Lecture # 7

Else if
and
Logical Operators

Introduction to Programming

`else if`

- So far we have written programs with selection between two alternatives.
- If this is true, or in any other case
- Programs often require more than two alternatives
- For example, in an exam you might score an A, or a B, or a C……
- `else if` allows us to factor in several alternatives

Sample Program 1

```
/* Program to illustrate the use of if else statements */
   import java.util.*;
   public class PosNegTester1
   {
      public static void main(String[] args)
      {
         Scanner keyboardIn = new Scanner(System.in);

         int number;

         System.out.print("Please enter an integer value:  ") ;
         number = keyboardIn.nextInt();

         if ( number > 0 )
         {
            System.out.print(number + " is a positive value ") ;
         }
          else
         {
            System.out.print(number + " is a negative value ") ;
         }
      }
   }
```

What is the output if `number` is 0?

Sample Program 2

```
/* Program to illustrate the use of else/if statements */


    if(number > 0 )
    {
          System.out.print(number + " is a positive value ") ;
    }
    else if (number < 0 )
    {
          System.out.print(number + " is a negative value ") ;
    }
    else
    {
           System.out.print("Number entered is zero") ;
    }
```

Order of evaluation

- The expressions are evaluated in order: if any expression is true, the statement associated with `if` is executed and it terminates the chain.

- The last `else` handles none of the above.

```
if (expression)
{
        statements;
}
else if (expression)
{
        statements;
}
else if (expression)
{
        statements;
}
else
{
        statements;
}
```

- The *conditional expressions* are evaluated from the top down

- Once a *true* condition is found, the statement associated with it is executed and the rest of the ladder is bypassed

- If none of the conditions is true, the final `else` statement will be executed

- If there is no final `else` and all other conditions are false, then no action will take place

- The final `else` is optional

```java
int mark;
char grade;
System.out.print( "Please enter an exam mark ") ;
mark = keyboardIn.nextInt();

if ( mark >= 70 )
{
        grade = 'A' ;
}
else if ( mark >= 60 )
{
        grade = 'B' ;
}
else if ( mark >= 50 )
{
        grade = 'C' ;
}
else if ( mark >= 40 )
{
        grade = 'D' ;
}
else
{
        grade = 'F' ;
}
System.out.print( "Mark:  " + mark + "  grade: " + grade);
```

Order matters…

```
if ( mark >= 0 )
{
   grade = 'F' ;
}
else if ( mark >= 30 )
{
   grade = 'E' ;
}
else if ( mark >= 40 )
{
   grade = 'D' ;
}
…
```

Alternative comparisons

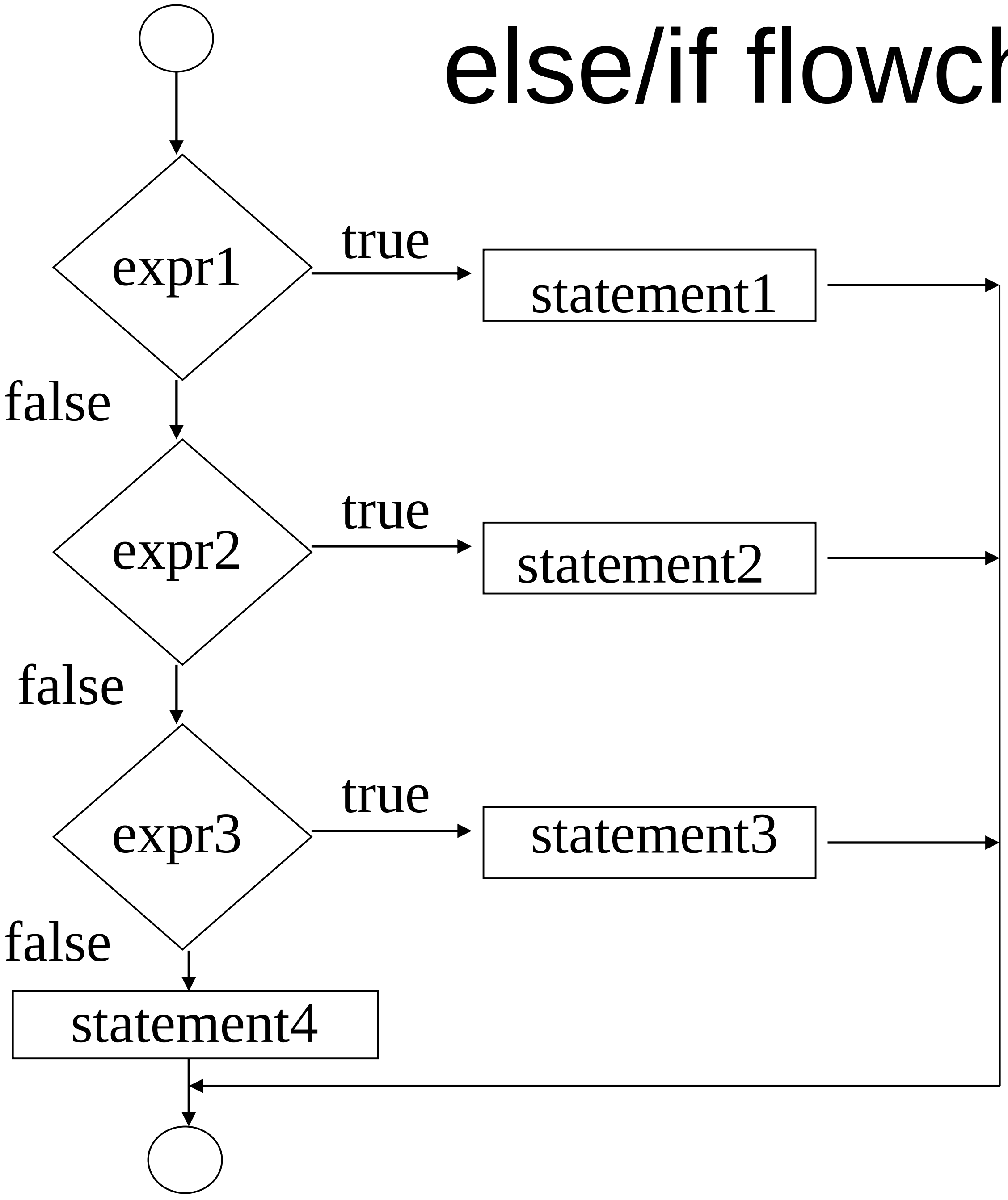
```
if ( mark < 30 )
{
   grade = 'F' ;
}
else if ( mark < 40 )
{
   grade = 'E' ;
}
…
```

Or…

```
if ( mark >= 70 )
{
      grade = 'A' ;
}
else if ( mark >= 60 )
{
      grade = 'B' ;
}
…
```

Order of evaluation is important!

else/if flowchart



```
if(expr1)
    statement1;
else if(expr2)
    statement2;
else if(expr3)
    statement3;
else
    statement4;
```

Logical operators

- Conditions often have more than one criteria.

- Logical operators are used to allow more than one criteria in a conditional statement.

- Example:
  - Cinema ticket system
  - Teenagers pay less than full price
  - This condition has more than one criteria (age must be more than 12 AND age must be less than 20)

Logical Operators

Logical Operators	**Tests for**	**Example**
&&	AND	Condition A && Condition B Return true if both condition A and B are true, otherwise return false
||	OR	Condition A || Condition B Return true if either Condition A or B is true for if both Condition A and B are true
!	NOT	!A If Condition A is true, result of this operation is false If Condition A is false, result of this operation is true

Example - `&&`

```
if(age > 12 && age < 20)
{
  System.out.println("You are a teenager");
}
```

- Here both conditions must be true for the message to be printed on screen
- If either condition is false, the message will not be printed.

Example - ||

```
if(age < 12 || age > 65)
{
 System.out.println("Admission is free");
}
```

- Here a minimum of one condition must be true for the message to be printed on screen
- If both conditions are false, the message will not be printed.

Truth Table for logical operators

Operands		***Results***		
p	***q***	***p && q***	***p || q***	***!p***
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

Where *p* and *q* are expressions that evaluate to TRUE or FALSE

Order of evaluation

- When Java sees a `&&` operator or a `||`, the expression on the left side of the operator is evaluated first.

```
int num1 = -2;
int num2 = 4;
if(num1 > 0 && num2 <100)
        …
```

- In this example, Java first checks to see if `num1 > 0` is true. Here this is false, so the entire condition must be false regardless of whether `num2 <100` is true or not, so Java doesn't bother checking `num2 <100`.

Order of evaluation

```
int num1 = 2;
int num2 = 4;
if(num1 > 0 || num2 <100)
...
```

- Again, Java first checks to see if `num1 > 0` is true. Here this is true, so the entire condition must be true regardless of whether `num2 <100` is true or not, so Java doesn't bother checking `num2 <100`.
- This is known as short-circuit evaluation.

True or False?

```
int num1, num2;
num1 = 15;
num2 = 20;

num1 > 16 && num1<25
num1 < 1 && num1 < 25
num2>=20 || num2<=25
num2>=20 && num2<=25
num1 != num2
((num2>=20) && (num2 <=25))
```

Precedence and Associativity

Operators (in order of precedence)	Associativity
( )	Left to right
-(unary) ++ -- !	Right to left
* / %	Left to right
+ -	Left to right
< <= > >=	Left to right
= = !=	Left to right
&&	Left to right
||	Left to right
?:	Right to Left
= += -= *= /= %=	Right to left