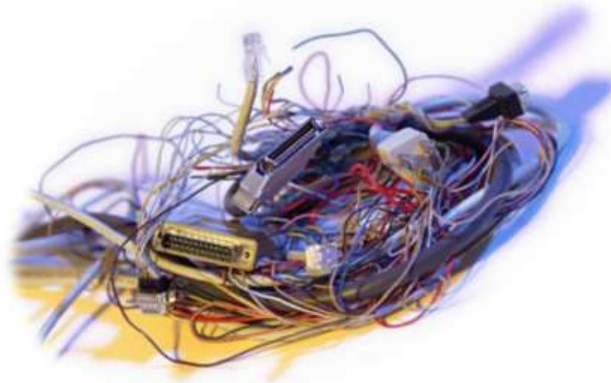


Nested IF

GEEN163

```
=IF(B18<=5,0,IF(AND(B18>5,B18<25),0.1,IF(AND(B18>24,B18<50),0.15,IF(AND(B18>49,B18<75),0.2,0.25))))
```



*“Some persons are very decisive
when it comes to avoiding
decisions.”*

Brendan Francis

TuringsCraft Assignment

- Answer at least 34 of the 62 questions in sections 5.1 – 5.5 of the TuringsCraft tutoring system
- You will earn 3 points for each correct answer
- Due midnight on Wednesday, October 2, 2013

Round Off Error with double

- If a program multiplies 3 times 4.95, it does not get exactly 14.85 like it should

```
double price = 4.95;  
double cost = 3.0 * price;  
System.out.println( cost );
```

displays 14.850000000000000001

- Some decimal fractions cannot be represented accurately in binary just like $1/3$ cannot be represented accurately in decimal

double Equality

- double variables store a binary approximation of a decimal value, accurate to about 16 decimal digits
- The result of calculations with doubles may not be exactly accurate
- The equality test (`==`) is true only if both sides are **exactly** equal

double if

- Some if statements with doubles will fail when you think they should be true

```
double dog = 3.0 * 4.95;  
if (dog == 14.85) // this is false
```

- Note that integers do not have round off error although they do drop the fraction part with integer division

Almost Equal

- When comparing doubles, it is best to see if they are almost equal

```
double dog = 3.0 * 4.95;
```

```
if (Math.abs(dog - 14.85) < 0.0001 ) // this is true
```

- *Math.abs* returns the absolute value
- You can determine the level of accuracy required

Double Zero

- If your program reads the value zero as input, it will be reliably stored as zero
- You can reliably compare double input values if you have not done any arithmetic with them

```
double dog = keyboard.nextDouble();
```

```
if ( dog == 0.0 ) // this will work correctly
```


Comparing Strings

- The comparison operators (such as `>`, `<`, `==`) do **not** work with Strings. They only work on primitive data types (`int`, `float`, `double` and `char`)
- You need to use the `equals` or `compareTo` methods to compare Strings.

equals Method

- equal is a boolean method of String that is true if the parameter has exactly the same characters

```
String cat = "frog", dog = "toad", cow = "frog";
```

```
cat.equals(dog) is false
```

```
cat.equals(cow) is true
```

Ignoring Case

- You can use `equalsIgnoreCase` to compare Strings regardless of upper or lower case

```
System.out.print("yes or no?");  
String yesNo = keyReader.next();  
if (yesNo.equalsIgnoreCase("NO")) {  
    ...  
}
```

What is gpa if “Williams” is entered?

```
double gpa = 0.0;
String lastName;
lastName = keyboard.next();
if ( lastName == “Williams” )
    gpa = 4.0;
else
    gpa = 2.0;
```

- A. 0.0
- B. 2.0
- C. 4.0
- D. none of the above

Possible Solution

```
double gpa = 0.0;
String lastName;
lastName = keyboard.next();
if ( lastName .equals("Williams") )
    gpa = 4.0;
else
    gpa = 2.0;
```

String Comparison

- There are several string comparison methods
- These methods return true or false

```
dog.equals( cat )
```

```
dog.equalsIgnoreCase( cat )
```

- These methods return 0 for equal, negative if dog < cat and positive if dog > cat

```
dog.compareTo( cat )
```

```
dog.compareToIgnoreCase( cat )
```

What is displayed?

```
double rate = 3.4;
double two = rate / 100.0 / 12.0;
double one = rate / 1200.0;
if ( one == two ) {
    System.out.println("same");
} else {
    System.out.println("diff");
}
```

- A. same
- B. diff
- C. cannot be determined
- D. none of the above

Multiple IFs

- Sometimes there are more than two logical options for what is to be done
- Multiple if statements can be run sequentially

if (*compare 1*)

statement A; // compare 1 is true

else if (*compare 2*)

statement B; // compare 1 is false and compare 2 is true

else

statement C; // both compare 1 and compare 2 are false

Multiple Alternatives

- The **else if** is generally used when there are multiple possible outcomes
- If the first comparison is not true then the program can use an **else if** to consider a second

else if

- An else clause can be immediately followed by another if statement.

```
if (cow == 3)
```

```
    x = 47; // only if cow is 3
```

```
else if (cow > 5)
```

```
    x = 2; // only if cow is not 3 and cow > 5
```

```
int cat=3 , dog=5 , cow=7 , rat=8 ;  
if (cat < dog)  
    rat = 6 ;  
else if (dog < cow)  
    rat = 4 ;
```

What is rat now?

- A. 4
- B. 6
- C. 8
- D. false

Multiple if else

- You can connect several if else statements

```
int a, b, c, x; // values input before if
if (a == 4)
    x = 12; // only if a is 4
else if (b == 3)
    x = 15; // only if a is not 4 and b is 3
else if (c == 2)
    x = 17; // only if a is not 4, b is not 3 and
            // c is 2
```

Final else

- The statement after the last else is executed only if all of the previous if else boolean expressions are false.

```
if (a == 3)
```

```
    x = 101; // only if a is 3
```

```
else if (b == 7)
```

```
    x = 102; // only if a is not 3 and b is 7
```

```
else
```

```
    x = 103; // if a is not 3 and b is not 7
```

Skipping the Rest

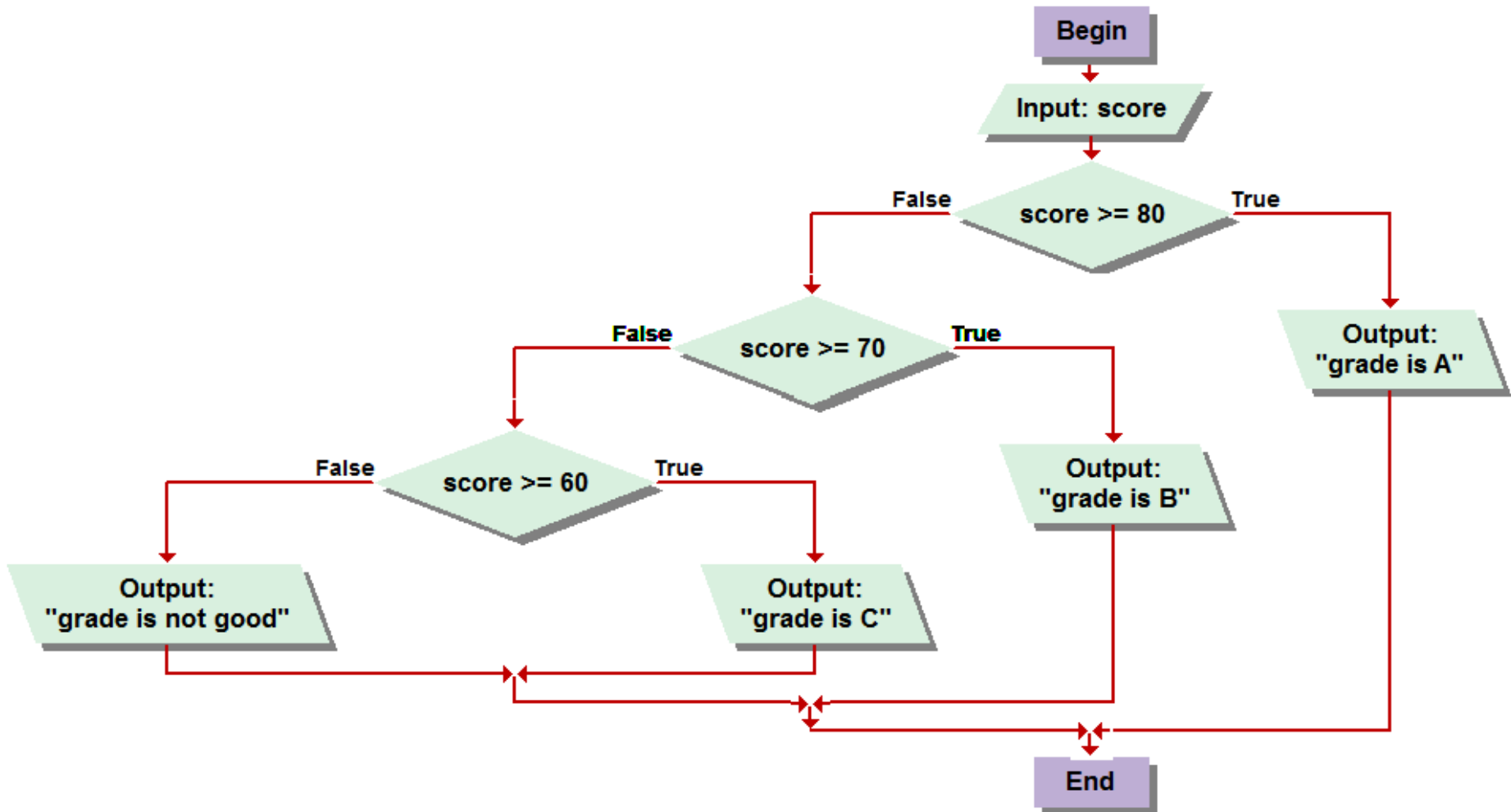
- In a long else if statement the statement after the first true comparison will be executed
- All the remaining else if statements in the list will be skipped

```
int cat=2,dog=4,bird=6, rat=8;  
if (cat >= dog)  
    rat = 5;  
else if (dog < bird)  
    rat = 3;
```

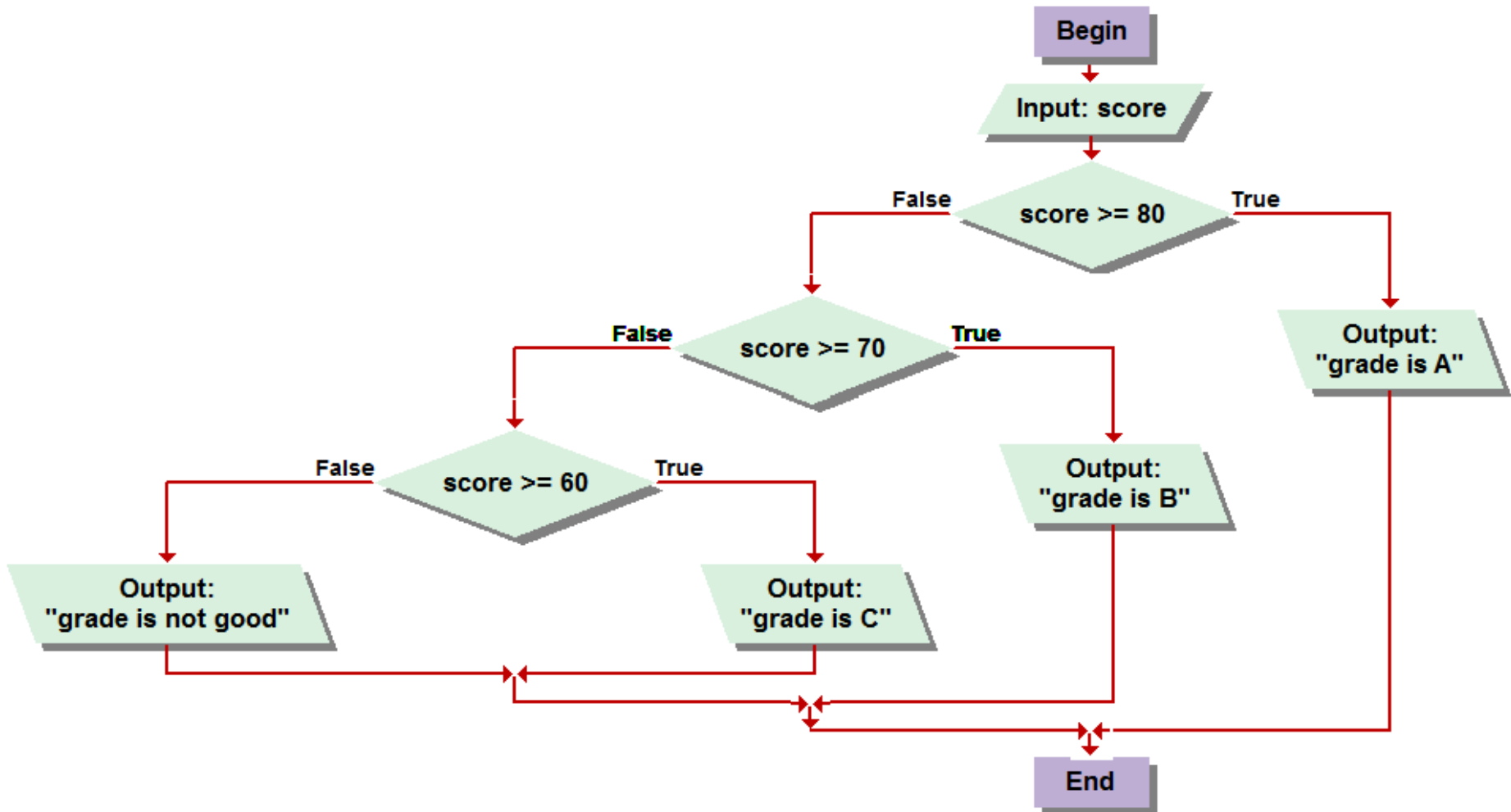
What is rat now?

- A. 3
- B. 4
- C. 5
- D. 8

else if Flowchart



Write the Java if statements



Possible Solution

```
if (score >= 80) {  
    System.out.println("Grade is A");  
} else if (score >= 70) {  
    System.out.println("Grade is B");  
} else if (score >= 60) {  
    System.out.println("Grade is C");  
} else {  
    System.out.println("Grade is not good");  
}
```

Unnecessary Logic

```
if (score >= 80) {  
    System.out.println("Grade is A");  
} else if (score >= 70 && score < 80) {  
    System.out.println("Grade is B");  
} else if (score >= 60 && score < 70) {  
    System.out.println("Grade is C");  
} else {  
    System.out.println("Grade is not good");  
}
```

Next Statement

- If a comparison is false, the next statement after an **if** is skipped

```
if (dog == cat)  
    cow = bull;
```

- The next statement can be another **if**

```
if (dog == cat)  
    if (rat == mouse)  
        cow = bull;  
  
    else  
        goat = 47;
```

nested if

```
if (a == 3)
```

```
    if (b == 7)
```

```
        x = 1; // a is 3 and b is 7
```

```
    else
```

```
        x = 2; // a is 3 and b is not 7
```

```
else
```

```
    x = 3; // a is not 3, b doesn't matter
```

Nested Indenting

- Lines that are inside two `if`s should be indented twice
- The following is exactly the same as the previous slide, but much harder to read

```
if (a==3) if (b==7) x=1 ; else  
x=2 ; else x=3 ;
```

Connecting **else** to **if**

- An else statement is always related to the if of the previous block or statement

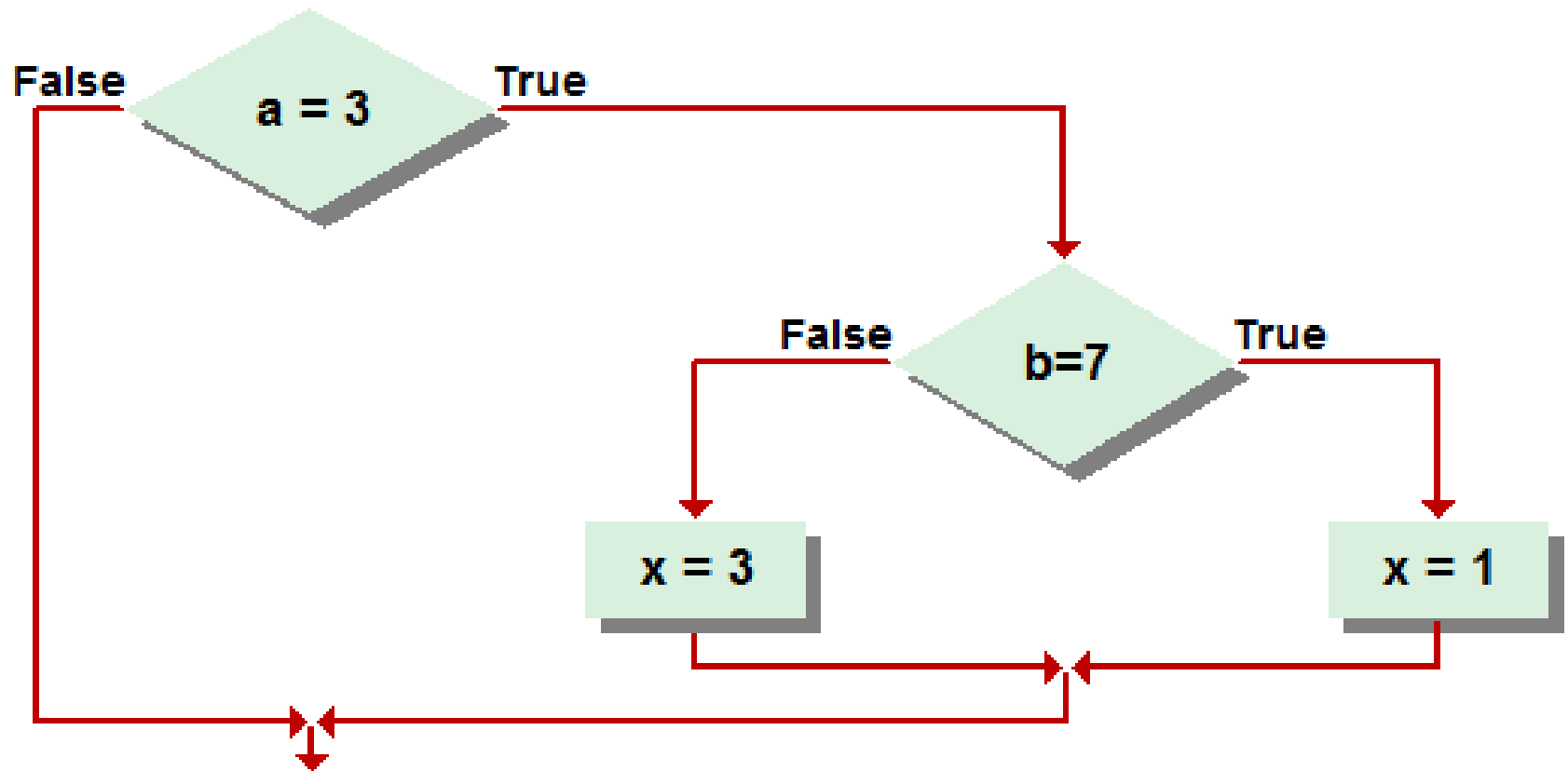
```
int a, b, c, x; // values input before if
if (a == 3)
    if (b == 7)
        x = 1; // a is 3 and b is 7
    else
        x = 3; // a is 3 and b is not 7
```

Connecting **else** to **if**

- The use of brackets to define blocks helps to clarify the nesting of if statements

```
int a, b, c, x; // values input before if
if (a == 3) {
    if (b == 7) {
        x = 1; // a is 3 and b is 7
    } else {
        x = 3; // a is 3 and b is not 7
    }
}
```


Nested if Flowchart

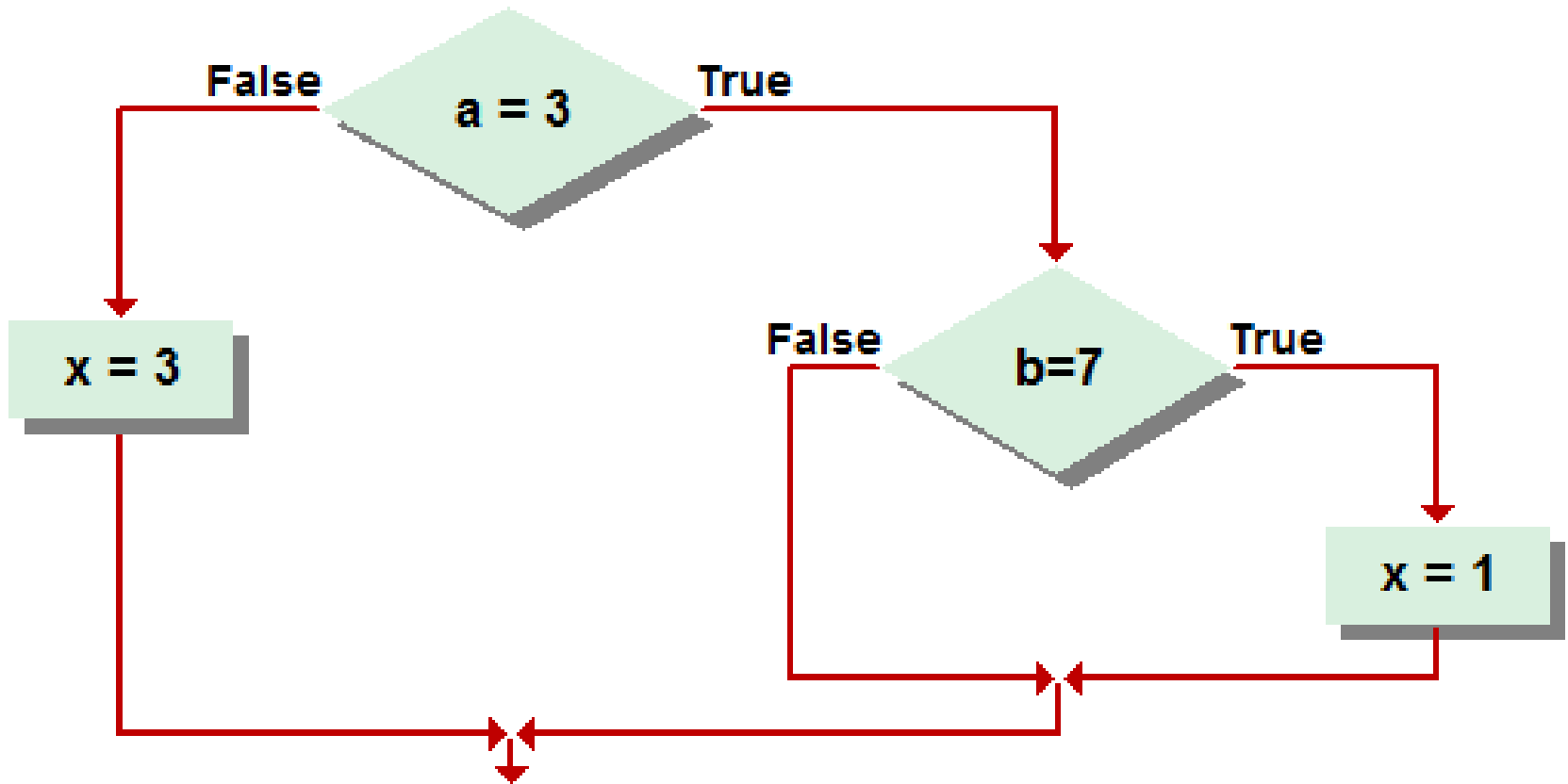


Connecting **else** to **if**

- The use of brackets to define blocks helps when the nesting gets complicated,

```
int a, b, c, x; // values input before if
if (a == 3) {
    if (b == 7) {
        x = 1; // a is 3 and b is 7
    }
} else {
    x = 3; // a is not 3 (b doesn't matter)
}
```

Nested if Flowchart



```
int cat=3, dog=5, cow=7, rat=8;  
if (cat > dog) // bad indenting  
if (dog < cow)  
rat = 9;  
else  
rat = 6;
```

What is rat now?

A. 5

B. 6

C. 8

D. 9

```
int cat=3, dog=5, cow=7, rat=8;
if (cat < dog) // bad indenting
if (dog < cow)
rat = 9;
else
rat = 6;
```

What is rat now?

A. 5

B. 6

C. 8

D. 9


Short Circuit Evaluation

- For a logical AND (**&&**) to be true, both sides of the **&&** have to be true
- When evaluating a logical expression with an **&&**, if Java determines that the left side is false, there is no need to evaluate the right side
- Avoiding evaluating the right part of an **&&** can be beneficial if an error might occur

No Need to Continue

- What is the result of the `if` ?

```
int dog = 5, cat = 3;
```

```
if ( dog == 47 &&  )
```

- Once Java determines the left side of the `&&` is false, the right side does not matter

Short Circuit Advantage

```
int dog = 3, goat = 17;  
System.out.print("Enter a number >");  
dog = keyboard.nextInt();  
if (dog != 0 && goat/dog < 4) {  
    goat = 4;  
}
```

- If a zero is read into dog, you will get a divide fault if goat/dog is executed

Short Circuit OR

- If either side of a logical OR (`||`) is true, then the whole expression will be true
- Java stops evaluating an expression with an `||` once it finds a true part

Short Circuit Advantage

```
int dog = 3, goat = 17;  
System.out.print("Enter a number >");  
dog = keyboard.nextInt();  
if (dog == 0 || goat/dog < 4) {  
    goat = 4;  
}
```

- If a zero is read into dog, you will get a divide fault if goat/dog is executed

Conditional Assignment

- Java provides a *little known* method for putting an if statement in the middle of an expression

logical expression ? true part : false part

```
dog = cat == 0 ? cow : goat;
```

- if cat is equal to zero, set dog to the value of cow, else set dog to the value of goat.

Conditional Example

```
int cow= 3, cat = 5, dog = 7, goat = 17;  
dog = keyboard.nextInt();
```

```
cow = (dog == 0 ? cat : goat) + 1;
```

is the same as

```
if (dog == 0 )
```

```
    cow = cat + 1;
```

```
else
```

```
    cow = goat + 1;
```

TuringsCraft Assignment

- Answer at least 34 of the 62 questions in sections 5.1 – 5.5 of the TuringsCraft tutoring system
- You will earn 3 points for each correct answer
- Due midnight on Wednesday, October 2, 2013