# CS 152
## Computer Programming Fundamentals
## *The* `if-else` *Statement*

Instructor:
Joel Castellanos
**e-mail:** joel@unm.edu
**Web:** http://cs.unm.edu/~joel/

**Office:** Electrical and Computer Engineering building (ECE).
Room 233

## Quiz:

Which is the Correct Formatting in *CS-152*?

a)

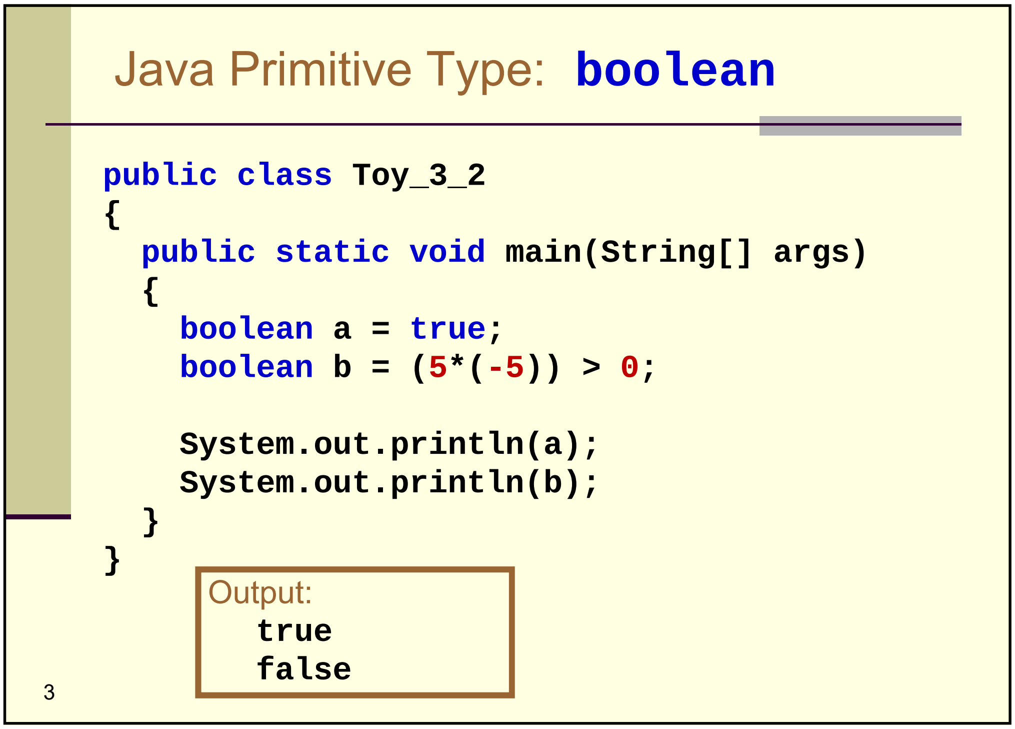
```
public class Tmp
{
  public static void main(String[] args)
  {
    System.out.println("Pick Me");
  }
}
```

b)

```
public class Tmp
  {
  public static void main(String[] args)
    {
  System.out.println("No, Me");
    }
  }
```

c)

```
public class Tmp
  {
  public static void main(String[] args)
  {
    System.out.println("Ooh, Ooh");
    }
  }
```

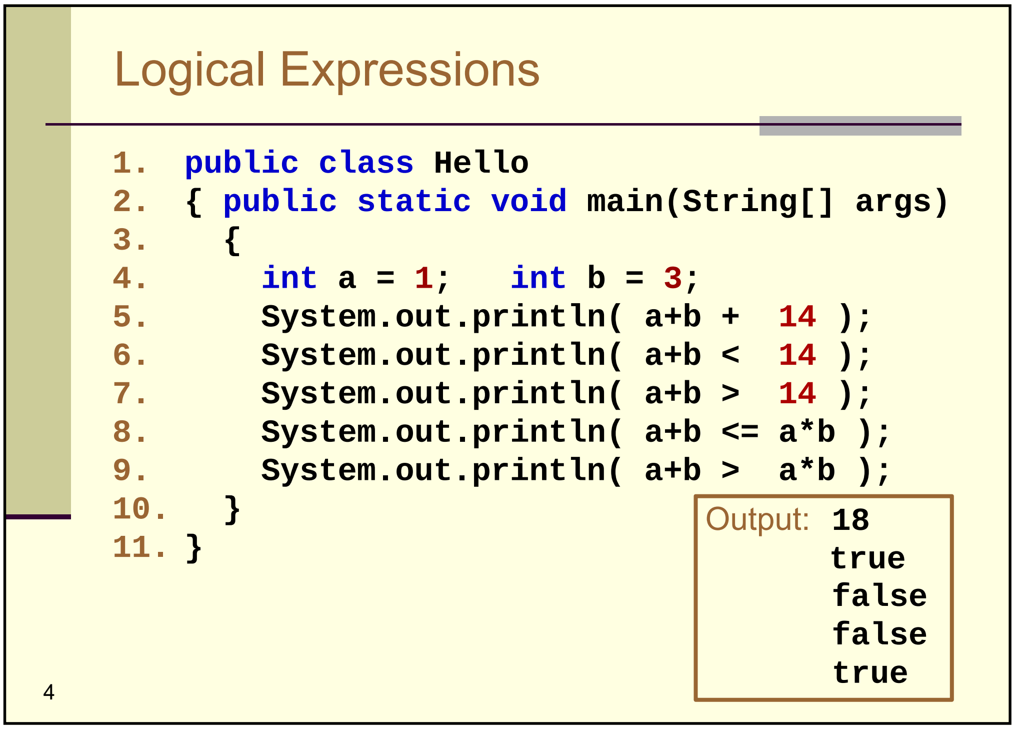


## Java Primitive Type: boolean

```
public class Toy_3_2
{
  public static void main(String[] args)
  {
    boolean a = true;
    boolean b = (5*(-5)) > 0;

    System.out.println(a);
    System.out.println(b);
  }
}
```

Output:

```
true
false
```

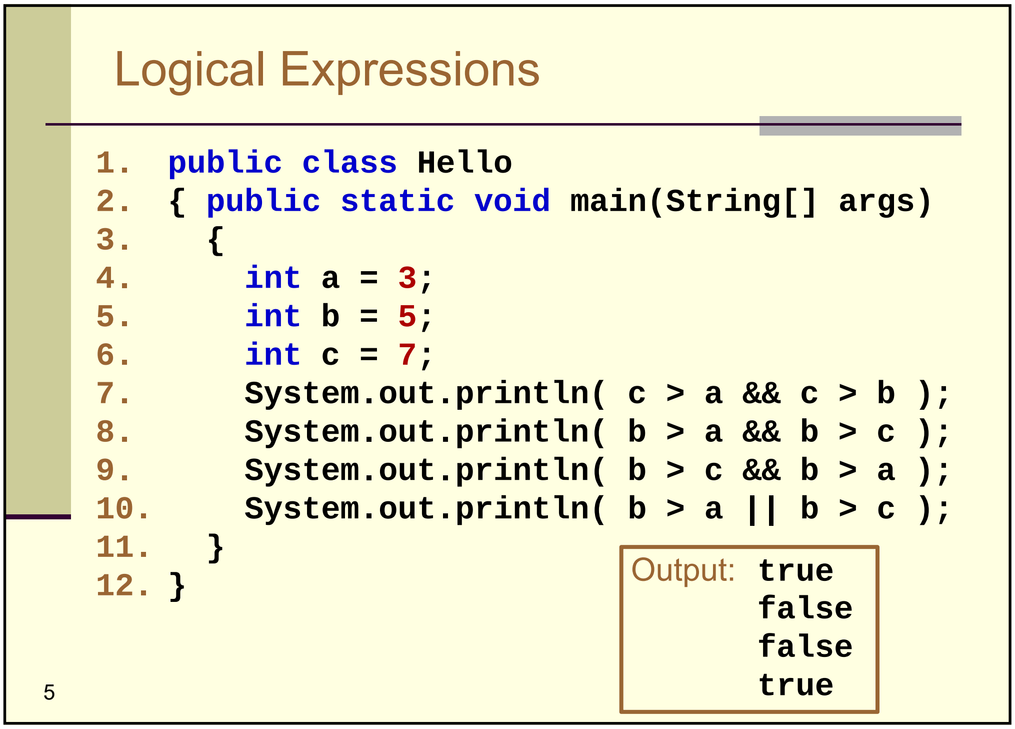


## Logical Expressions

```
public class Hello
{ public static void main(String[] args)
  {
    int a = 1;   int b = 3;
    System.out.println( a+b +  14 );
    System.out.println( a+b <  14 );
    System.out.println( a+b >  14 );
    System.out.println( a+b <= a*b );
    System.out.println( a+b >  a*b );
  }
}
```

Output:

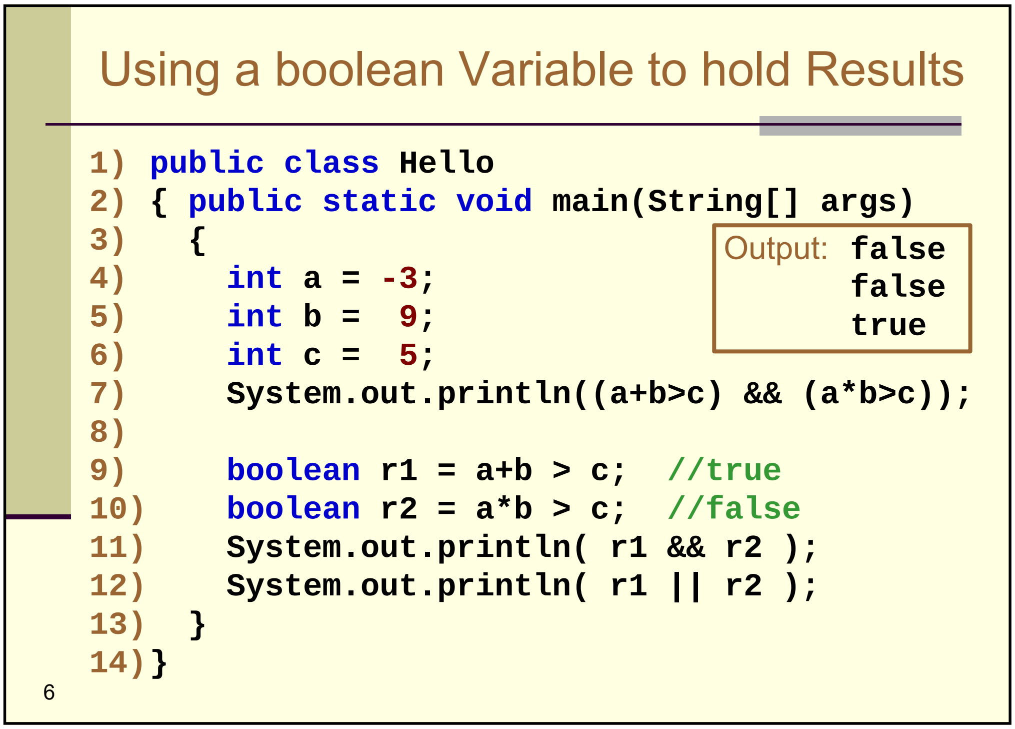
```
18
true
false
false
true
```



## Logical Expressions

```java
public class Hello
{ public static void main(String[] args)
  {
    int a = 3;
    int b = 5;
    int c = 7;
    System.out.println( c > a && c > b );
    System.out.println( b > a && b > c );
    System.out.println( b > c && b > a );
    System.out.println( b > a || b > c );
  }
}
```

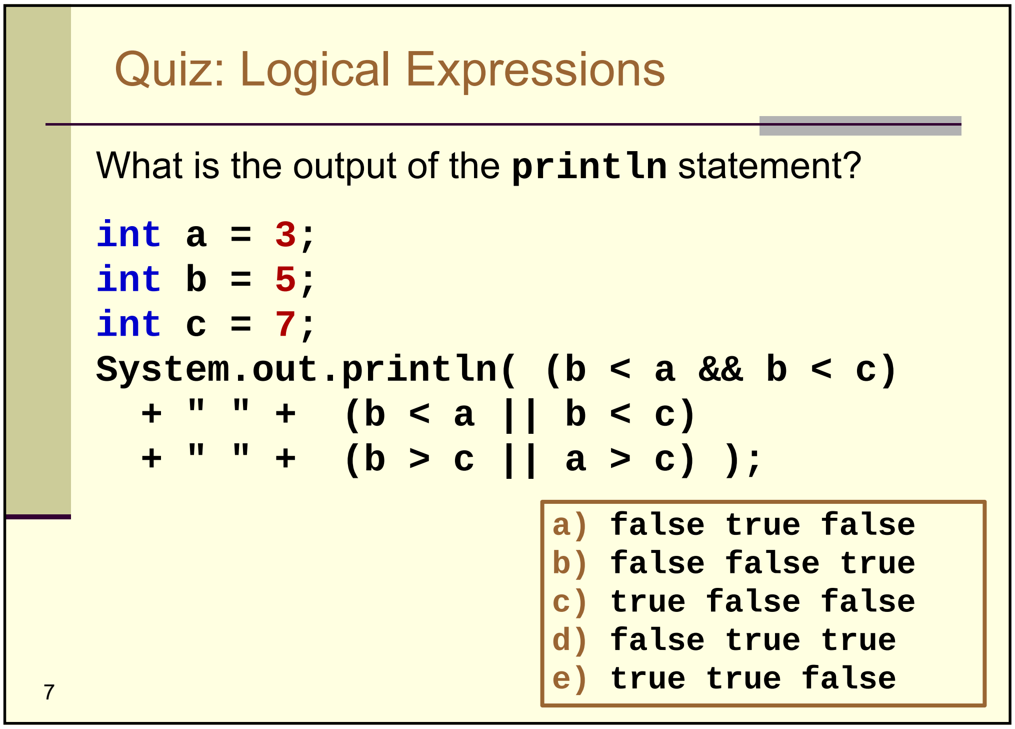
Output:
```
true
false
false
true
```



## Using a boolean Variable to hold Results

```java
public class Hello
{ public static void main(String[] args)
  {
    int a = -3;
    int b =  9;
    int c =  5;
    System.out.println((a+b>c) && (a*b>c));

    boolean r1 = a+b > c;  //true
    boolean r2 = a*b > c;  //false
    System.out.println( r1 && r2 );
    System.out.println( r1 || r2 );
  }
}
```

Output:
```
false
false
true
```



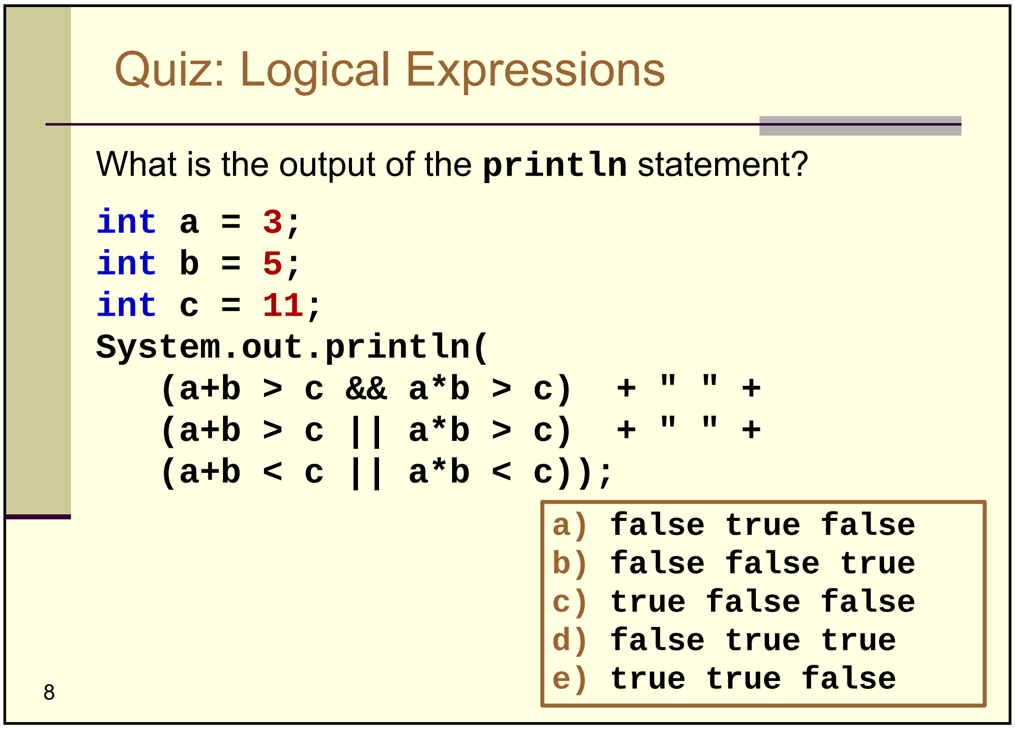
## Quiz: Logical Expressions

What is the output of the **println** statement?

```
int a = 3;
int b = 5;
int c = 7;
System.out.println( (b < a && b < c)
  + " " +  (b < a || b < c)
  + " " +  (b > c || a > c) );
```

```
a) false true false
b) false false true
c) true false false
d) false true true
e) true true false
```

7



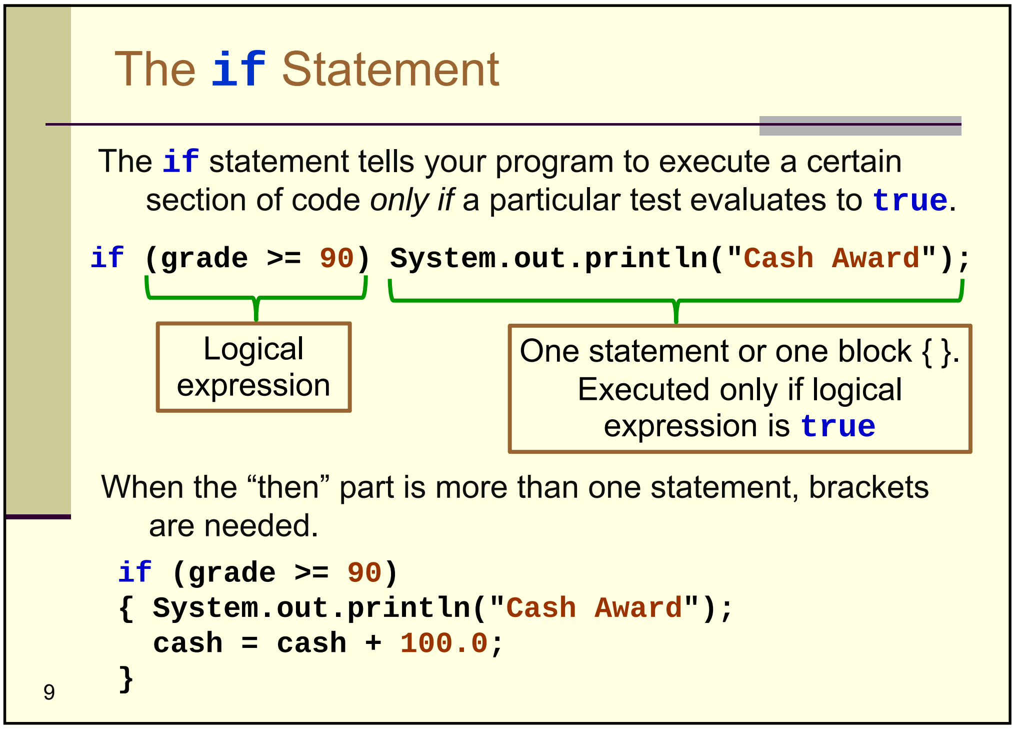
## Quiz: Logical Expressions

What is the output of the **println** statement?

```
int a = 3;
int b = 5;
int c = 11;
System.out.println(
   (a+b > c && a*b > c)  + " " +
   (a+b > c || a*b > c)  + " " +
   (a+b < c || a*b < c));
```

```
a) false true false
b) false false true
c) true false false
d) false true true
e) true true false
```

8



# The if Statement

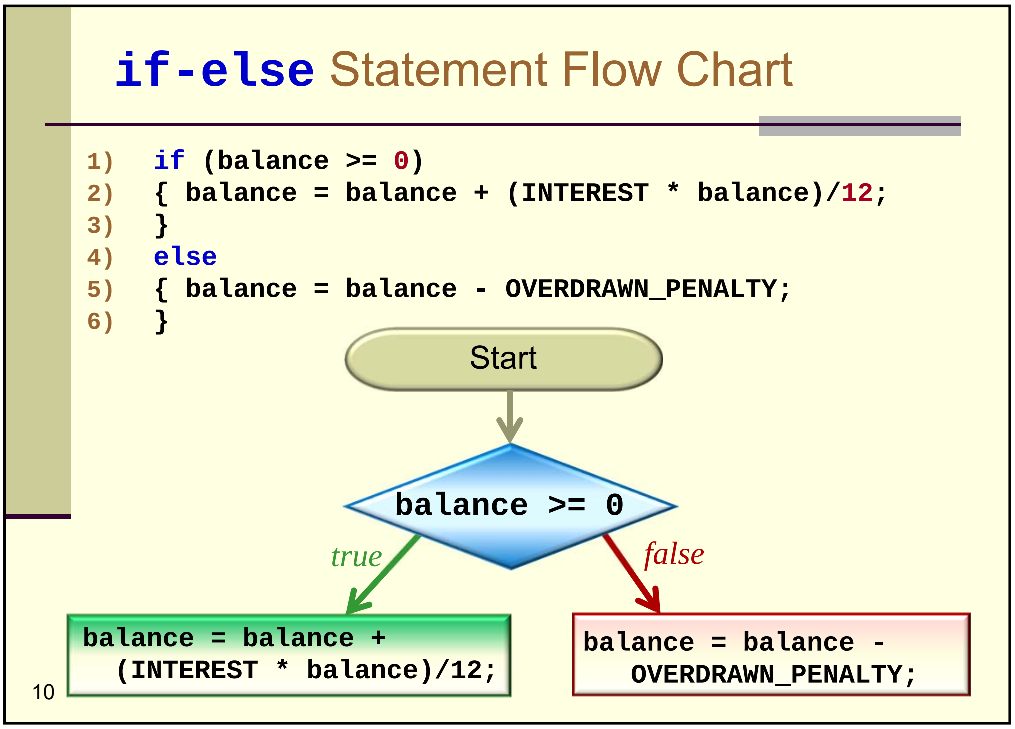
The if statement tells your program to execute a certain section of code *only if* a particular test evaluates to true.

```
if (grade >= 90) System.out.println("Cash Award");
```

Logical expression
One statement or one block { }. Executed only if logical expression is true

When the “then” part is more than one statement, brackets are needed.

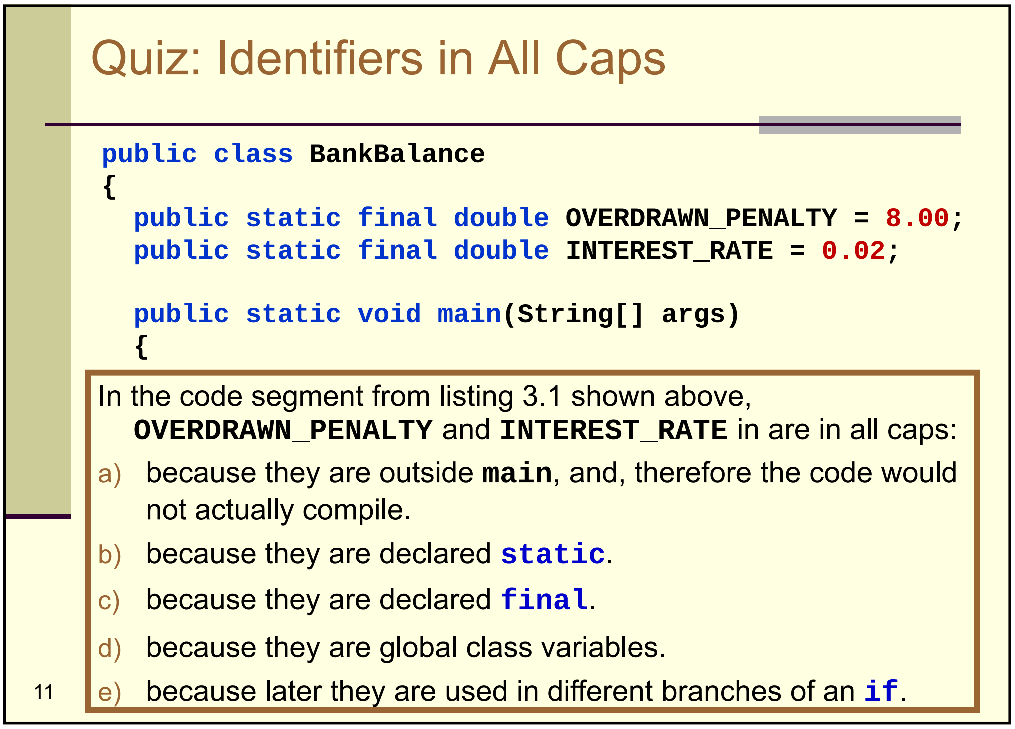
```
if (grade >= 90)
{ System.out.println("Cash Award");
  cash = cash + 100.0;
}
```



# if-else Statement Flow Chart

```
if (balance >= 0)
{ balance = balance + (INTEREST * balance)/12;
}
else
{ balance = balance - OVERDRAWN_PENALTY;
}
```

Start
balance >= 0
true
false
balance = balance + (INTEREST * balance)/12;
balance = balance - OVERDRAWN_PENALTY;



## Quiz: Identifiers in All Caps

```
public class BankBalance
{
  public static final double OVERDRAWN_PENALTY = 8.00;
  public static final double INTEREST_RATE = 0.02;

  public static void main(String[] args)
  {
```

In the code segment from listing 3.1 shown above, **OVERDRAWN_PENALTY** and **INTEREST_RATE** in are in all caps:

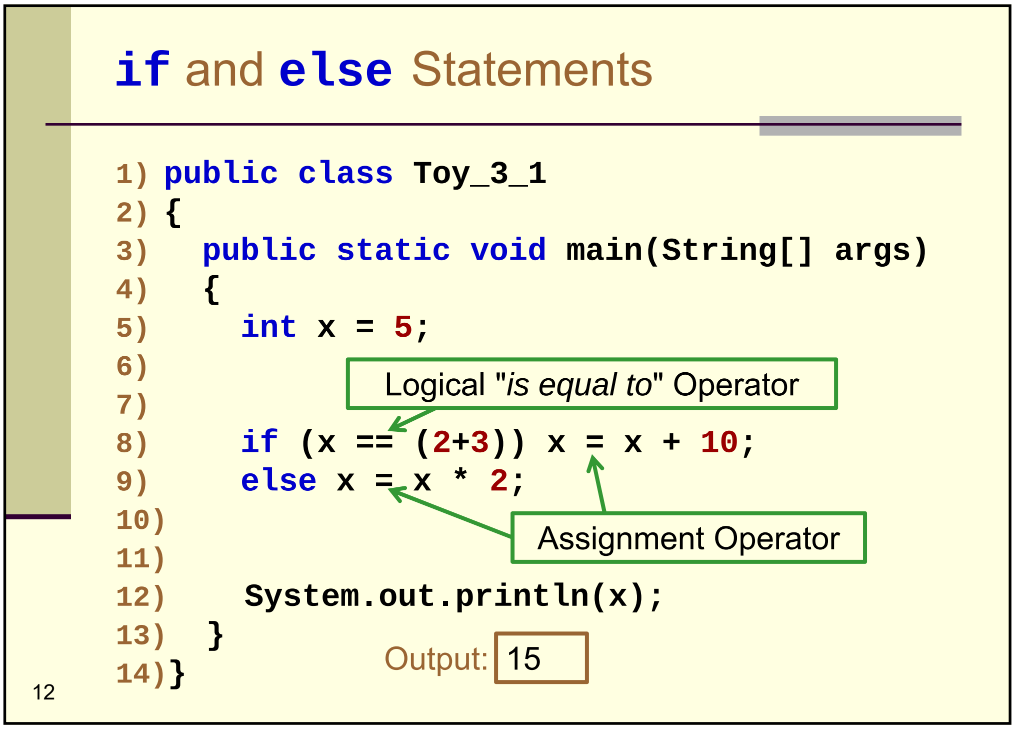
a) because they are outside **main**, and, therefore the code would not actually compile.

b) because they are declared **static**.

c) because they are declared **final**.

d) because they are global class variables.

e) because later they are used in different branches of an **if**.



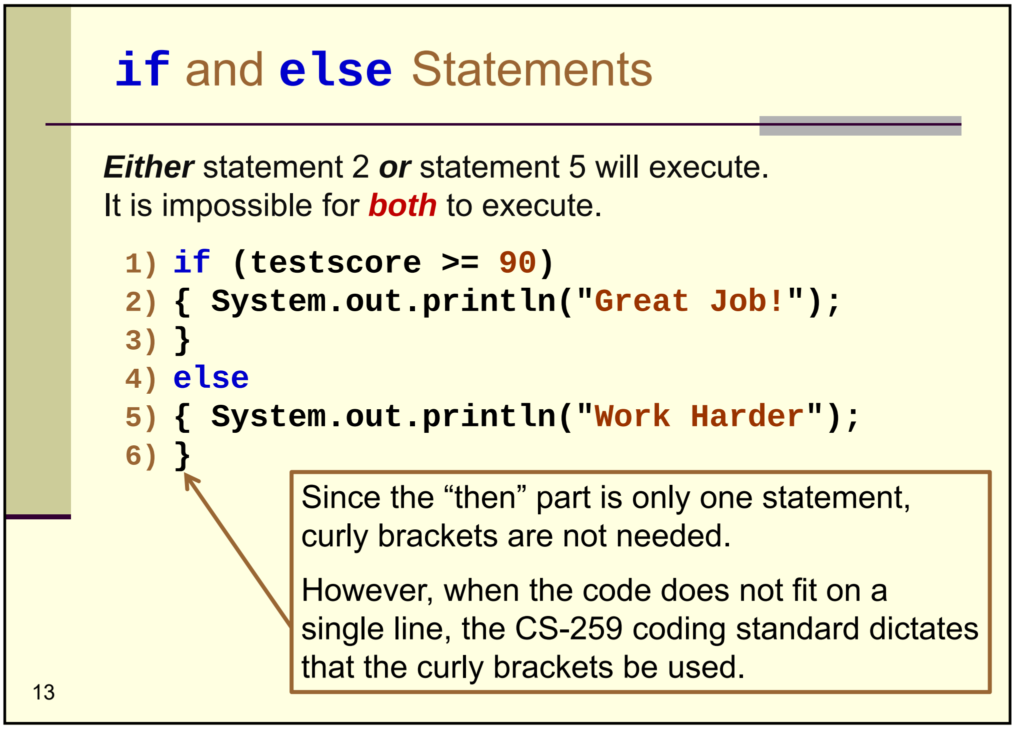
## if and else Statements

```
public class Toy_3_1
{
  public static void main(String[] args)
  {
    int x = 5;


    if (x == (2+3)) x = x + 10;
    else x = x * 2;


    System.out.println(x);
  }
}
```

Logical "*is equal to*" Operator

Assignment Operator

Output: 15



## if and else Statements

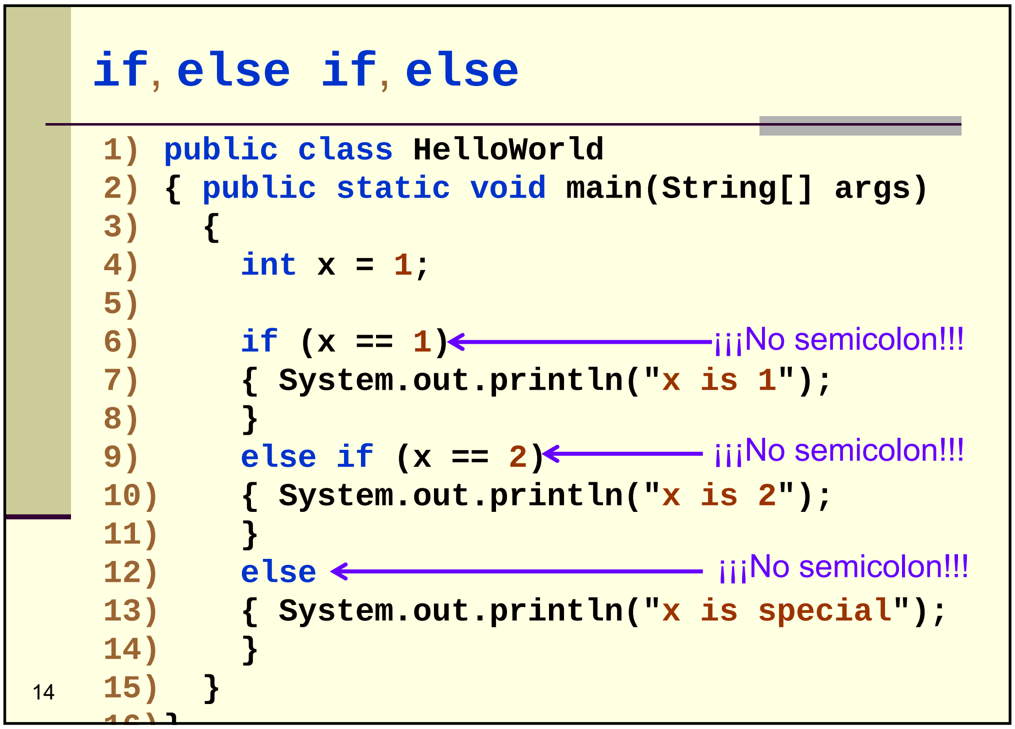
***Either*** statement 2 ***or*** statement 5 will execute.
It is impossible for ***both*** to execute.

```
1) if (testscore >= 90)
2) { System.out.println("Great Job!");
3) }
4) else
5) { System.out.println("Work Harder");
6) }
```

Since the “then” part is only one statement, curly brackets are not needed.

However, when the code does not fit on a single line, the CS-259 coding standard dictates that the curly brackets be used.

13



## if, else if, else

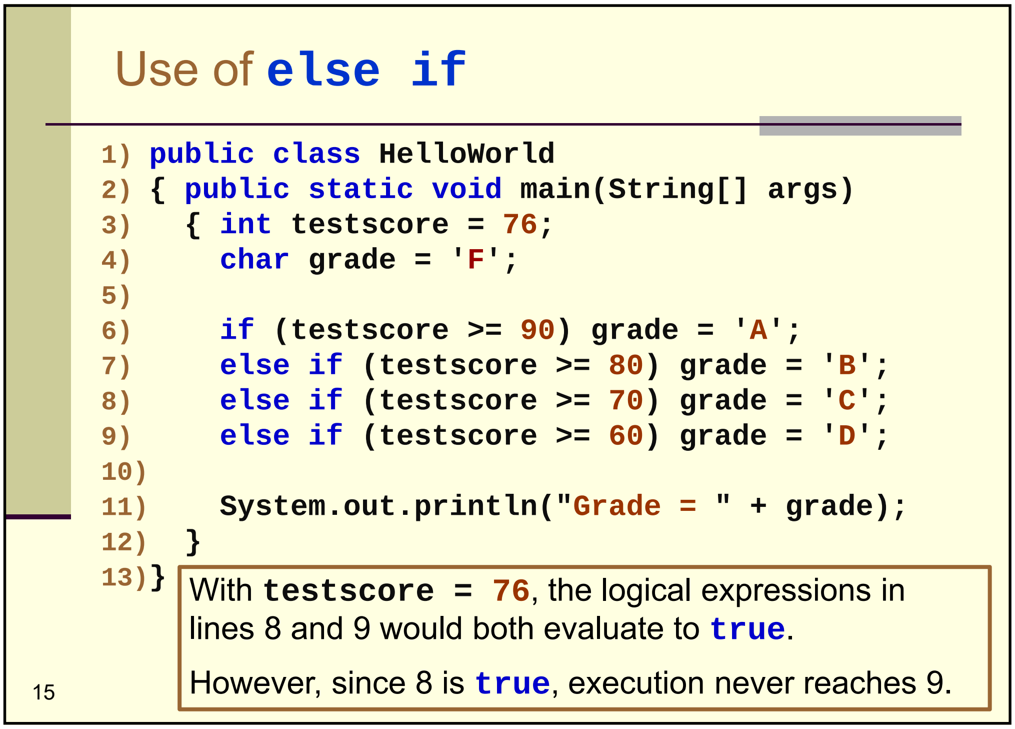
```
public class HelloWorld
{ public static void main(String[] args)
  {
    int x = 1;

    if (x == 1)
    { System.out.println("x is 1");
    }
    else if (x == 2)
    { System.out.println("x is 2");
    }
    else
    { System.out.println("x is special");
    }
  }
}
```

¡¡¡No semicolon!!!

¡¡¡No semicolon!!!

¡¡¡No semicolon!!!

14



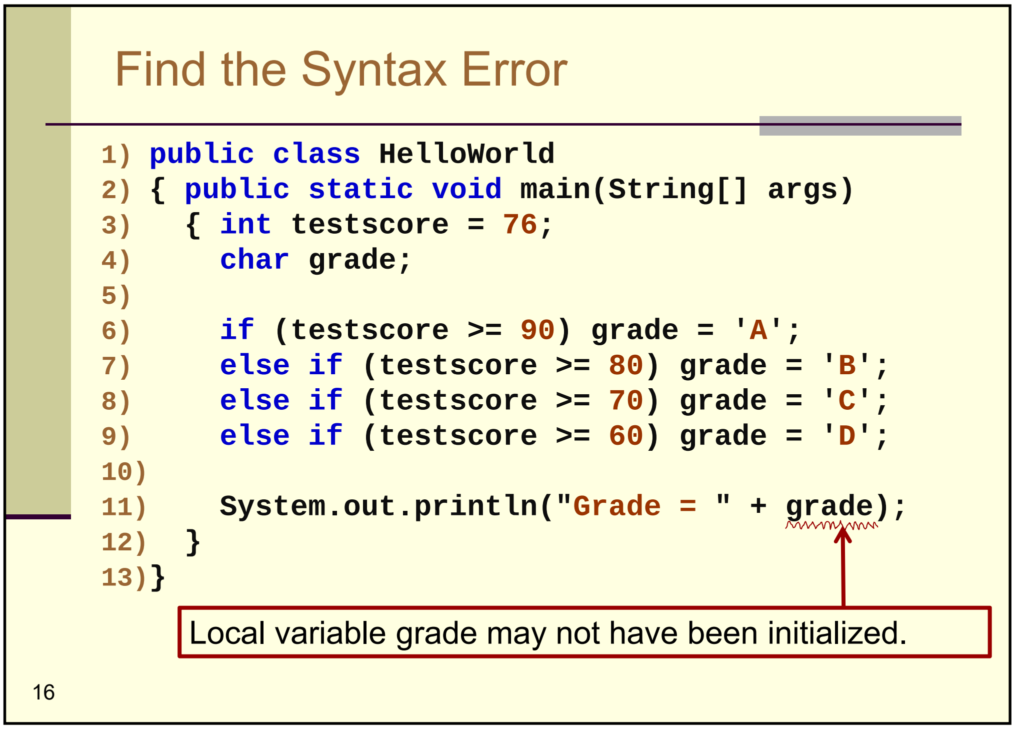
## Use of `else if`

```
1) public class HelloWorld
2) { public static void main(String[] args)
3)   { int testscore = 76;
4)     char grade = 'F';
5)
6)     if (testscore >= 90) grade = 'A';
7)     else if (testscore >= 80) grade = 'B';
8)     else if (testscore >= 70) grade = 'C';
9)     else if (testscore >= 60) grade = 'D';
10)
11)    System.out.println("Grade = " + grade);
12)  }
13)}
```

With `testscore = 76`, the logical expressions in lines 8 and 9 would both evaluate to `true`.

However, since 8 is `true`, execution never reaches 9.

15

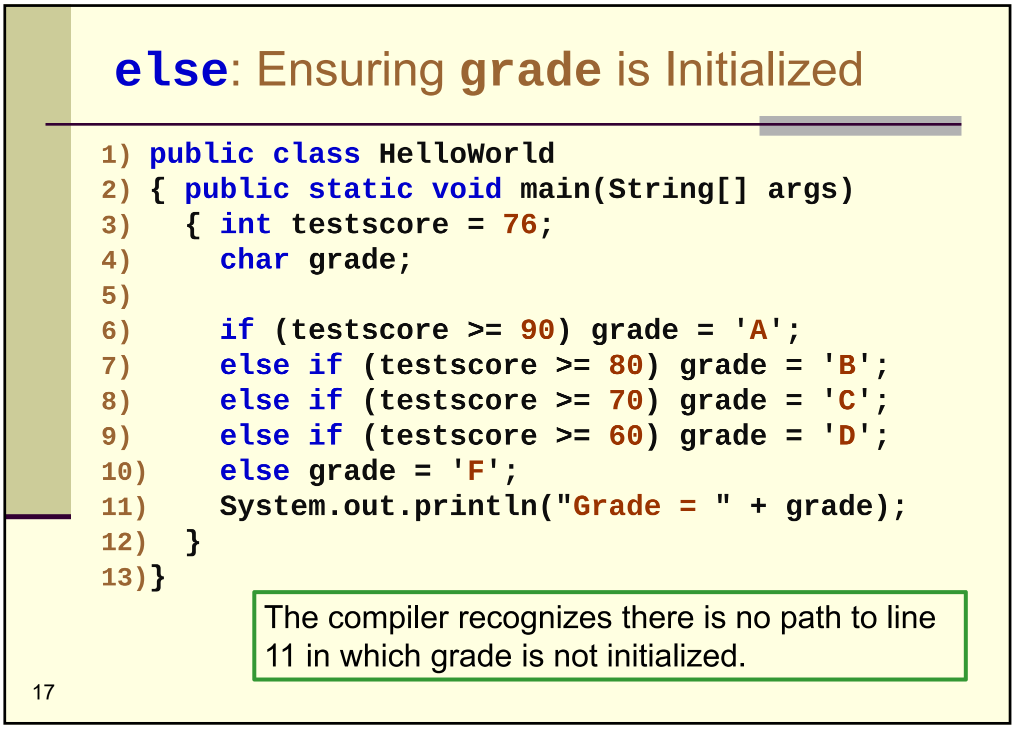


## Find the Syntax Error

```
1) public class HelloWorld
2) { public static void main(String[] args)
3)   { int testscore = 76;
4)     char grade;
5)
6)     if (testscore >= 90) grade = 'A';
7)     else if (testscore >= 80) grade = 'B';
8)     else if (testscore >= 70) grade = 'C';
9)     else if (testscore >= 60) grade = 'D';
10)
11)    System.out.println("Grade = " + grade);
12)  }
13)}
```

Local variable grade may not have been initialized.

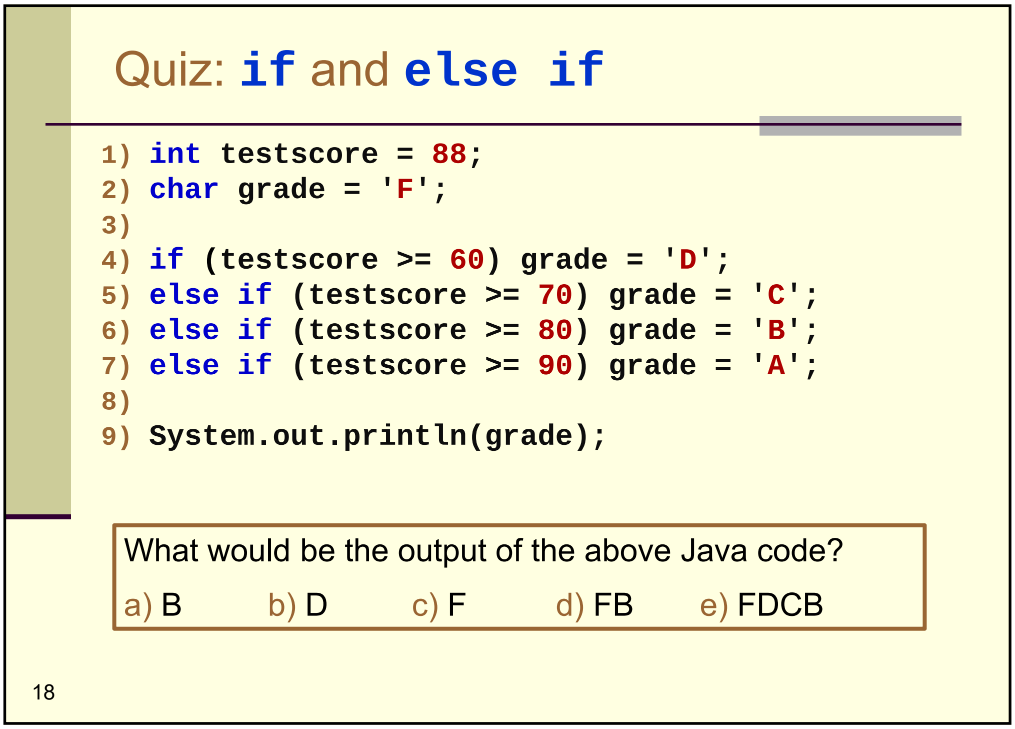
16



## else: Ensuring grade is Initialized

```
1) public class HelloWorld
2) { public static void main(String[] args)
3)   { int testscore = 76;
4)     char grade;
5)
6)     if (testscore >= 90) grade = 'A';
7)     else if (testscore >= 80) grade = 'B';
8)     else if (testscore >= 70) grade = 'C';
9)     else if (testscore >= 60) grade = 'D';
10)    else grade = 'F';
11)    System.out.println("Grade = " + grade);
12)  }
13)}
```

The compiler recognizes there is no path to line 11 in which grade is not initialized.

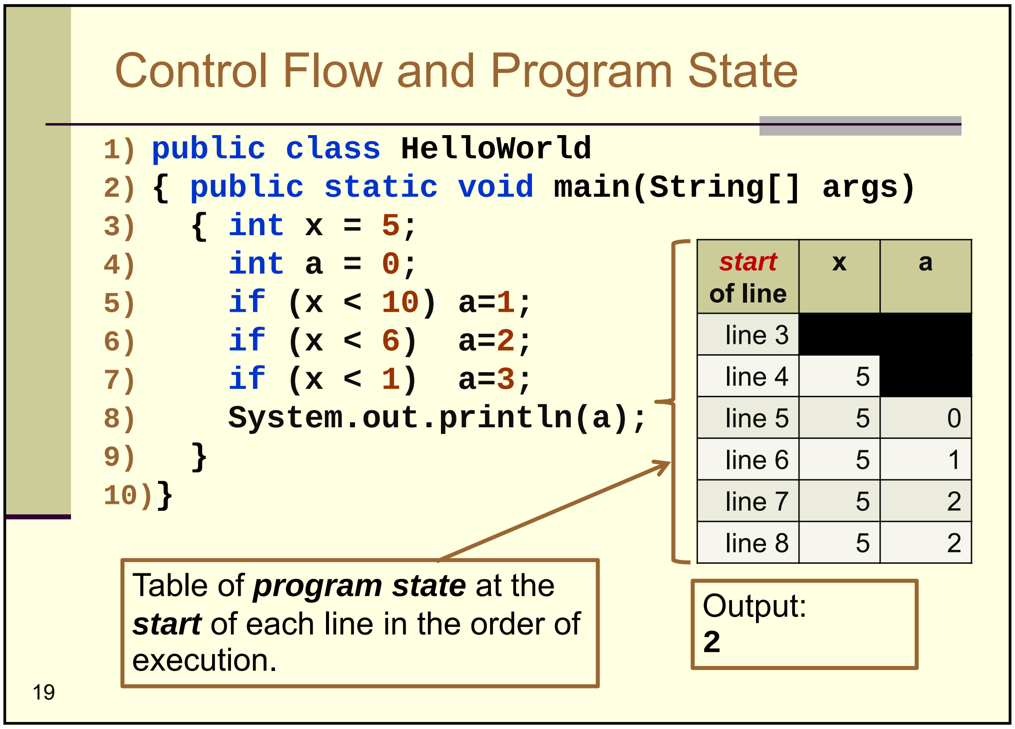


## Quiz: if and else if

```
int testscore = 88;
char grade = 'F';

if (testscore >= 60) grade = 'D';
else if (testscore >= 70) grade = 'C';
else if (testscore >= 80) grade = 'B';
else if (testscore >= 90) grade = 'A';

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

What would be the output of the above Java code?

a) B b) D c) F d) FB e) FDCB



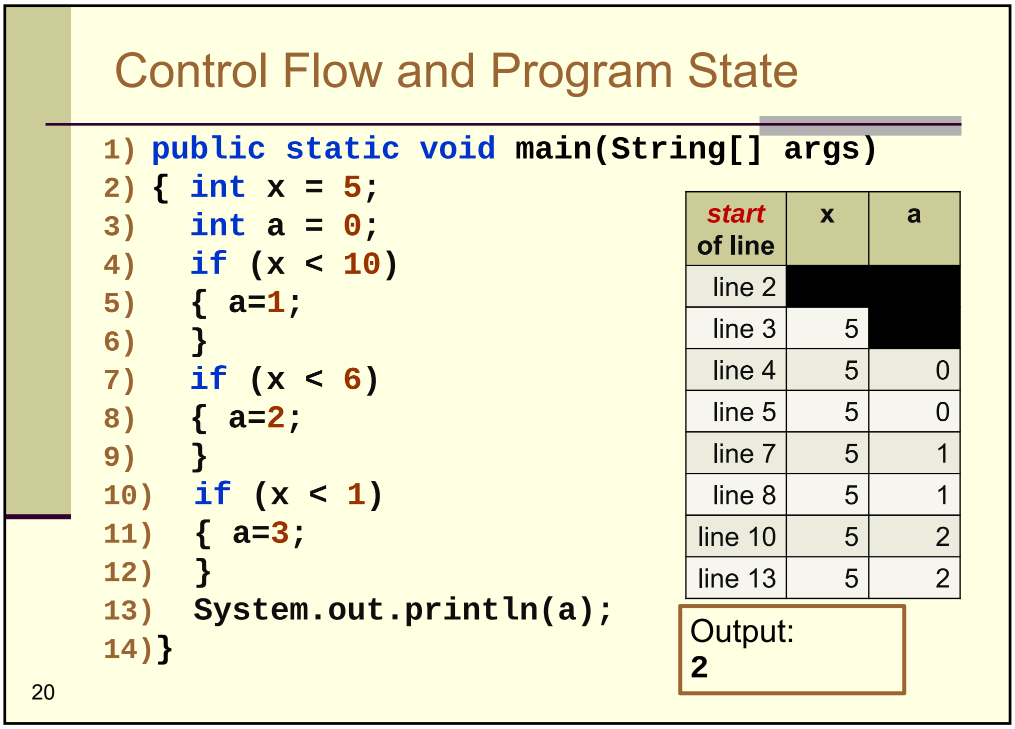
## Control Flow and Program State

```
1) public class HelloWorld
2) { public static void main(String[] args)
3)   { int x = 5;
4)     int a = 0;
5)     if (x < 10) a=1;
6)     if (x < 6)  a=2;
7)     if (x < 1)  a=3;
8)     System.out.println(a);
9)   }
10)}
```

| *start* of line | x | a |
|---|---|---|
| line 3 | ■ | ■ |
| line 4 | 5 | ■ |
| line 5 | 5 | 0 |
| line 6 | 5 | 1 |
| line 7 | 5 | 2 |
| line 8 | 5 | 2 |

Table of ***program state*** at the ***start*** of each line in the order of execution.

Output:
**2**

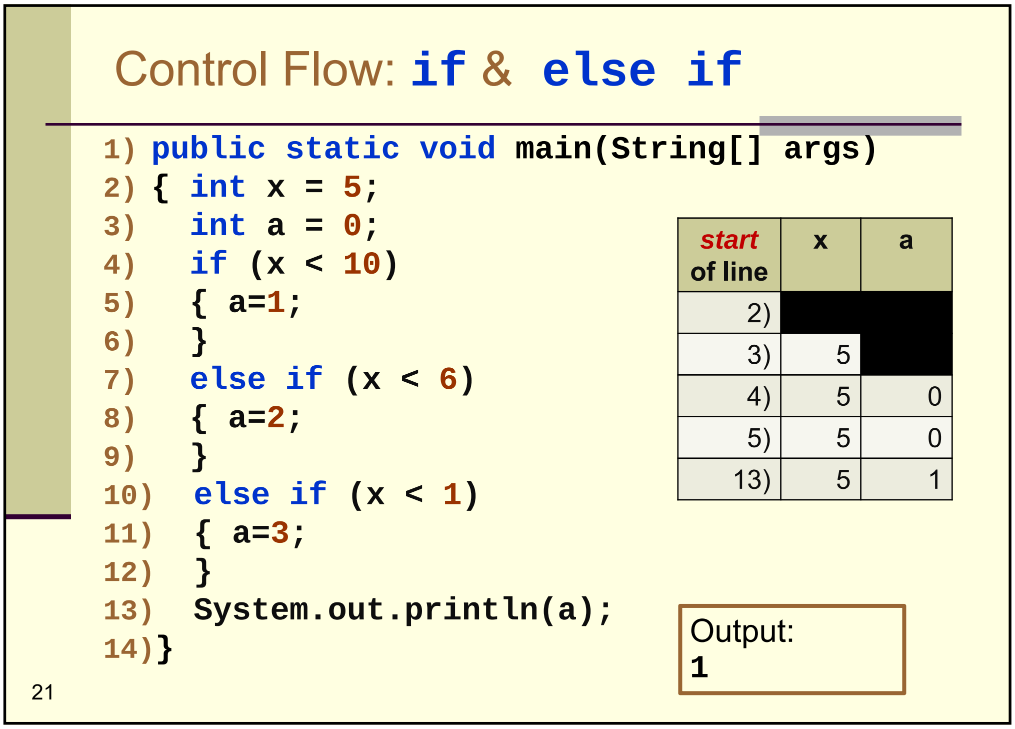


## Control Flow and Program State

```
1) public static void main(String[] args)
2) { int x = 5;
3)   int a = 0;
4)   if (x < 10)
5)   { a=1;
6)   }
7)   if (x < 6)
8)   { a=2;
9)   }
10)  if (x < 1)
11)  { a=3;
12)  }
13)  System.out.println(a);
14)}
```

| *start* of line | x | a |
|---|---|---|
| line 2 | ■ | ■ |
| line 3 | 5 | ■ |
| line 4 | 5 | 0 |
| line 5 | 5 | 0 |
| line 7 | 5 | 1 |
| line 8 | 5 | 1 |
| line 10 | 5 | 2 |
| line 13 | 5 | 2 |

Output:
**2**

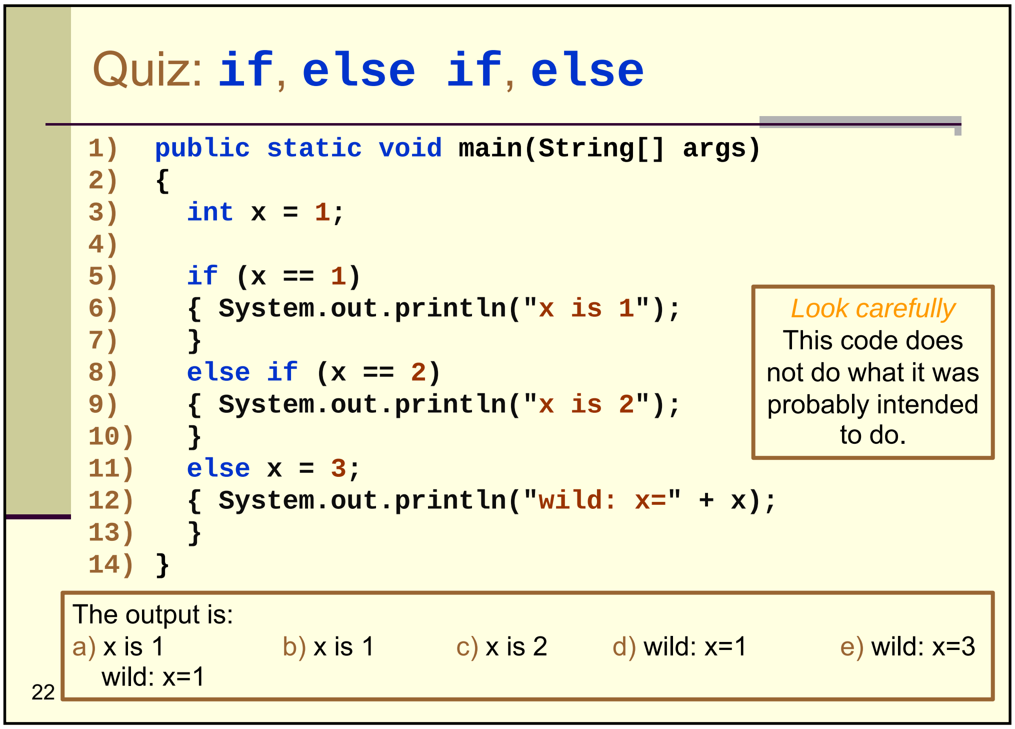


# Control Flow: if & else if

```
1) public static void main(String[] args)
2) { int x = 5;
3)   int a = 0;
4)   if (x < 10)
5)   { a=1;
6)   }
7)   else if (x < 6)
8)   { a=2;
9)   }
10)  else if (x < 1)
11)  { a=3;
12)  }
13)  System.out.println(a);
14)}
```

| *start* of line | x | a |
|---|---|---|
| 2) | ■ | ■ |
| 3) | 5 | ■ |
| 4) | 5 | 0 |
| 5) | 5 | 0 |
| 13) | 5 | 1 |

Output:
**1**

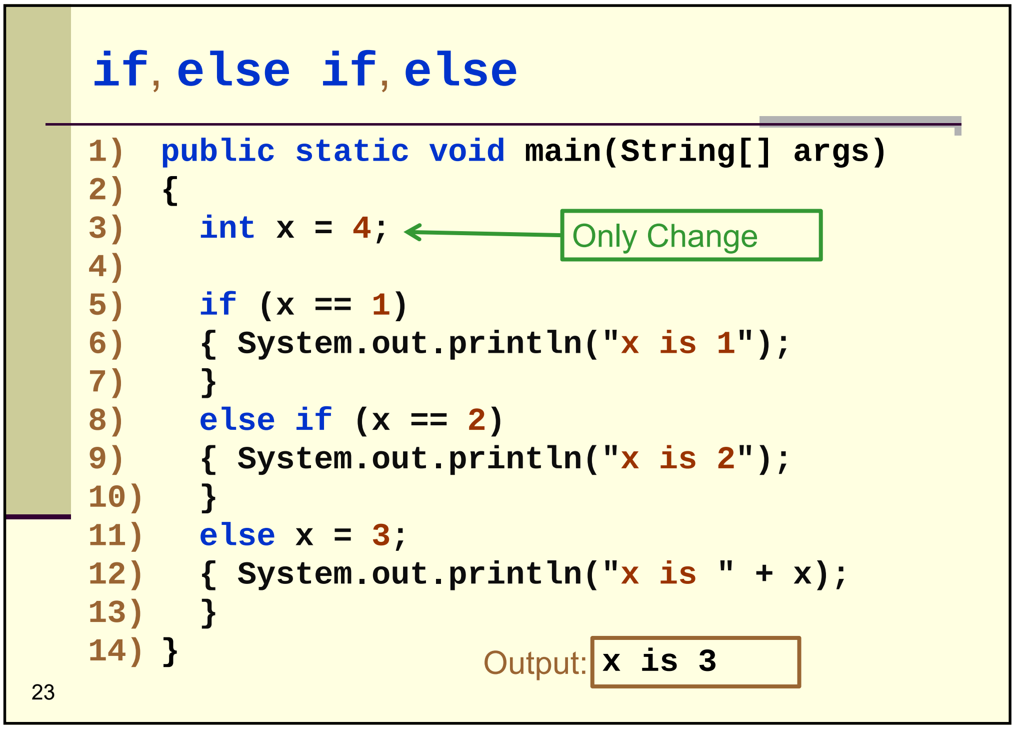


# Quiz: if, else if, else

```
public static void main(String[] args)
{
  int x = 1;

  if (x == 1)
  { System.out.println("x is 1");
  }
  else if (x == 2)
  { System.out.println("x is 2");
  }
  else x = 3;
  { System.out.println("wild: x=" + x);
  }
}
```

*Look carefully*
This code does not do what it was probably intended to do.

The output is:
a) x is 1
wild: x=1
b) x is 1
c) x is 2
d) wild: x=1
e) wild: x=3

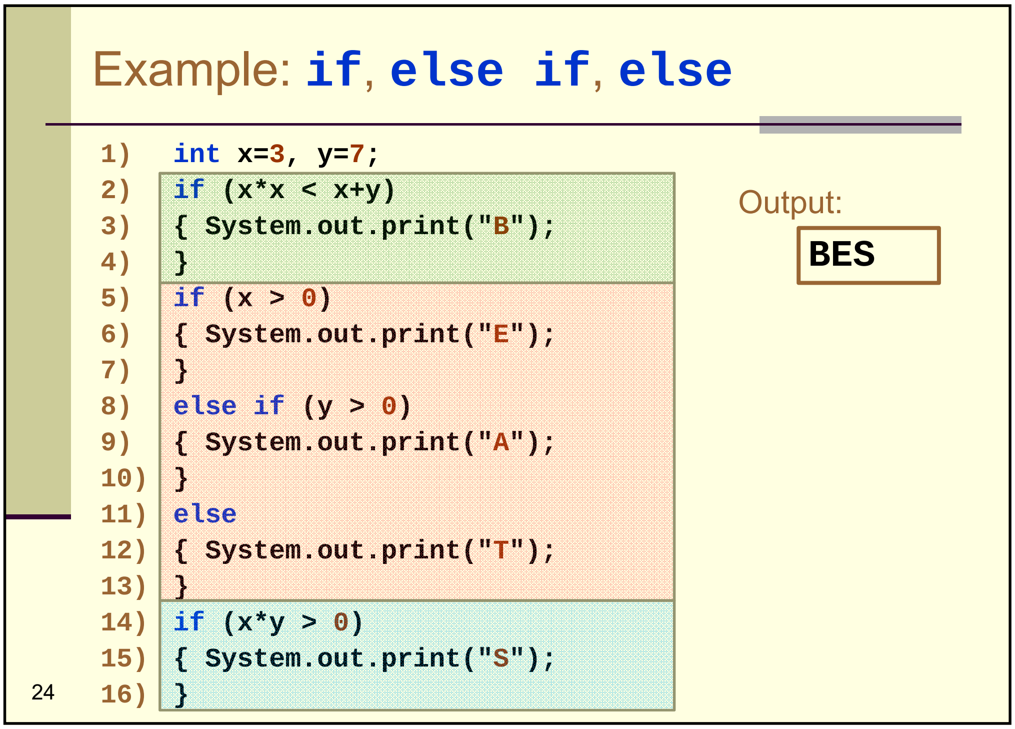


## if, else if, else

```
public static void main(String[] args)
{
  int x = 4;

  if (x == 1)
  { System.out.println("x is 1");
  }
  else if (x == 2)
  { System.out.println("x is 2");
  }
  else x = 3;
  { System.out.println("x is " + x);
  }
}
```

Only Change

Output: x is 3

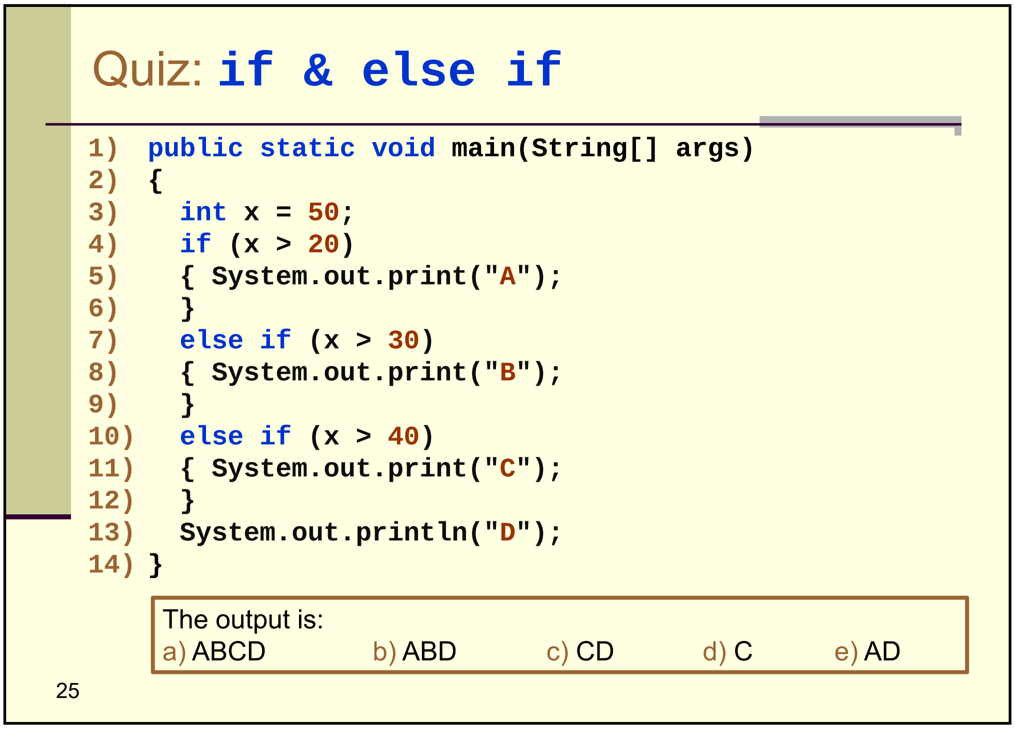


## Example: if, else if, else

```
int x=3, y=7;
if (x*x < x+y)
{ System.out.print("B");
}
if (x > 0)
{ System.out.print("E");
}
else if (y > 0)
{ System.out.print("A");
}
else
{ System.out.print("T");
}
if (x*y > 0)
{ System.out.print("S");
}
```

Output:

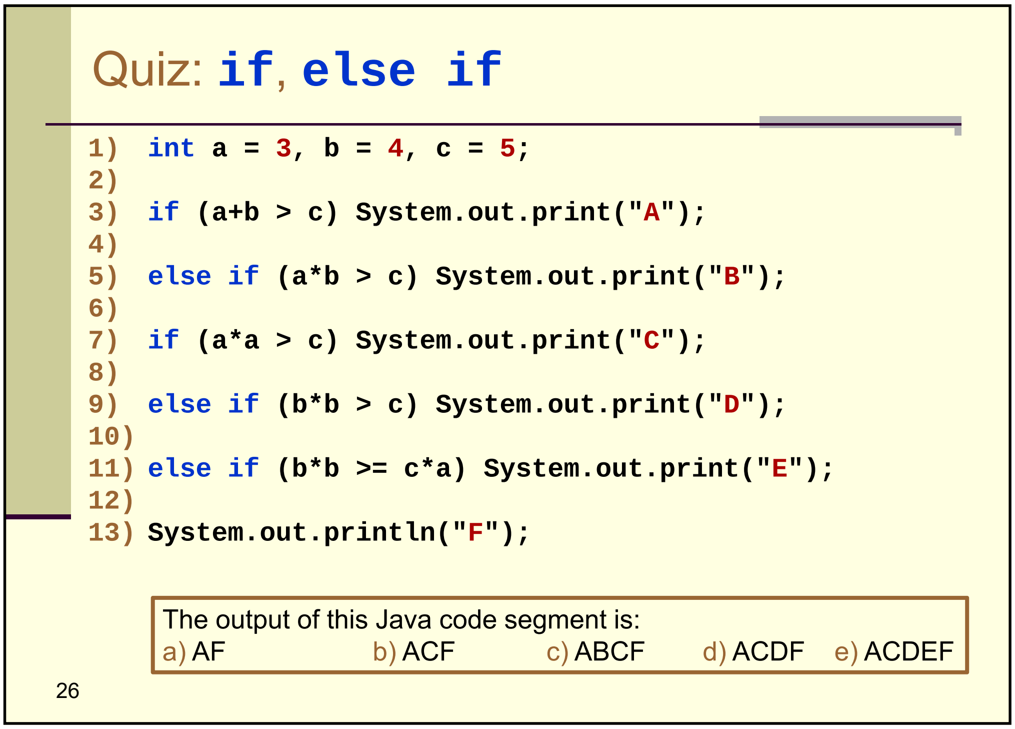
BES



## Quiz: `if & else if`

```java
public static void main(String[] args)
{
  int x = 50;
  if (x > 20)
  { System.out.print("A");
  }
  else if (x > 30)
  { System.out.print("B");
  }
  else if (x > 40)
  { System.out.print("C");
  }
  System.out.println("D");
}
```

The output is:
a) ABCD b) ABD c) CD d) C e) AD



## Quiz: `if`, `else if`

```java
int a = 3, b = 4, c = 5;

if (a+b > c) System.out.print("A");

else if (a*b > c) System.out.print("B");

if (a*a > c) System.out.print("C");

else if (b*b > c) System.out.print("D");

else if (b*b >= c*a) System.out.print("E");

System.out.println("F");
```

The output of this Java code segment is:
a) AF b) ACF c) ABCF d) ACDF e) ACDEF

## Logical Operators

| | |
|---|---|
| == | Equals |
| != | Not Equal |
| < | Less than |
| > | Grater than |
| <= | Less than or Equal to |
| >= | Greater than or Equal to |
| \|\| | Logical OR |
| && | Logical AND |
| ! | Logical NOT |

The bitwise operators are not covered in CS-259.

```
| bitwise OR
& bitwise AND
^ bitwise exclusive OR
~ bitwise NOT
```

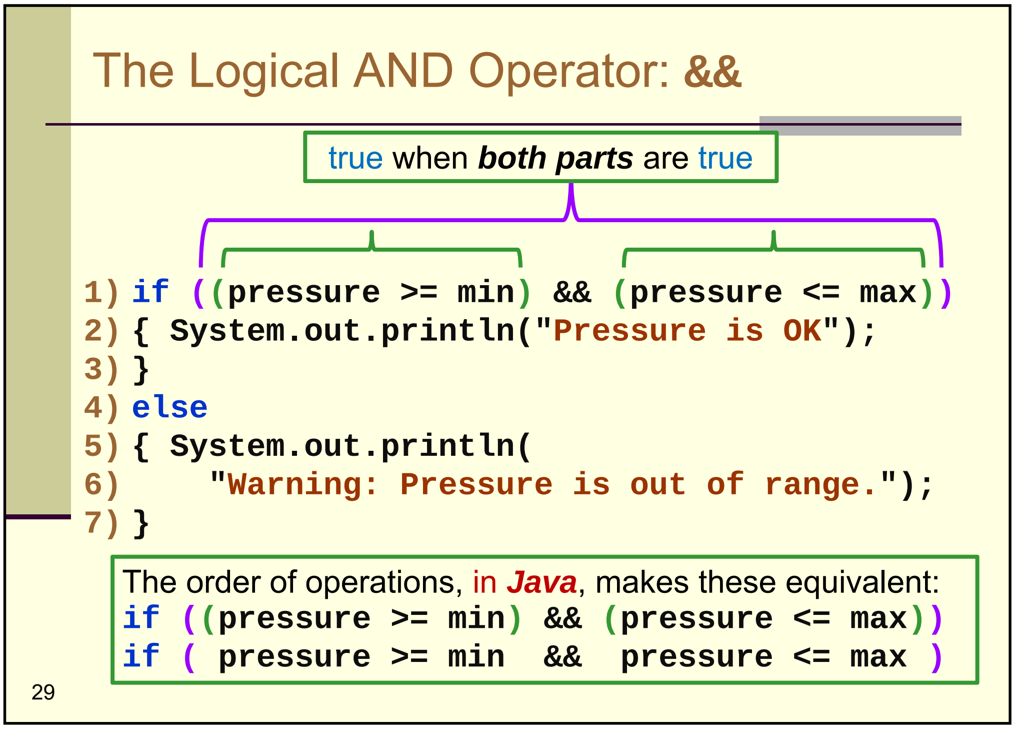
## Truth Tables for AND, OR and NOT

```
public class TruthTables
{
  public static void main(String[] args)
  {
    System.out.println( true && true );
    System.out.println( true && false );
    System.out.println( false && true);
    System.out.println( false && false );

    System.out.println( true || true );
    System.out.println( true || false );
    System.out.println( false || true );
    System.out.println( false || false );

    System.out.println(!true);
    System.out.println(!false);
  }
}
```

```
true
false
false
false
```

```
true
true
true
false
```

```
false
true
```



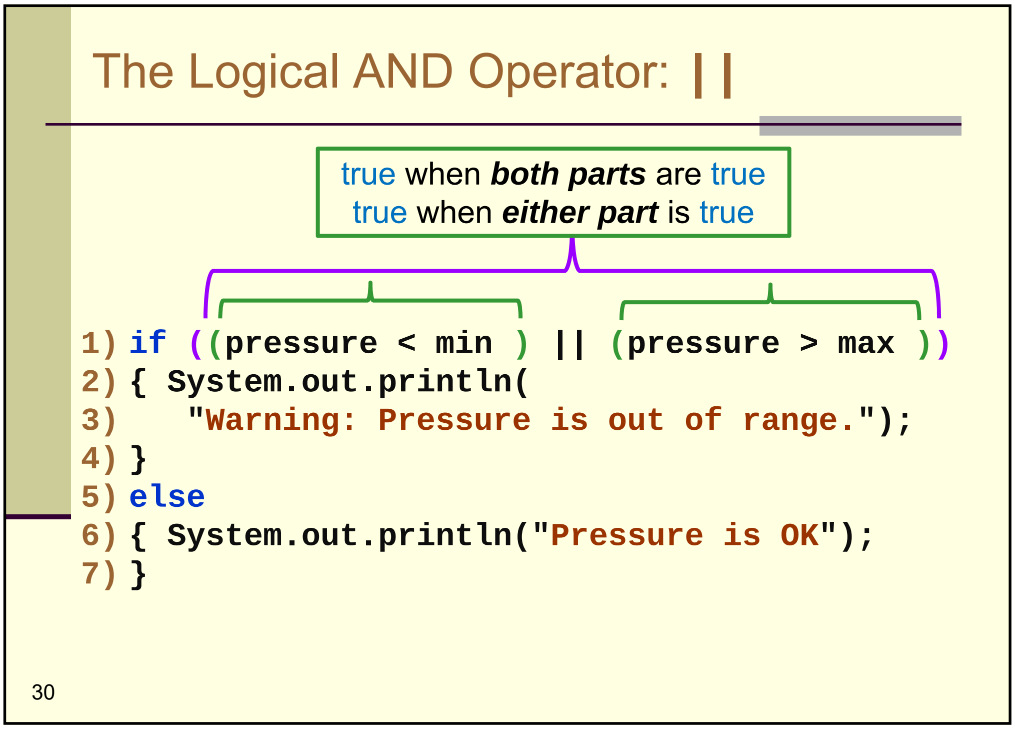
## The Logical AND Operator: &&

true when ***both parts*** are true

```
if ((pressure >= min) && (pressure <= max))
{ System.out.println("Pressure is OK");
}
else
{ System.out.println(
     "Warning: Pressure is out of range.");
}
```

The order of operations, in ***Java***, makes these equivalent:

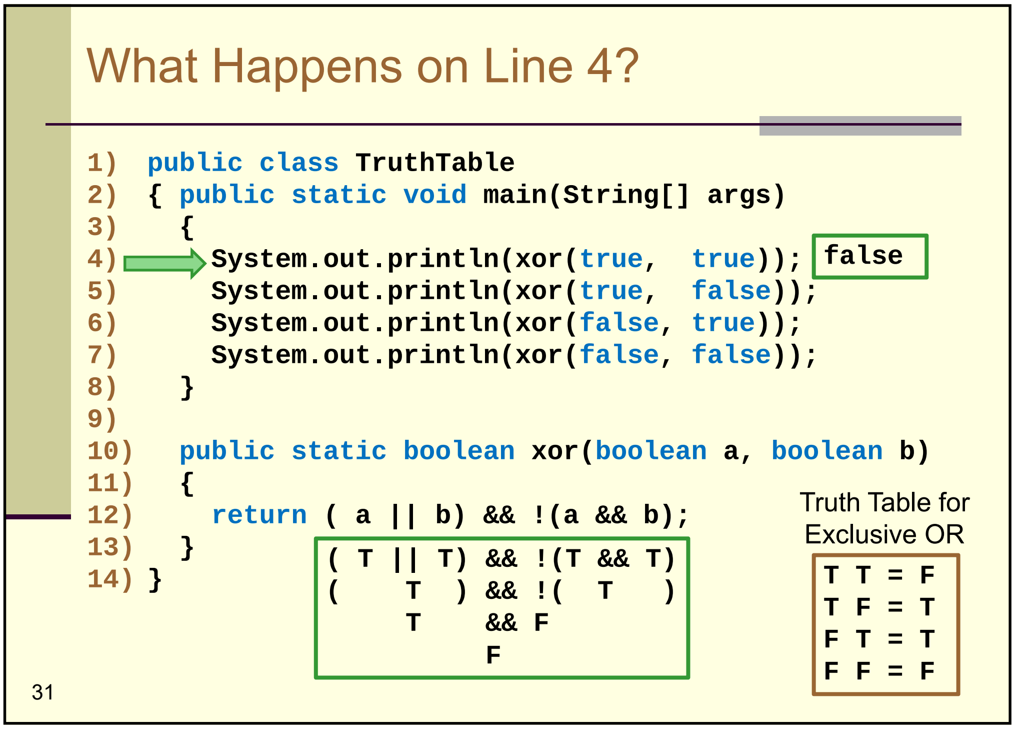
```
if ((pressure >= min) && (pressure <= max))
if ( pressure >= min  &&  pressure <= max )
```



## The Logical AND Operator: | |

~~true when ***both parts*** are true~~
true when ***either part*** is true

```
if ((pressure < min ) || (pressure > max ))
{ System.out.println(
    "Warning: Pressure is out of range.");
}
else
{ System.out.println("Pressure is OK");
}
```



## What Happens on Line 4?

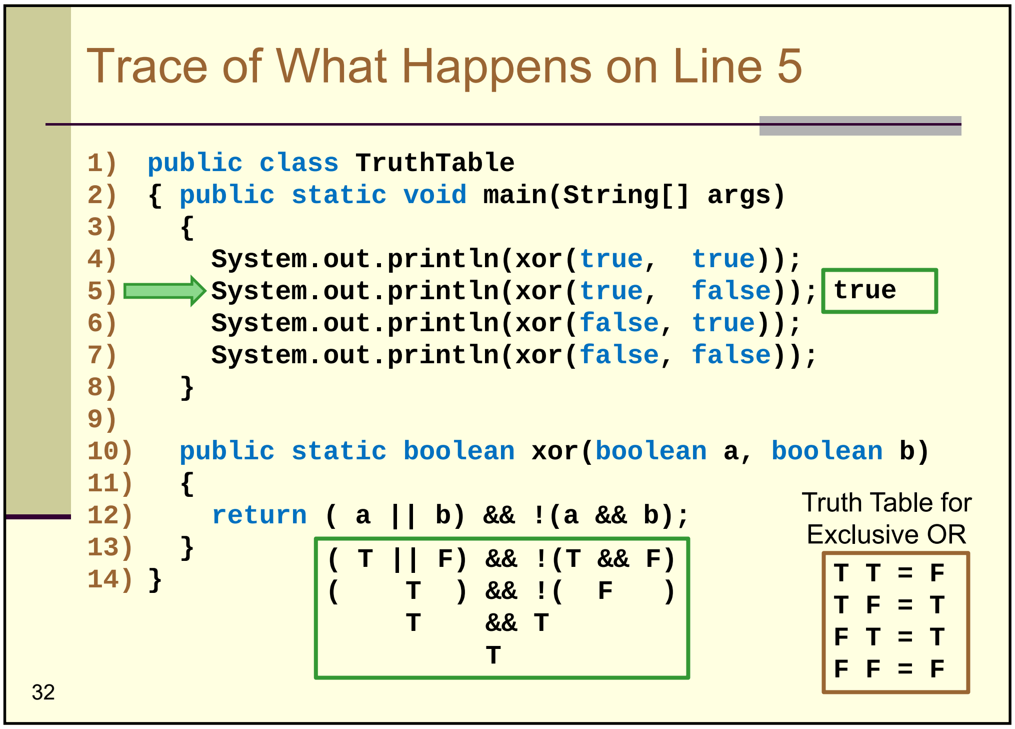
```
1)  public class TruthTable
2)  { public static void main(String[] args)
3)    {
4)      System.out.println(xor(true,  true));
5)      System.out.println(xor(true,  false));
6)      System.out.println(xor(false, true));
7)      System.out.println(xor(false, false));
8)    }
9)
10)   public static boolean xor(boolean a, boolean b)
11)   {
12)     return ( a || b) && !(a && b);
13)   }
14) }
```

```
false
```

```
( T || T) && !(T && T)
(    T  ) && !(  T   )
     T    && F
          F
```

Truth Table for Exclusive OR

```
T T = F
T F = T
F T = T
F F = F
```



## Trace of What Happens on Line 5

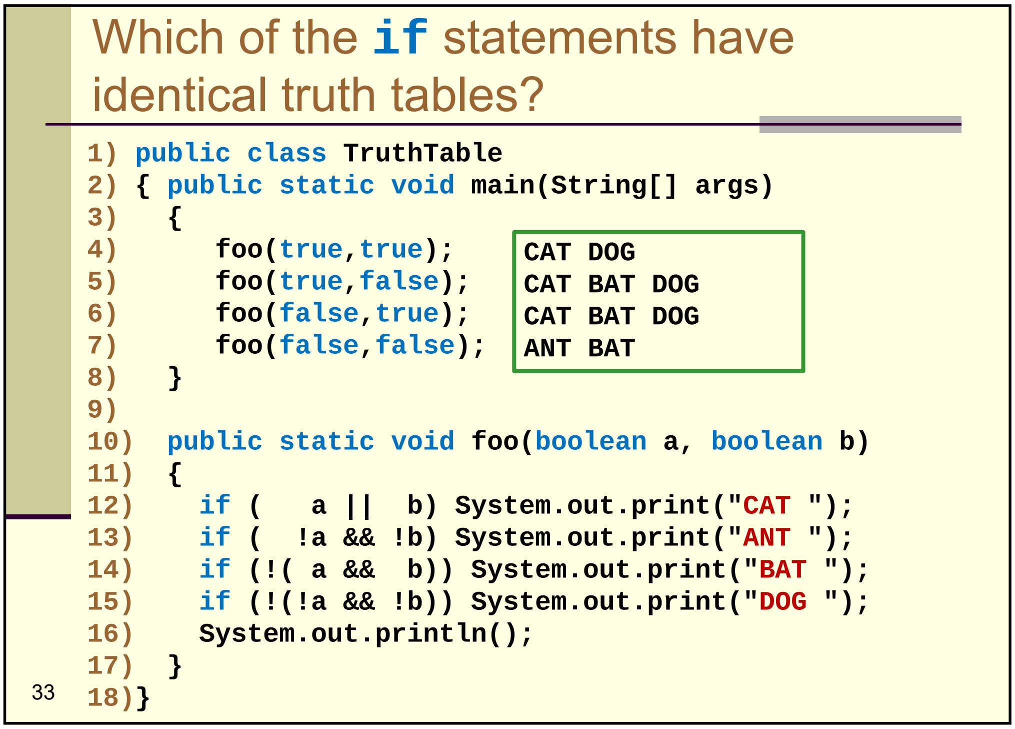
```
1)  public class TruthTable
2)  { public static void main(String[] args)
3)    {
4)      System.out.println(xor(true,  true));
5)      System.out.println(xor(true,  false));
6)      System.out.println(xor(false, true));
7)      System.out.println(xor(false, false));
8)    }
9)
10)   public static boolean xor(boolean a, boolean b)
11)   {
12)     return ( a || b) && !(a && b);
13)   }
14) }
```

```
true
```

```
( T || F) && !(T && F)
(    T  ) && !(  F   )
     T    && T
          T
```

Truth Table for Exclusive OR

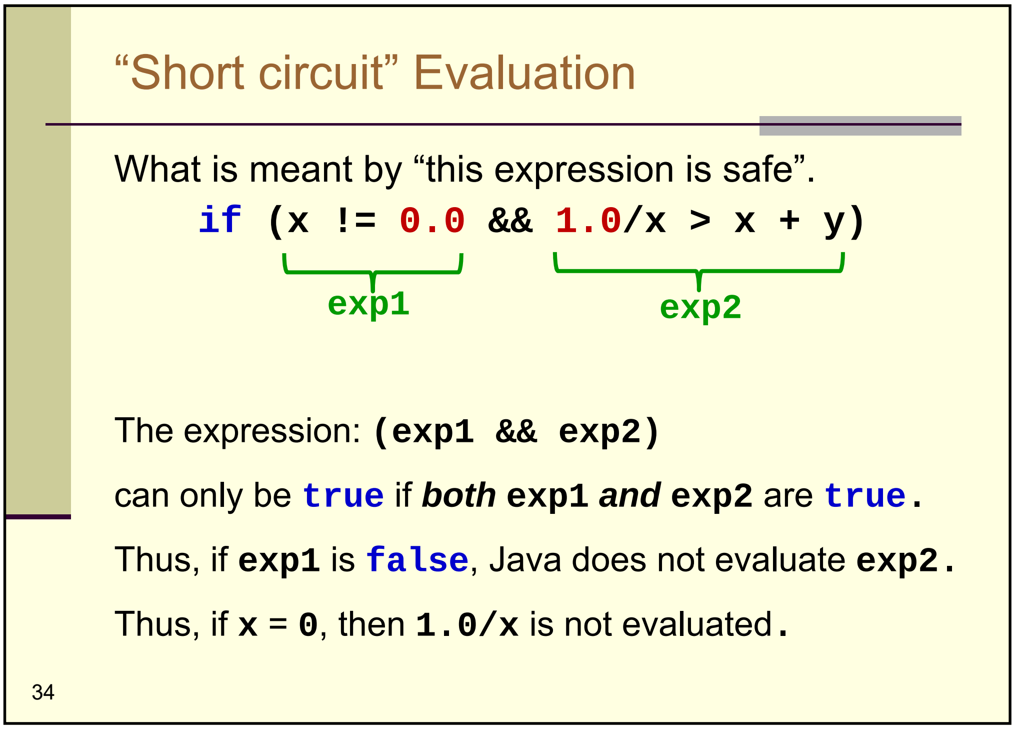
```
T T = F
T F = T
F T = T
F F = F
```



## Which of the **if** statements have identical truth tables?

```java
public class TruthTable
{ public static void main(String[] args)
  {
      foo(true,true);
      foo(true,false);
      foo(false,true);
      foo(false,false);
  }

  public static void foo(boolean a, boolean b)
  {
    if (   a ||  b) System.out.print("CAT ");
    if (  !a && !b) System.out.print("ANT ");
    if (!( a &&  b)) System.out.print("BAT ");
    if (!(!a && !b)) System.out.print("DOG ");
    System.out.println();
  }
}
```

```
CAT DOG
CAT BAT DOG
CAT BAT DOG
ANT BAT
```



## “Short circuit” Evaluation

What is meant by “this expression is safe”.

$$\texttt{if (}\underbrace{\texttt{x != 0.0}}_{\texttt{exp1}}\ \texttt{\&\&}\ \underbrace{\texttt{1.0/x > x + y}}_{\texttt{exp2}}\texttt{)}$$

The expression: **`(exp1 && exp2)`**

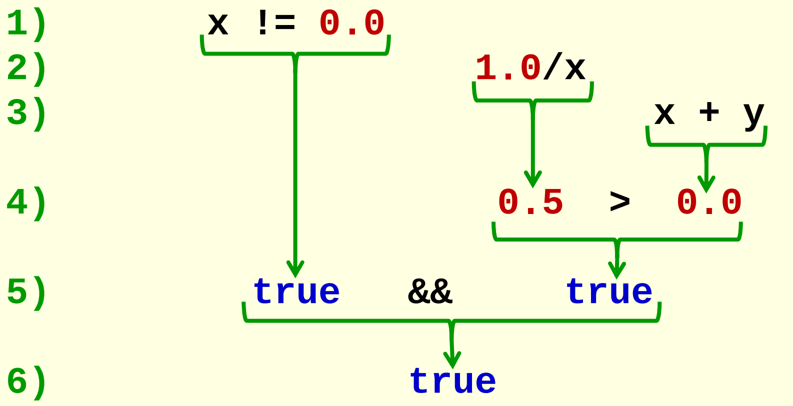
can only be **`true`** if ***both*** **`exp1`** ***and*** **`exp2`** are **`true`**.

Thus, if **`exp1`** is **`false`**, Java does not evaluate **`exp2`**.

Thus, if **`x`** = **`0`**, then **`1.0/x`** is not evaluated.

## Order of Evaluation

```
double x =  2.0;
double y = -2.0;
if (x != 0.0 && 1.0/x > x + y)
```



1) x != 0.0

2) 1.0/x

3) x + y

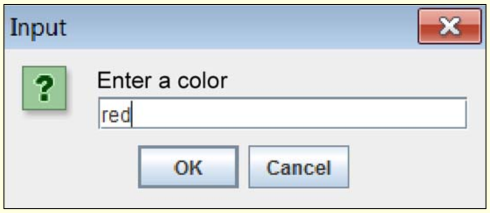
4) 0.5 > 0.0

5) true && true

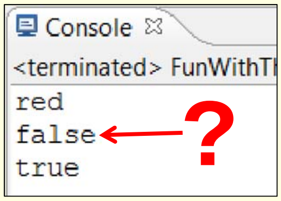
6) true

## Comparing Strings

```
String myColor = "red"
System.out.println(myColor);
System.out.println(myColor == "red");
System.out.println(myColor.equals("red"));
```



red



```
<terminated> FunWithTl
red
false
true
```

## Comparing Addresses verses Strings

```
String myColor;
//Some code that puts data in myColor.
```

```
myColor == "red"
```

**true** if and only if the ***memory location*** of **myColor** is the same as the ***memory location*** of "**red**"

```
myColor.equals("red")
```

**true** if and only if the ***data*** in **myColor** is "**red**" .

## Possible Null Pointer Exception

```
1)   String str = JOptionPane.showInputDialog(null,
2)     "Enter red");
3)   if (str.equals(""))
4)   { System.out.println("Clicked OK with no data");
5)   }
6)   if (str == null)
7)   { System.out.println("Clicked Cancel");
8)   }
```

A Null Pointer Exception occurs when code attempts to access a ***member of*** an object using an object reference that is not pointing anywhere.

In line 3, **equals** is a method that is a ***member of*** a String object.

However, if **str** is **null**, then, even though str was ***declared as*** being a ***reference to*** a String object, it doesn't, in fact, ***point to*** anything.

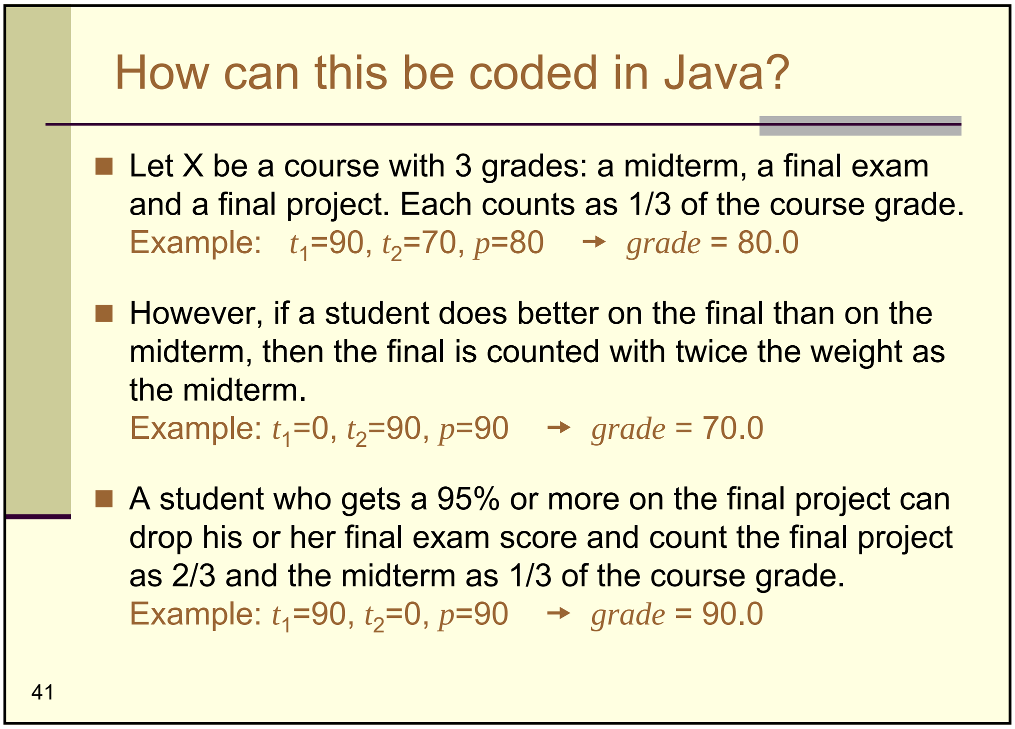
# Does This Fix the Problem?

```
String str = JOptionPane.showInputDialog(null,
  "Enter red");
if (str == null)
{ System.out.println("Clicked Cancel");
}
if (str.equals(""))
{ System.out.println("Clicked OK with no data");
}
```

# Checking Input: All Golden

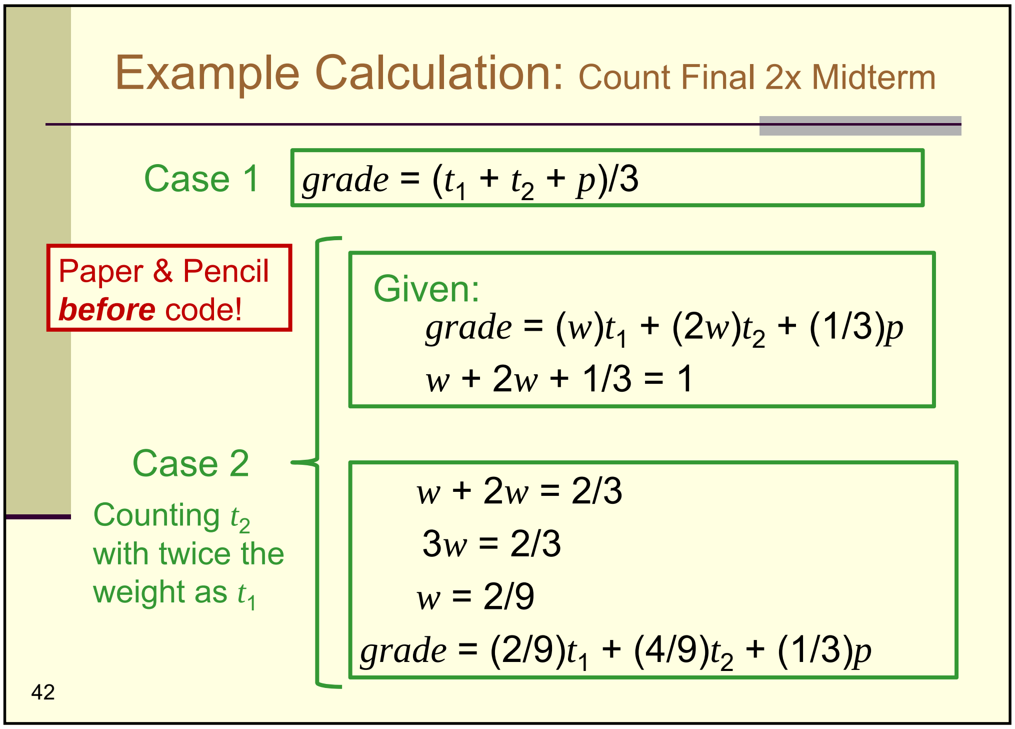
```
String str = JOptionPane.showInputDialog(
   null, "Enter red");
if (str == null)
{ System.out.println("Clicked Cancel");
}
else if (str.equals(""))
{ System.out.println("OK with no data");
}
```

```
String str = JOptionPane.showInputDialog(
   null, "Enter red");
if (str == null)
{ System.out.println("Clicked Cancel");
  System.exit(0);
}
if (str.equals(""))
{ System.out.println("OK with no data");
}
```



## How can this be coded in Java?

- Let X be a course with 3 grades: a midterm, a final exam and a final project. Each counts as 1/3 of the course grade.
  Example: $t_1$=90, $t_2$=70, $p$=80 → $grade$ = 80.0
- However, if a student does better on the final than on the midterm, then the final is counted with twice the weight as the midterm.
  Example: $t_1$=0, $t_2$=90, $p$=90 → $grade$ = 70.0
- A student who gets a 95% or more on the final project can drop his or her final exam score and count the final project as 2/3 and the midterm as 1/3 of the course grade.
  Example: $t_1$=90, $t_2$=0, $p$=90 → $grade$ = 90.0



## Example Calculation: Count Final 2x Midterm

Case 1 $\boxed{grade = (t_1 + t_2 + p)/3}$

**Paper & Pencil *before* code!**

Given:

$$grade = (w)t_1 + (2w)t_2 + (1/3)p$$

$$w + 2w + 1/3 = 1$$

Case 2

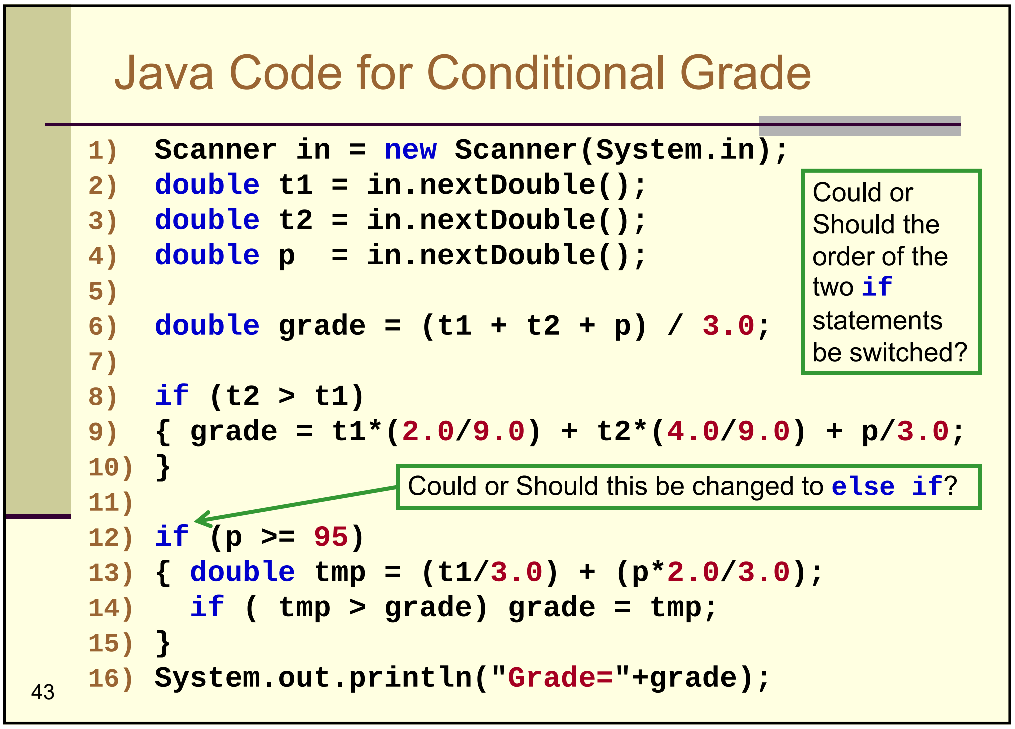
Counting $t_2$ with twice the weight as $t_1$

$$w + 2w = 2/3$$

$$3w = 2/3$$

$$w = 2/9$$

$$grade = (2/9)t_1 + (4/9)t_2 + (1/3)p$$



# Java Code for Conditional Grade

```
Scanner in = new Scanner(System.in);
double t1 = in.nextDouble();
double t2 = in.nextDouble();
double p  = in.nextDouble();

double grade = (t1 + t2 + p) / 3.0;

if (t2 > t1)
{ grade = t1*(2.0/9.0) + t2*(4.0/9.0) + p/3.0;
}

if (p >= 95)
{ double tmp = (t1/3.0) + (p*2.0/3.0);
  if ( tmp > grade) grade = tmp;
}
System.out.println("Grade="+grade);
```

Could or Should the order of the two **if** statements be switched?

Could or Should this be changed to **else if**?