

# CS 302: Introduction to Programming

## Chapter 3

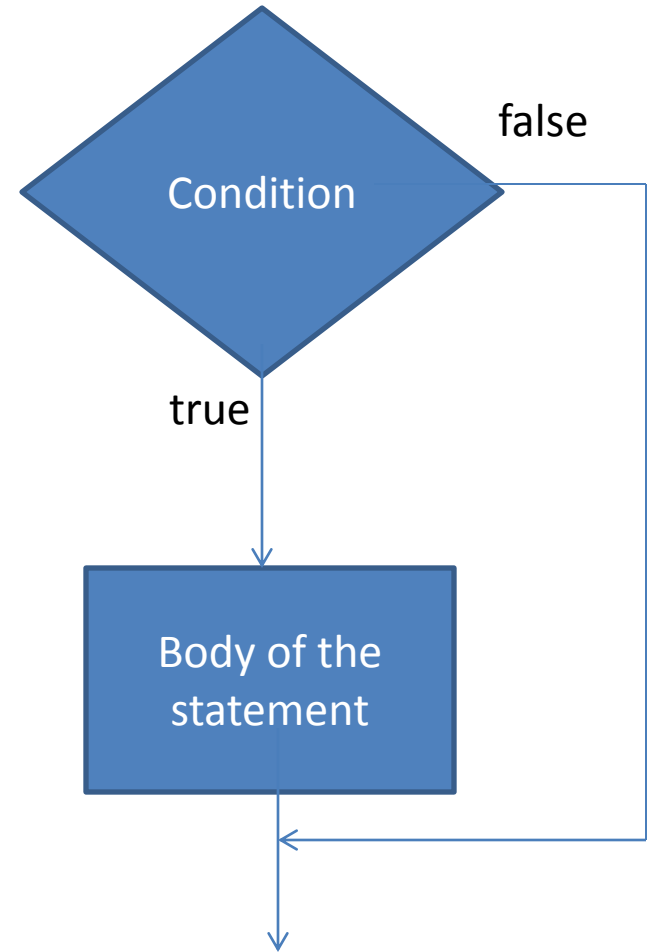
# if statements

Sequenced flow-lines of code executed in sequence

(ex: code we have previously written, without loops or if statements)

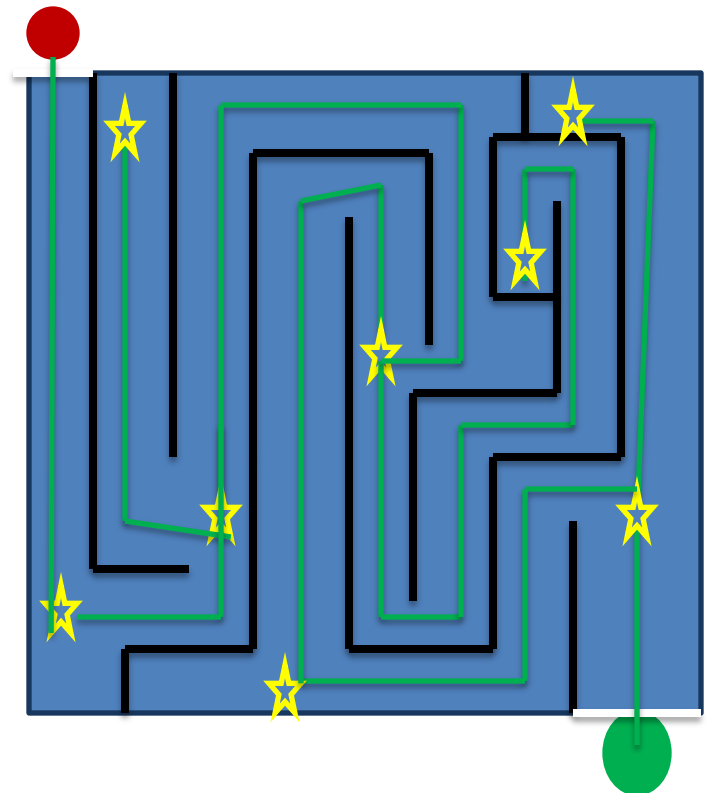
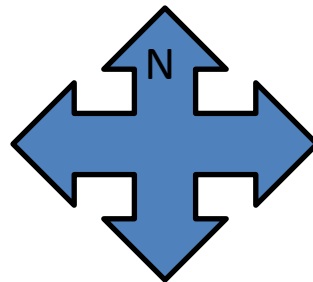
Controlled flow-some statements are executed only when conditions are met

(ex: if statements)



# Motivation for if statements

- Maze solving algorithm: always turn right  
if you are at an intersection, turn right  
if you are facing NORTH, turn EAST  
if you are facing EAST, turn SOUTH  
if you are facing SOUTH, turn WEST  
if you are facing WEST, turn NORTH



# single if statements: “one or none”

- Syntax

```
if ( CONDITION )  
{  
    CODE GOES HERE;  
}  
MORE CODE HERE;
```

## Relational operators:

>, <, >=, <=, ==, !=

Algebraic operator	Java operator	Java example	What it means
=	=	x==y	x is equal to y
≠	!=	x != y	x is not equal to y
>	>	x>y	x is greater than y
<	<	x<y	x is less than y
≥	>=	x>=y	x is greater than or equal to y
≤	<=	x<=y	x is less than or equal to y

- Execution:

- “One or none”

- Either the condition is true, in which case we execute the body of the loop, or the condition is false, in which case we jump to the code that follows the loop.

- Example 1

```
int heightOfChild = 56; //inches  
int MIN_HEIGHT = 48; //inches  
boolean mayRide = false;  
...  
if (heightOfChild >= MIN_HEIGHT)  
{  
    mayRide = true;  
}
```

# if statements

- Example 2

```
Scanner in = new Scanner(System.in);
```

```
System.out.println("Please enter withdrawl amount: ");
```

```
if ( in.hasNextDouble())
```

```
{
```

```
    double withdrawl = in.nextDouble();
```

```
    if (withdrawl > balance)
```

```
    {
```

```
        System.out.println("You may not withdraw more money than  
        you have in your account. You may withdraw only: " +  
        balance);
```

```
        withdrawl = balance;
```

```
    }//end if
```

```
    balance = balance – withdrawl;
```

```
    System.out.printf("Your new balance is: %d", balance);
```

```
}//end if
```

# If statements

- Examples 3 and 4

- You can perform arithmetic inside the if statement

```
int nuPoints = 176; //out of 212 for the semester
```

```
//if you have at least 90 percent
```

```
if (nuPoints / 212 >= 90)
```

```
{
```

```
    System.out.println("You get an A");
```

```
}
```

- You can have method calls inside the if statement

```
String name ; //set elsewhere
```

```
if ( name.length() > 10 )
```

```
{
```

```
    System.out.println("You have a longer than average name.");
```

```
}
```

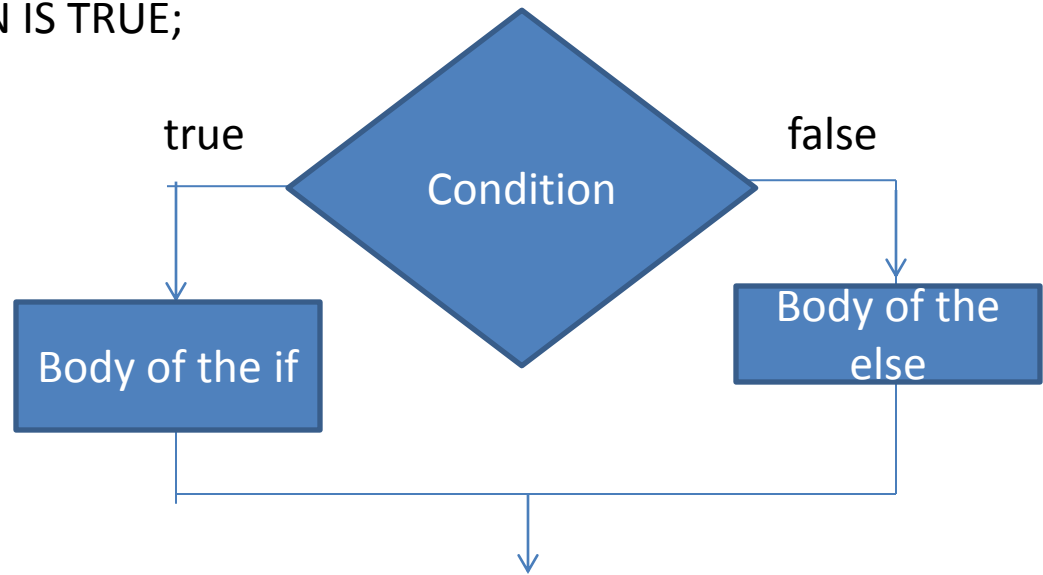
# if/else statements: “one or another”

- Syntax

```
if (CONDITION)
{
    DO THIS WHEN CONDITION IS TRUE;
}
else
{
    DO THIS OTHERWISE;
}
```

- Example 1

```
if (heightOfChild >= MIN_HEIGHT)
{
    //allow child to ride
}
else
{
    //send child to area of the park designed for younger children
}
```



## if/else statements

- Example 2

```
int RED = 1;
int BLUE = 2;
int currentPlayer; //set elsewhere in code
int bluePoints = 0;
int redPoints = 0;
...
Random rng = new Random();
int diceRoll = rng.nextInt(6) + 1; //possible to roll 1,2,3,4,5 or 6

if (diceRoll == 6) //earn points for rolling a 6
{
    if (currentPlayer == RED)
    {
        redPoints++;
    } //end if
    else
    {
        bluePoints++;
    } //end else
} //end if
```

- Example 3: We are scheduling appointments which all run an integer number of hours and begin on the hour. We will use military time (hours 0 if/else statements to 24). Set a variable to determine if two appointments overlap or not.

Assume we are given integer values start1, start2, end1 and end2 for the two appointments.

```
boolean appointmentsOverlap = false;
int laterStartTime;
int earlierEndTime;

if( start1 > start2)
    laterStartTime = start1;
else
    laterStartTime = start2;

if(end1 < end2)
    earlierEndTime = end1;
else
    earlierEndTime = end2;

if( laterStartTime < earlierEndTime)
    appointmentsOverlap = true;
else
    appointmentsOverlap = false;
```

Trace with one appointment from 10-12 and one from 11-13.

start1 = 10, start2=11; end1=12, end2=13

start1>start2 evaluates to false  
laterStartTime set to 11

end1<end2 evaluates to true  
earlierEndTime set to 12

laterStartTime < earlierEndTime evaluates to true  
therefore appointments overlap.  
This is what we expected

# In Class Practice with if/else statements

- Given a String variable, I want to know what the middle character is. For an even string, return the two middle characters.

String word; //already provided

a	l	p	h	a	b	e	t
0	1	2	3	4	5	6	7

```
if(word.length() %2 == 0)
{
    //then the word is even
    System.out.println("Middle two characters : " + word.charAt( word.length()/2) +
        " and " + word.charAt( word.length()/2 -1) );
}
else
{
    //then the word is odd
    System.out.println("Middle character: " + word.charAt( word.length() / 2 ));
}
```

c	r	a	t	e
0	1	2	3	4

## if/else statements

- Use Whitespace Correctly

```
if (x>5)
    if (y>5)
        System.out.println("x and y are greater than 5");
else
    System.out.println("x is less than or equal to 5");
```

- The above is incorrect. The else belongs with the if it follows. To make it clear what is happening here, this should be written with correct indentation and braces:

```
if (x>5) {
    if (y>5) {
        System.out.println("x and y are greater than 5");
    }
    else {
        System.out.println("x is less than or equal to 5");
    }
}
//end else
}
//end if
```

- To make the code do what you intended, add the braces:

```
if (x>5)
{
    if (y>5)
        System.out.println("x and y are greater than 5");
}
else
    System.out.println("x is less than or equal to 5");
```

- Syntax

## Multiple if/else statements: “one of many”

```
if(CONDITION_1)
{
}
else if (CONDITION_2)
{
} //more else/if statements are allowed if desired
else
{
}
```

- Execution

- Check CONDITION\_1

- true: execute body of if statement, then jump to outside the if-else-if

- false: check CONDITION\_2

- true: execute body of this if, jump to outside block

- false: go to the else and execute body of statement before moving out of the block.

- Summary: subsequent if/else and else statements in this block are only checked if the ones preceding them were false. Once a true condition is found, the body of that if/else is executed and the remaining if/else parts of the block are ignored.

# Multiple if/else statements

- Example

Assume that we are running an ice cream stand and a customer may choose vanilla (VAN), chocolate (CHOC), or strawberry (STRAW) ice cream.

```
if (choice == CHOC)
{
    //give them chocolate ice cream
}
else if (choice ==VAN)
{
    //give them vanilla
}
else if (choice == STRAW)
{
    //give them strawberry
}
else
{
    //they made no choice so they don't get any icecream
}
```

# Multiple if/else statements

- Suppose we are organizing data from a race in which two competitors completed the 100 meter race in time1 and time2 respectively. How do we determine the winner?

```
//String person1, String person2, double time1, and double time2 already exist
```

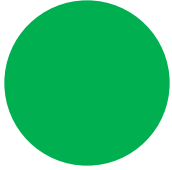
```
String winner = "";  
if(time1 < time2)  
{  
    winner = person1;  
}  
else if (time2 < time1)  
{  
    winner = person2;  
}  
else //they ran in exactly the same amount of time  
{  
    winner = "Tie between " + person1 + " and " + person2;  
}
```

# Multiple if/else statements

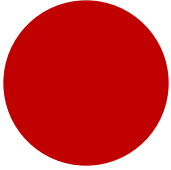
```
//find which spot to take inside the maze solving algorithm
if( isIntersection(currentRow,currentColumn))
{
    System.out.println("intersection");
    if( isLeftTurnAvailable(currentRow,currentColumn) )
    {
        turnLeft(); //method that exists elsewhere in the code
        stepForward();
        System.out.println("turn L, step");
    }
    else if (isStraightAvailable(currentRow,currentColumn))
    {
        stepForward();
        System.out.println("step");
    }
    else
    {
        turnRight();
        stepForward();
        System.out.println("turn R, step");
    }
}
```

# Order of statements

```
if (grade >= 90)
    //assign grade of A
else if (grade >=80)
    //grade of B
else if( grade >=70)
    //grade of C
else if(grade >=60)
    //grade of D
```



```
if (grade >= 60)
    //assign grade of D
else if (grade >=70)
    //grade of C
else if( grade >=80)
    //grade of B
else if(grade >=90)
    //grade of A
```



Which of the above is correct?

Suppose Karen scored 62% in the course.

Fails the first three if statements on the right, earns a D

Passes the first statement on the left, earns a D

Suppose Rachel scored 96% in the course.

Passes the first statement on the left, earns an A

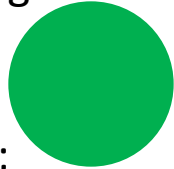
96>60, so passes first statement on the right, earns a D. No more are considered.

Anything > 90 is by definition also >60 so the ordering on the right is incorrect.

# if-else-if statements compared to multiple if statements

//assume int facing, NORTH, SOUTH, EAST, and WEST are set elsewhere in code,  
//this would be in the code section where we have decided to turn left already  
//in our maze solving algorithm.

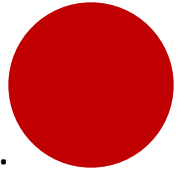
```
if (facing==NORTH)
    facing = EAST;
else if(facing == EAST)
    facing = SOUTH;
else if(facing ==SOUTH)
    facing =WEST;
else if(facing==WEST)
    facing =NORTH;
```



facing = NORTH

facing = EAST

```
if (facing==NORTH)
    facing = EAST;
if(facing == EAST)
    facing = SOUTH;
if(facing ==SOUTH)
    facing =WEST;
if(facing==WEST)
    facing =NORTH;
```



facing = NORTH

facing = EAST

facing = SOUTH

facing = WEST

facing = NORTH

# switch statement

- Syntax:

```
switch ( VARIABLE )
{
    case CONDITION1:
        //what to do if variable = condition1;
        break; // jump to end of the switch statement
    case CONDITION2:
        //what to do if variable = condition2;
        break;
    ....//more cases
    default:
        //what to do if none of the above conditions are met
        break;
}
```

- VARIABLE must be of type int or char

# switch statement: example 1

- Problem: return the English word for each numeric digit

```
public String getWord(int number)
{
    switch(number)
    {
        case 0:
            return "zero";
            break;
        case 1:
            return "one";
            break;
        case 2:
            return "two";
            break;
        case 3:
            return "three";
            break;
        case 4:
            return "four";
            break;
        case 5:
            return "five";
            break;
        case 6:
            return "six";
            break;
        case 7:
            return "seven";
            break;
        case 8:
            return "eight";
            break;
        case 9:
            return "nine";
            break;
    } //end switch
} //end method
```

## switch statement: example 2

- Our maze algorithm used an integer for the direction, so we can use a switch statement.
- Note that constants (as long as they are ints or chars) can be the cases.

Switch (facing)

```
{
    case NORTH:
        facing = EAST;
        break;
    case SOUTH:
        facing = WEST;
        break;
    case EAST:
        facing = SOUTH;
        break;
    case WEST:
        facing = NORTH;
        break;
    default:
        System.out.println("Error: your robot is not facing an allowed direction");
        break;
}
```

## switch statement : missing break statements causes errors

```
switch( score )
{
    case 50:
        System.out.println("You are halfway there!");
        break;
    case 98:
        System.out.println("You are almost done!");
    case 100:
        System.out.println("You win!");
        break;
}
```

- score = 50, output to console:  
You are halfway there!

- Score = 98, output to console:  
You are almost done!  
You win!

## switch statement : intentional fall through

```
for (int d=1; d<=12; d++)
{
    System.out.printf("On the %s day of Christmas my true love gave to me
        \n", getWord(d));
    switch(d)
    {
        case 12: System.out.println(" 12 maids a milking");
        case 11: System.out.println(" 11 of something else");
        case 10: System.out.println(" 10 of something");
        case 9: System.out.println(" 9 ladies dancing");
        case 8: System.out.println(" 8 pipers piping");
        case 7: System.out.println(" 7 swans a swimming ");
        case 6: System.out.println(" 6 geese a laying ");
        case 5: System.out.println(" 5 gold rings and ");
        case 4: System.out.println(" 4 calling birds ");
        case 3: System.out.println(" 3 french hens");
        case 2: System.out.println(" 2 turtle doves and");
        case 1: System.out.println(" A partridge in a pear tree");

        }//end switch

        System.out.println();
    }//end for
```

On the First day of Christmas my true love gave to me

A partridge in a pear tree

On the Second day of Christmas my true love gave to me

2 turtle doves and

A partridge in a pear tree

On the Fifth day of Christmas my true love gave to me

5 gold rings and

4 calling birds

3 french hens

2 turtle doves and

A partridge in a pear tree

On the Twelfth day of Christmas my true love gave to me

12 maids a milking

11 of something else

10 of something

9 ladies dancing

8 pipers piping

7 swans a swimming

6 geese a laying

5 gold rings and

4 calling birds

3 french hens

2 turtle doves and

A partridge in a pear tree

Output to console  
(with some lines  
omitted)

## switch statement : translate if/else to switch

```
int choice; //set earlier

if (choice == CHOC)
{
    System.out.println("chocolate");
}
else if (choice ==VAN)
{
    System.out.println("vanilla");
}
else if (choice == STRAW)
{
    System.out.println("strawberry");
}
else
{
    System.out.println("No ice
        cream");
}
}
```

```
int choice;

switch (choice)
{
    case CHOC:
        System.out.println("chocolate");
        break;
    case VAN:
        System.out.println("vanilla");
        break;
    case STRAW:
        System.out.println("strawberry");
        break;
    default:
        System.out.println("No ice cream");
        break;
}
```

# switch statement : key points

- Only used with int or char
- Variable checked against multiple cases
- default executes when none of the cases are met
- Use instead of multiple if-else-if statements when appropriate
- Break statement needed to prevent fall through

# Conditional Operator

- Set a variable based on a condition

- Syntax:

```
VARIABLE = CONDITION ? VALUE_1 : VALUE_2 ;  
    //if condition is true, variable is set to value1  
    //if condition is false, variable is set to value2
```

- Example:

```
withdral = ( withdrawl > balance ) ? balance : withdrawl ;
```

- Equivalent code:

```
if ( withdral > balance)  
{  
    withdrawl = balance;  
}  
else  
{  
    //withdrawl = withdrawl;  
}
```

# Conditional Operator (cont.)

```
VARIABLE = CONDITION ? VALUE_1 : VALUE_2 ;  
    //if condition is true, variable is set to value1  
    //if condition is false, variable is set to value2
```

```
withdral = ( withdrawl > balance ) ? balance : withdrawl ;
```

```
System.out.println( studentGrade >= 60 ? "Passed" : "Failed" );
```

```
//assume int a, int b already exist  
int max = (a > b) ? a : b;
```

# Comparisons with Strings

- Equality

```
String myChoir = "Kantori"; //set after an audition
```

```
String choirIWant = "Kantori";
```

```
if(choirIWant == myChoir)
```

```
{
```

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

```
}
```

- The "==" operator checks for identical objects, not identical String literals
- The if statement will evaluate to false

```
if (choirIWant == "Kantori") { }
```

- choirIWant was declared to be exactly the String literal "Kantori" thus both strings point to the same memory location and this will evaluate to true.

# Comparisons with Strings

- Equality

- Do the following evaluate to true or false?

```
String name = "Keladry";
```

```
String nickname = name.substring(0,3);
```

```
if (name == "Keladry") {}
```

- true

```
if (nickname == "Kel") {}
```

- nickname is set to the literal "Kel"
- false, different objects

- Compare String literals with .equals()

```
if(choirIWant.equals(myChoir) ) //returns true
```

```
if(!choirIWant.equals(myChoir) ) //returns false
```

```
if (nickname.equals("Kel") { } //returns true
```

A String object

A String method

“.” indicates call a method of this object

# Comparisons with Strings

- Inequalities (for numbers: >, <, >=, <=)
  - Lexicographic ordering
    - Dictionary ordering
    - Capital letters occur below lowercase (Z precedes a)
    - Spaces precede letters or symbols
    - Numbers precede letters
    - Punctuation marks have a specific ordering (see Appendix A)
  - Use String method compareTo()
    - Returns an integer
      - <0 (often -1)      string1 comes first
      - >0 (often 1)      string2 comes first
      - 0                      string1 and string2 are equal
  - Example

```
String animal1 = "cat";  
String animal2 = "zebra";
```

```
animal1.compareTo(animal2); //returns a positive integer
```

# Comparisons with Floating Point Numbers

- Two numbers may be different in the 100<sup>th</sup> or 1000<sup>th</sup> digit. The “==” operator will return false, but we may consider them close enough to be equal.
- Check that difference is less than epsilon ( $\epsilon$ ), usually set to  $1 \times 10^{-14}$

- Example

```
final double EPSILON = 1E-14;
final double r = Math.sqrt(2.0);

double rSq = r*r; // 2.000000000000000044

if (rSq == 2.0) //false

if (Math.abs(rSq - 2.0) < EPSILON) //true
```

# Multiple Checks

- Nested branches

- Pseudocode Example: High School students may take a course that is offered first and third hour only if they have that period free. Also, you must be a senior to take this class unless you have special permission.

```
boolean mayRegister = false;
```

```
if(student wants to register for class)
{
    if (student is a senior)
    {
        if (student has first hour free)
            mayRegister = true;
        else if (student has third hour free)
            mayRegister = true;
    } //end if is a senior
    else //student is not a senior
    {
        if (student has special permission)
            mayRegister = true;
    } //end else not a senior
} //end if wants to register
```

Remember that else is always matched with the if that precedes it, unless braces indicate otherwise.

# Nested Branches

- On a chess board, each square has a name that contains a letter and a number. Suppose we need to know what color the square is.

String currentSquare; //of the form "a7" or "d3"

```
char letter = currentSquare.charAt(0);  
int number = Integer.parseInt( currentSquare.substring(1) );
```

```
String color = "";  
if( letter == 'a' || letter == 'c' || letter == 'e' || letter == 'g' )
```

```
{  
    if(number % 2 == 0)  
        color = "white"; //braces omitted due to space
```

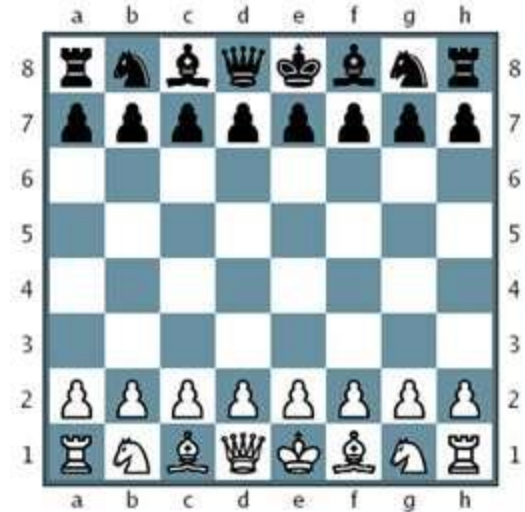
```
    else  
        color = "black";
```

```
}
```

```
else  
{  
    if(number % 2 == 0)  
        color = "black";
```

```
    else  
        color = "white";
```

```
}
```



Trace for g5

-first if statement evaluates to true

- $5 \% 2 == 1$  so second if is false and move to the else

-result is "black"

Trace for d7

-first if statement evaluates to false so move to else

- $7 \% 2 == 1$  so second if is false and move to the else

-result is "white"

# Multiple Checks

- Boolean logic

- AND

- Both conditions must be true

<b>A</b>	<b>B</b>	<b>A and B</b>
true	true	true
true	false	false
false	true	false
false	false	false

- Java symbol: &&

- Example 1:

```
if (age > 18 && citizenship.equals("American"))  
    mayVote = true;
```

- Example 2:

```
if ( x > 10 && x < 100)           //java version of 10<x<100
```

# Multiple Checks

- Boolean logic

- OR

- One condition must be true

<b>A</b>	<b>B</b>	<b>A or B</b>
true	true	true
true	false	true
false	true	true
false	false	false

- Java symbol: ||

- Example:

```
if (apCredit || passedCollegeClass)
    fulfilledRequirement= true;
```

# Multiple Checks

- Boolean Logic

- Note that you must write the entire expression for each of the checks

- ```
if( response == 'Q' || 'q' ) // WRONG
```

- ```
if( response == 'Q' || response == 'q') //correct
```

- ```
if( x<90 && >=80) //WRONG
```

- ```
    score = "B";
```

- ```
if( x<90 && x>=80) //correct
```

- ```
    score = "B";
```

- Lazy Evaluation

- Conditions checked left to right; stop checking when truth is determined

- Precedence

- && occurs before ||



a)  $(n > -7) \ \&\& \ (n < 3) \ \&\& \ ((n == 5) \ || \ (n == 7))$

b)  $(n > -7) \ \&\& \ (n < 3) \ \ || \ (n == 5) \ \ || \ (n == 7)$

c)  $(n > -7) \ \ || \ (n < 3) \ \ || \ (n == 5) \ \ || \ (n == 7)$

d)  $((n > -7) \ || \ (n < 3)) \ \&\& \ (n == 5) \ \&\& \ (n == 7)$



# Multiple Checks

- Boolean logic

- NOT

- Negate the condition

<b>A</b>	<b>!A</b>
true	false
false	true

- Java symbol: !

- Example:

```
if( !(number % 5 == 0) )  
    isMultipleOfFive = false;
```

# Multiple Checks

- Boolean logic
  - DeMorgan's Law

$$\!(A \ || \ B) = \!A \ \&\& \ \!B$$

$$\!(A \ \&\& \ B) = \!A \ || \ \!B$$

- Simplify where possible

$\!(x > y)$  becomes  $x \leq y$

$\!(x = y)$  becomes  $x \neq y$

A	B	A    B	!(A    B)
false	false	false	true
false	true	true	false
true	false	true	false
true	true	true	false

A	B	!A	!B	!A && !B
false	false	true	true	true
false	true	true	false	false
true	false	false	true	false
true	true	false	false	false



•Other logic rules

$$A \ || \ B = B \ || \ A,$$

$$A \ || \ \text{true} = \text{true},$$

$$A \ || \ !A = \text{true},$$

$$A \ || \ A = A,$$

$$A \ \&\& \ B = B \ \&\& \ A$$

$$A \ \&\& \ \text{false} = \text{false}$$

$$A \ \&\& \ !A = \text{false}$$

$$A \ \&\& \ A = A$$

$$A \ || \ (B \ || \ C) = (A \ || \ B) \ || \ C$$

$$A \ \&\& \ (B \ \&\& \ C) = (A \ \&\& \ B) \ \&\& \ C$$

$$A \ \&\& \ (B \ || \ C) = (A \ || \ B) \ \&\& \ (A \ || \ C)$$

$$A \ || \ (B \ \&\& \ C) = (A \ || \ B) \ \&\& \ (A \ || \ C)$$

•Example 1

Given integer variables x and y, set a boolean when x is larger than y, with neither of them negative?

```
if(x > y && x > 0 && y > 0)
    boolean conditionsMet = true;
```


•Example 2

Consider the following expression:

```
!(n > 11 && n <= 22)
```

Which of these is equivalent?

a)  $n < 11 \ || \ n \geq 12$

b)  $n \leq 11 \ || \ n > 22$  

c)  $n > 11 \ || \ n \leq 22$

d)  $n < 11 \ \&\& \ n \geq 22$

e)  $n \leq 11 \ \&\& \ n > 22$

```
!(n > 11) || !(n <= 22)
```

```
n <= 11 || n > 22
```

•Does the given month have 30 days?

```
if(month.equals("January"))
    System.out.println("no");
if(month.equals("February"))
    System.out.println("no");
if(month.equals("March"))
    System.out.println("no");
if(month.equals("April"))
    System.out.println("yes");
if(month.equals("May"))
    System.out.println("no");
if(month.equals("June"))
    System.out.println("yes");
if(month.equals("July"))
    System.out.println("no");
if(month.equals("August"))
    System.out.println("no");
if(month.equals("September"))
    System.out.println("yes");
if(month.equals("October"))
    System.out.println("no");
if(month.equals("November"))
    System.out.println("yes");
if(month.equals("December"))
    System.out.println("no");
```

Compare if, if-else, and boolean operations

What happens if the month we are asking for is January?

Check all 12 months

```
if(month.equals("January"))
    System.out.println("no");
else if(month.equals("February"))
    System.out.println("no");
else if(month.equals("March"))
    System.out.println("no");
else if(month.equals("April"))
    System.out.println("yes");
else if(month.equals("May"))
    System.out.println("no");
else if(month.equals("June"))
    System.out.println("yes");
else if(month.equals("July"))
    System.out.println("no");
else if(month.equals("August"))
    System.out.println("no");
else if(month.equals("September"))
    System.out.println("yes");
else if(month.equals("October"))
    System.out.println("no");
else if(month.equals("November"))
    System.out.println("yes");
else if(month.equals("December"))
    System.out.println("no");
```

Compare if, if-else, and  
boolean operations

What happens if the  
month we are asking for is  
January?

Check only the first month,  
then jump over other  
checks since true was  
found.

Code is more efficient.

```
else if(month.equals("April"))
    System.out.println("yes");
else if(month.equals("June"))
    System.out.println("yes");
else if(month.equals("September"))
    System.out.println("yes");
else if(month.equals("November"))
    System.out.println("yes");
else
    System.out.println("no");
```

Compare if, if-else, and  
boolean operations

Code is now shorter.

```
if( month.equals("September") ||  
    month.equals("April")    || month.equals("June")  
    || month.equals("November"))  
{  
    System.out.println("yes");  
}  
else  
{  
    System.out.println("no");  
}
```

Compare if, if-else, and  
boolean operations

# Boolean Variables

- Flag variables
  - Used to indicate whether or not a program should continue on
  - Example: `gameOver`

- Note for boolean variables

Type:

```
if(!failedTest)
    receiveCertification = true;
```

Rather than:

```
if(failedTest == false)
    receiveCertification = true;
```

# Input Validation with if Statements

- Example: suppose you ask the user to choose how many credits they wish to take a course for. The only options are 1,2, or 3.

```
int nuCredits; //set already
if (nuCredits >=1 && nuCredits<=3)
    //allow them to register for this course
else
    //tell them to try again (with a while loop perhaps)
```

- Also check for integer input

```
Scanner in = new Scanner(System.in);
System.out.println("Input number of credits desired (1,2, or 3): ");
```

```
if (in.hasNextInt() )
    int nuCredits = in.nextInt();
else
    //tell them to enter an integer and try again
```

# Input Validation with if Statements

- Combine the two checks with nested branches

```
Scanner in = new Scanner(System.in);
```

```
System.out.println("Input number of credits desired (1,2, or 3): ");
```

```
if (in.hasNextInt() ) {
```

```
    int nuCredits = in.nextInt();
```

```
    if (nuCredits >=1 && nuCredits<=3)
```

```
        //allow them to register for this course
```

```
    else
```

```
        //tell them to try again (with a flag variable set false perhaps)
```

```
}
```

```
else
```

```
    //tell them to enter an integer and try again
```

# Boundary Conditions

- Remember to consider inclusive vs. exclusive boundary conditions

- Example: if you are assigning grades should you have

- A)

- (grade > 90) //exclude 90 percent

- OR

- B)

- (grade >= 90) //include 90 percent





# What is the value of name when this code is run?

```
String name = "";  
int rank = 2; //the nth most common last name
```

```
switch( year )  
{  
case 1:  
    name = "Smith";  
    break;  
case 2:  
    name = "Johnson";  
    break;  
case 3:  
    name = "Wlliams";  
    break;  
default:  
    name = "No data provided.";  
    break;  
}
```

Answer: Johnson

# What is printed when this code is run?

```
int x = 7;

switch( x )
{
case x < 0:
    System.out.println("x is negative");
    break;
case 0:
    System.out.println("x is 0");
    break;
case x > 0:
    System.out.println("x is positive");
    break;
}
```

This code will not compile because you can not use > or < signs in a switch statement.

# What is the value of x after this code is run?

```
int x = 1;
int y = 12;

switch( x )
{
case 10:
    x += 2;
case 12:
    x *= y;
case 14:
    x++;
}
```

In case 12

x set to  $x * y = 1 * 12 = 12$

In case 14

x set to  $x+1 = 12+1 = 13$

Note the missing break statements.

# Do I have to go to work after this code is run?

```
boolean workDay = true;
String day = "Monday";
String WEEKEND_1= "Sunday";
String WEEKEND_2 = "Saturday";
```

```
switch( day )
{
    case WEEKEND_1:
        workDay = false;
        break;
    case WEEKEND_2:
        workDay = false;
        break;
    default:
        workDay = true;
        break;
}
```

This code will not compile because you can not switch on a String variable.

Using constants as the cases (when they are ints or chars) will work.

# If inside a switch

```
int x = 15;  
int z = x % 2;
```

```
switch(z)  
{  
  case 0: //even  
    if ( x % 5 == 0)    {  
      System.out.println("x is an even multiple of 5.");  
    }  
    else    {  
      System.out.println("x is even but not a multiple of 5.");  
    }  
    break;  
  case 1: //odd  
    if ( x % 5 == 0)    {  
      System.out.println("x is an odd multiple of 5.");  
    }  
    else    {  
      System.out.println("x is odd and a multiple of 5.");  
    }  
  }  
}
```

# Choose if or switch statement

- if

- When inequalities are needed
- When multiple conditions must be met
- When the variable to be considered is not an int or a char

- switch

- When the variable to be considered is an int or a char
- When you the variable can be one of a series of values



DEPARTMENT OF  
**Computer Sciences**  
UNIVERSITY OF WISCONSIN-MADISON