is extremely important for code readability.

(Tutors are instructed to **NOT** assist you if your code is not properly indented)



```
import java.util.Scanner;
public class Twist
{
    public static void main(String[] args)
    {
        Scanner scnr = new Scanner(System.in);
        System.out.println("Enter your X value:");
        int xval, result;
        xval=scnr.nextInt();
        if ( (xval % 2) == 0 )
            result = (xval + 2) * 7;
        else
            result = (xval + 3) * 9;
        System.out.println("result is " + result);
    }
}
```

Statement Blocks and Indentation

There are two types of people.

```
if (Condition)
{
    Statements
    /*
     ...
     */
}
```

```
if (Condition) {
    Statements
    /*
     ...
     */
}
```

Programmers will know.

Some VIM terms

- I recommend that you indent with TABS
 - Other may recommend spaces.
 - Be Consistent in your own code
- * Some Settings that control appearance in VIM and autoindent
 - tabstop = # of spaces for the TAB character (def: 8)
 - shiftwidth = affects “>>”, “<<“, “==“ commands
 - expandtabs = determine if tabs are converted to spaces
- * Command to re-indent an entire file
 - in command mode “gg=G”
- * vim comment line at end of file
 - // vim:ts=4:sw=4:tw=78: