September 30, 2013: Announcements

Assignment 3 is due Wednesday (Oct 2nd) at 9:00 pm. Please don't wait until the last few hours to submit and test your code. WebGS cannot handle 400 students submitting at one time.

Assignment 4 is due next Wednesday (Oct 9nd) at 9:00 pm.

Office Hours: 1-3pm on Monday and Wednesday (4 total hours of office hours)

The first exam is October 16th!

Remember to complete your OELs. Log on to the site to check if you have any. If you have trouble, or you don't understand the message you are getting, post a note on Piazza.

Regarding the questions about WebGS: the test cases for Assignment 2 and Assignment 3 are checking for an exact match to output. As has been mentioned on Piazza, we will be doing some grading manually to give partial credit where appropriate.

Comparison Operators (continued):

Let's look at another example:

Write a program that gives suggestions for what someone should wear based on the weather outside

Ask the user for the temperature outside

Evaluate the input: print a message that is appropriate for the data entered

If it's below 70F, tell the user to grab a windbreaker

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

If it's between 85F and 92F, tell the user to go for a swim

If it's above 92F, tell the user to stay in an air-conditioned building

How do we do that with code? First we have to figure out how to decide what to tell the user; we need to figure out what our **`boolean`** expressions are.

Comparison Operators (continued):

Ask the user for the temperature outside

```
Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the temperature: ");
int temp = keyboard.nextInt();
```

Evaluate the input: print a message that is appropriate for the data entered

If it's below 70F, tell the user to grab a windbreaker

```
boolean useWindbreaker = temp < 70;
```

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

I want to say **`71 <= temp < 85`**

If it's between 85F and 92F, tell the user to go for a swim

I want to say **`85 <= temp < 92`**

If it's above 92F, tell the user to stay in an air-conditioned building

```
boolean stayInside = temp > 92;
```


Comparison Operators (continued):

Logical Operators (continued):

The **`&&`** operator:

Performs an AND operation.

Has two operands.

Returns **`true`** if both operands are **`true`**; otherwise, **`false`** is returned.

Example:

```
int age;
age = inputScan.nextInt();

boolean teenAge;

teenAge = (age >= 13) && (age <= 19);

System.out.println("teenAge is " + teenAge);

teenAge = (age > 12) && (age < 20);
```

Comparison Operators (continued):

Logical Operators (continued):

The **&&** operator:

Performs an AND operation.

Has two operands.

Returns **true** if both operands are **true**; otherwise, **false** is returned.

Example:

```
int age;
age = inputScan.nextInt();

boolean teenAge, oldEnough, youngEnough;

oldEnough = age >= 13;
youngEnough = age <= 19;

teenAge = largeEnough && youngEnough;
```


Comparison Operators (continued):

Logical Operators (continued):

The **||** operator:

Performs an OR operation.

Has two operands.

There are two ways to think about what the OR operator does:

Returns **true** if either or both operands are **true**; otherwise, **false** is returned.

Returns **false** if both operands are **false**; otherwise, **true** is returned.

Comparison Operators (continued):

Logical Operators (continued):

The || operator:

Example:

```
int myAge;
myAge = inputScan.nextInt();

int sisterAge = 34, brotherAge = 39;

boolean notYoungest;

notYoungest = (myAge > brotherAge) || (myAge >
sisterAge);

System.out.println("notYoungest is " + notYoungest);
```


Comparison Operators (continued):

Logical Operators (continued):

Summary in the form of a *truth table*:

a	b	!a	a && b	a || b
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

Comparison Operators (continued):

Logical Operators (continued):

So, back to that problem… We want to identify a value falling in a range

Examples: telling the user what to wear, identifying liquid water, even that teenager problem from a few slides ago!

Let's look at the liquid water problem…

It is tempting to write this as:

```
double waterTemp;
waterTemp = keyboard.nextDouble();
boolean liquidWater = 0.0 <= waterTemp <= 100.0;
```

The last line produces the error:

"`operator <= cannot be applied to boolean,float`"

Comparison Operators (continued):

Logical Operators (continued):

```
// this will cause an error...
liquidWater = 0.0 <= waterTemp <= 100.0;
```

What is the right way to make this range comparison?

```
liquidWater = (0.0 <= waterTemp) && (waterTemp <= 100.0);
```

How would we calculate the non-liquid case?

```
boolean iceOrSteam;
iceOrSteam = !( (0.0 <= waterTemp) &&
                (waterTemp <= 100.0) );
iceOrSteam = !liquidWater;
```

Comparison Operators (continued):

Logical Operators (continued):

DeMorgan's Laws:

1. NOT(A AND B) = (NOT A) OR (NOT B)
2. NOT(A OR B) = (NOT A) AND (NOT B)

Consider the teenage example from before:

```
teenAge = (age >= 13) && (age <= 19);
```

Ask the question: is the age not a teenager?

```
notTeenAge = !( (age >= 13) && (age <= 19) );
```

Apply the first of DeMorgan's Laws:

```
notTeenAge = !(age >= 13) || !(age <= 19);
```


Comparison Operators (continued):

Logical Operators (continued):

From the previous slide:

```
notTeenAge = !(age >= 13) || !(age <= 19);
```

This can be simplified:

Consider: `!(age >= 13)`

How can this be written without the NOT?

```
age < 13
```

Do the same for `!(age <= 19)`

```
age > 19
```

The final result:

```
notTeenAge = (age < 13) || (age > 19);
```

Comparison Operators (continued):

Logical Operators (continued):

One more look at this. The following statements are all equivalent:

```
notTeenAge = !( (age >= 13) && (age <= 19) );
notTeenAge = !(age >= 13) || !(age <= 19);
notTeenAge = (age < 13) || (age > 19);
```

As are these statements:

```
teenAge = (age >= 13) && (age <= 19);
teenAge = !( (age < 13) || (age > 19) );
teenAge = !(age < 13) && !(age > 19);
```


Comparison Operators (continued):

Let's go back to this example:

Write a program that gives suggestions for what someone should wear based on the weather outside

Ask the user for the temperature outside

Evaluate the input: print a message that is appropriate for the data entered

If it's below 70F, tell the user to grab a windbreaker

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

If it's between 85F and 92F, tell the user to go for a swim

If it's above 92F, tell the user to stay in an air-conditioned building

How do we do that with code? First we have to figure out how to decide what to tell the user; we need to figure out what our **`boolean`** expressions are.

Comparison Operators (continued):

Ask the user for the temperature outside

```
Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the temperature: ");
int temp = keyboard.nextInt();
```

Evaluate the input: print a message that is appropriate for the data entered

If it's below 70F, tell the user to grab a windbreaker

```
boolean useWindbreaker = temp < 70;
```

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

```
boolean wearShorts = (71 <= temp && temp < 85);
```

If it's between 85F and 92F, tell the user to go for a swim

```
boolean goSwimming = (85 <= temp && temp < 92);
```

If it's above 92F, tell the user to stay in an air-conditioned building

```
boolean stayInside = temp > 92;
```


Comparison Operators (continued):

So, now that I have my **`boolean`** conditions, how do I check them?

Let's look at everything we've learned about so far:

We have **`if`** statements, so we could make a separate if statement for each one of the temperature ranges

We have **`if/else`** statements, but that was really for when we have two clauses

If they have 15 items or less, tell the user to go to the checkout line

If they have more than 15 items, tell the user to go in the normal line

We need to do something different for each temperature range, but we have more than just then just the two situations.

What if we could say "if we don't satisfy the first situation, we need to check something different to see if we satisfy the next situation", and continue that until we're out of situations to check?

`if/else if` Statements

Reading: Section 5.4

Used when data falls into multiple mutually exclusive categories.

Want program to do different things for each category.

Temperature example:

If it's below 70F, tell the user to grab a windbreaker

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

If it's between 85F and 92F, tell the user to go for a swim

If it's above 92F, tell the user to stay in an air-conditioned building

For any given temperature, we're only going to fall into one category, so we can (and should) use and **`if/else`** statement here!

`if/else if` Statements (Continued):

Temperature example:

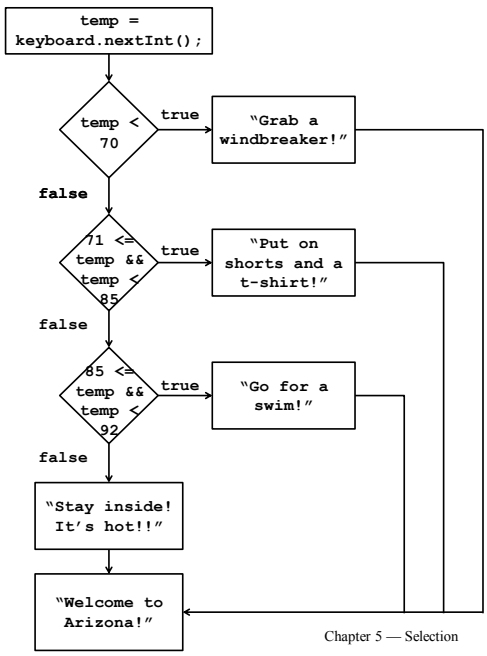
Before we look at the code, let's look at the flow chart

If it's below 70F, tell the user to grab a windbreaker

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

If it's between 85F and 92F, tell the user to go for a swim

If it's above 92F, tell the user to stay in an air-conditioned building



`if/else if` Statements (Continued):

And those clauses match our **`boolean`** conditions!

- If it's below 70F, tell the user to grab a windbreaker

  `(temp < 70)`

- If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

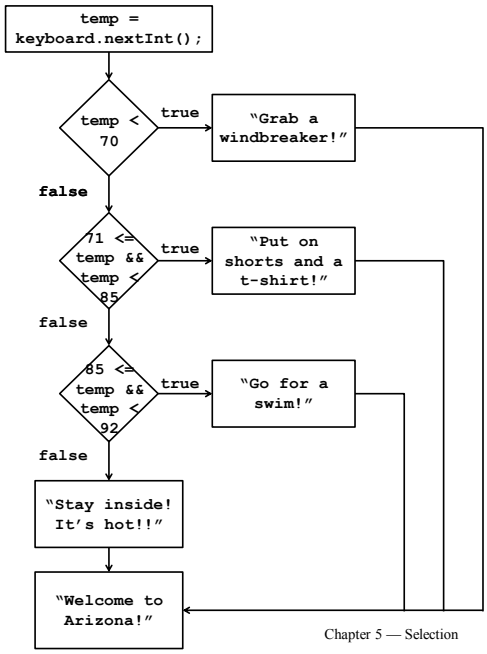
  `(temp >= 70) && (temp < 85)`

- If it's between 85F and 92F, tell the user to go for a swim

  `(temp >= 85) && (temp < 92)`

- If it's above 92F, tell the user to stay in an air-conditioned building

  `(temp >= 92)`



`if/else if` Statements (Continued):

But do we really need to do all of those checks?

Let's just examine part of the flow chart for now

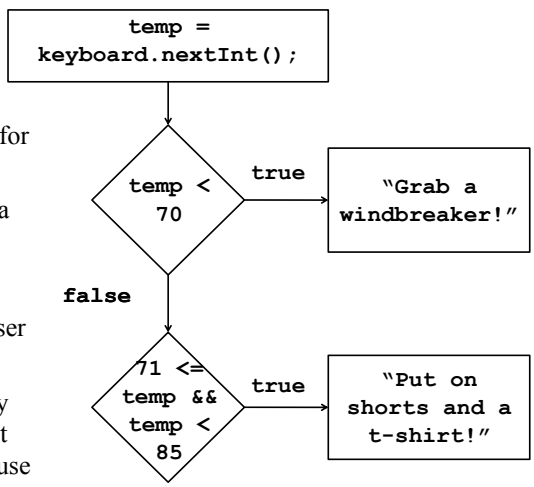
- If it's below 70F, tell the user to grab a windbreaker

  `(temp < 70)`

- If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

  - If it's not below 70, then we already know that the temperature is at least 71, so we don't really need that clause

  `(temp >= 71) && (temp < 85)`



`if/else if` Statements (Continued):

But do we really need to do all of those checks?

Let's just examine part of the flow chart for now

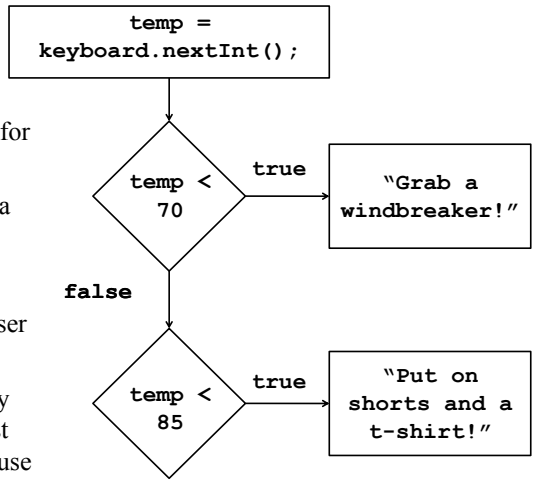
If it's below 70F, tell the user to grab a windbreaker

```
(temp < 70)
```

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

If it's not below 70, then we already know that the temperature is at least 71, so we don't really need that clause

```
(temp < 85)
```



`if/else if` Statements (Continued):

Ok, so we've fixed some of those, but why isn't there a clause for our last option?

If it's below 70F, tell the user to grab a windbreaker

```
(temp < 70)
```

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

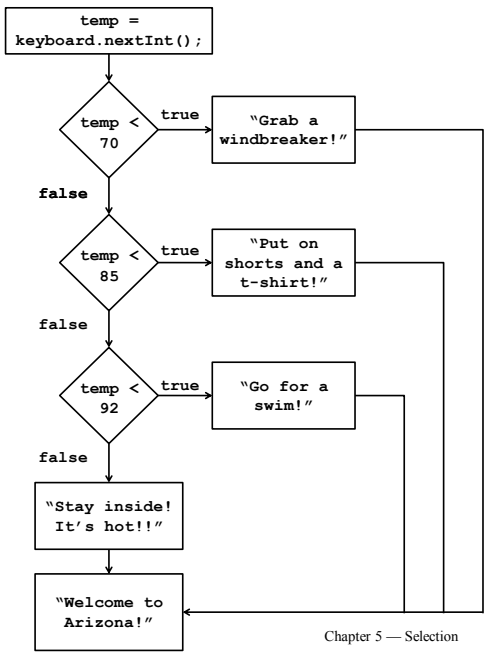
```
(temp < 85)
```

If it's between 85F and 92F, tell the user to go for a swim

```
(temp < 92)
```

If it's above 92F, tell the user to stay in an air-conditioned building

```
(temp >= 92)
```



`if/else if` Statements (Continued):

Ok, so we've fixed some of those, but why isn't there a clause for our last option?

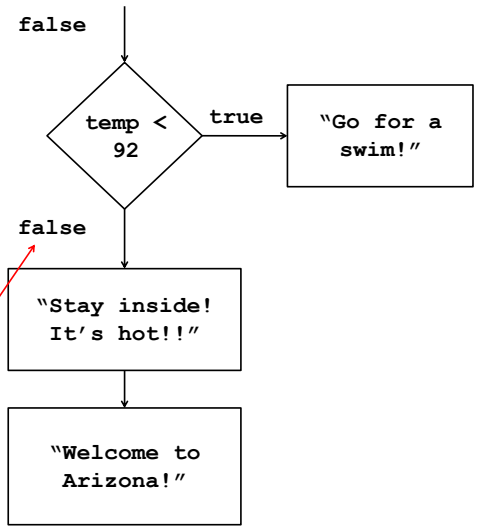
If it's between 85F and 92F, tell the user to go for a swim

`(temp < 92)`

If it's above 92F, tell the user to stay in an air-conditioned building

Well, if the temperature wasn't below 92 degrees, then it's the same as saying that the temperature is at least 92 degrees. This is just like our scenario where we used the **`if/else`**

`(temp >= 92)`



`if/else if` Statements (Continued):

Ok, so we've fixed some of those, but why isn't there a clause for our last option?

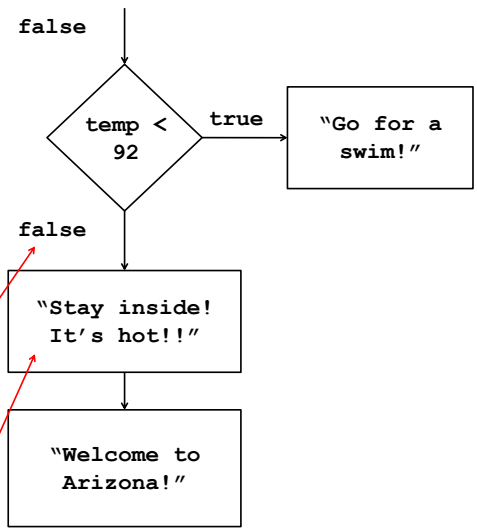
If it's between 85F and 92F, tell the user to go for a swim

`(temp < 92)`

If it's above 92F, tell the user to stay in an air-conditioned building

Well, if the temperature wasn't below 92 degrees, then it's the same as saying that the temperature is at least 92 degrees. This is just like our scenario where we used the **`if/else`**

This correlates with the fact that we don't have an extra diamond up there; we just go to the block for the **`else`**



`if/else if` Statements (Continued):

So, now we've finalized our flow chart. One last look before we translate it to code.

If it's below 70F, tell the user to grab a windbreaker

`(temp < 70)`

If it's between 71F and 85F, tell the user to put on shorts and a t-shirt

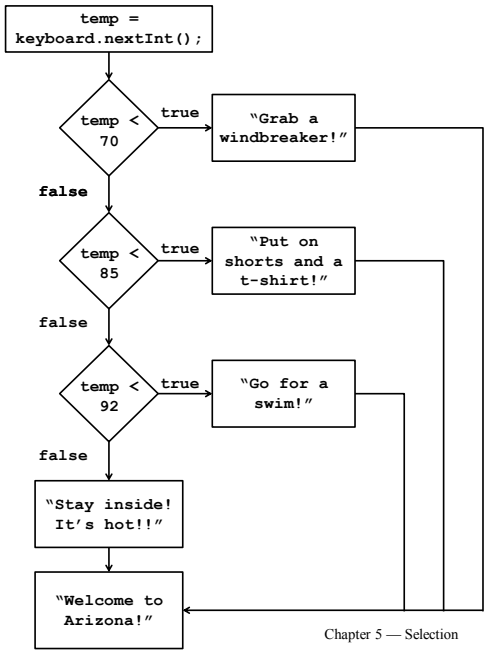
`(temp < 85)`

If it's between 85F and 92F, tell the user to go for a swim

`(temp < 92)`

If it's above 92F, tell the user to stay in an air-conditioned building

`Will be caught by else`



`if/else if` Statements (Continued):

Here's the version using the compound boolean statements:

```
Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the temperature: ");
int temp = keyboard.nextInt();
if ( temp < 70 ) {
   System.out.println("Grab a windbreaker!")
} else if ( 71 <= temp && temp < 85 ) {
   System.out.println("Put on shorts and a t-shirt!");
} else if ( 85 <= temp && temp < 92 ) {
   System.out.println("Go for a swim!");
} else if ( 92 <= temp ) {
   System.out.println("Stay inside!  It's hot!");
}
System.out.println("Welcome to Arizona!");
```

`if/else if` Statements (Continued):

Here's the version using the simplified boolean statements:

```
Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the temperature: ");
int temp = keyboard.nextInt();
if ( temp < 70 ) {
   System.out.println("Grab a windbreaker!")
} else if ( temp < 85 ) {
   System.out.println("Put on shorts and a t-shirt!");
} else if ( temp < 92 ) {
   System.out.println("Go for a swim!");
} else {
   System.out.println("Stay inside!  It's hot!");
}
System.out.println("Welcome to Arizona!");
```


Flow of Control (Continued):

Let's go back to the grocery store example, but with a twist: all the lines are really long, and you have chocolate chunk peanut butter swirl ice cream in your cart

Write a program that would decide which line people should use at the grocery store based on how many items they have in their cart and whether or not they have cold items that they need to get home soon.

Ask the user for how many items they have

Ask the user if there are cold items in the basket

Evaluate the input: print a message that is appropriate for the data entered

If they have 15 items or less, just tell them to use the express lane

If they have cold items, check to see if they have 17 items or less, and suggest that they put a few things back so that they can use the express lane. If they have more than 17 items, then suggest they put the cold items back.

If they don't have cold items, then just tell them to use the regular lane.

Flow of Control (Continued):

Let's see if we can re-write this with distinct (mutually exclusive) categories, because if we can, we just use **`if/else`** statements.

Evaluate the input: print a message that is appropriate for the data entered

- If they have 15 items or less, just tell them to use the express lane.
- If they have cold items, check to see if they have 17 items or less, and suggest that they put a few things back so that they can use the express lane. If they have more than 17 items, then suggest they put the cold items back
- If they don't have cold items, then tell them to use the regular lane

Evaluate the input: print a message that is appropriate for the data entered

- If they have 15 items or less, just tell them to use the express lane
- Otherwise, if they have cold items, and they have 17 items or less, suggest that they put a few things back so that they can use the express lane.
- Otherwise, if they have cold items, and they have more than 17 items, then suggest they put the cold items back.
- Otherwise, if they don't have cold items, then tell them to use the regular lane.

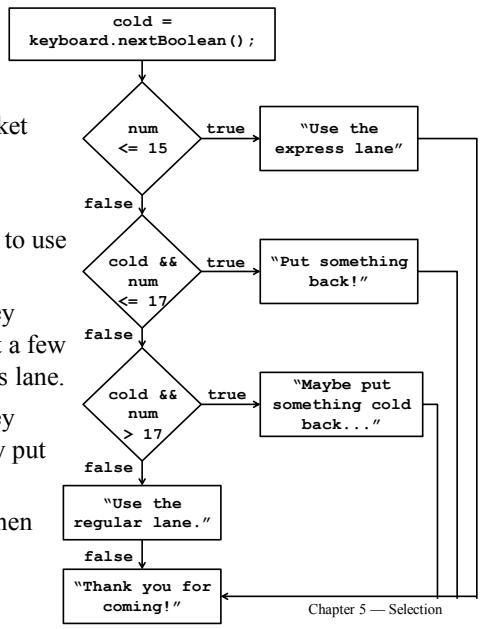
Flow of Control (Continued):

Ask the user for how many items they have

Ask the user if there are cold items in the basket

Evaluate the input: print a message that is appropriate for the data entered

- If they have 15 items or less, just tell them to use the express lane
- Otherwise, if they have cold items, and they have 17 items or less, suggest that they put a few things back so that they can use the express lane.
- Otherwise, if they have cold items, and they have more than 17 items, then suggest they put the cold items back.
- Otherwise, if they don't have cold items, then tell them to use the regular lane.



Flow of Control (continued):

```
System.out.print("Enter number of items you have: ");
int numItems = keyboard.nextInt();
System.out.print("Do you have cold items: ");
boolean coldItems = keyboard.nextBoolean();
if ( numItems <= 15 ) {
  System.out.println("Use the express lane!")
} else if ( coldItems && numItems <= 17) {
  System.out.println("Get those cold things home!");
  System.out.println("See if you can put a few things back.");
} else if ( coldItems && numItems > 17) {
  System.out.println("Consider putting the cold items back.");
} else {
  System.out.println("Use the regular check out line.");
}
System.out.println("Thank you for coming!");
```


Flow of Control (continued):

```
System.out.print("Enter number of items you have: ");
int numItems = keyboard.nextInt();
System.out.print("Do you have cold items: ");
boolean coldItems = keyboard.nextBoolean();
if ( numItems <= 15 ) {
  System.out.println("Use the express lane!")
} else if ( coldItems && numItems <= 17) {
  System.out.println("Get those cold things home!");
  System.out.println("See if you can put a few things back.");
} else if ( coldItems && numItems > 17) {
  System.out.println("Consider putting the cold items back.");
} else {
  System.out.println("Use the regular check out line.");
}
System.out.println("Thank you for coming!");
```

Flow of Control (continued):

```
System.out.print("Enter number of items you have: ");
int numItems = keyboard.nextInt();
System.out.print("Do you have cold items: ");
boolean coldItems = keyboard.nextBoolean();
if ( numItems <= 15 ) {
  System.out.println("Use the express lane!")
} else if ( coldItems && numItems <= 17) {
  System.out.println("Get those cold things home!");
  System.out.println("See if you can put a few things back.");
} else if ( coldItems && numItems > 17) {
  System.out.println("Consider putting the cold items back.");
} else {
  System.out.println("Use the regular check out line.");
}
System.out.println("Thank you for coming!");
```


Flow of Control (Continued):

Ask the user for how many items they have

Ask the user if there are cold items in the basket

Evaluate the input: print a message that is appropriate for the data entered

If they have 15 items or less, just tell them to use the express lane

Otherwise, if they have cold items, do a second check for how many items they have:

If they have 17 items or less, suggest that they put a few things back so that they can use the express lane.

If they have more than 17 items, then suggest they put the cold items back.

Otherwise, if they don't have cold items, then tell them to use the regular lane.

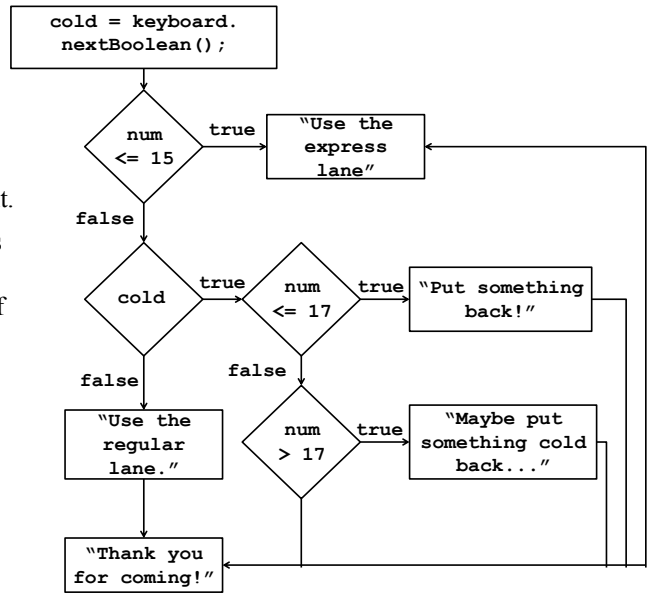
Well, how do I do that with code? I'm trying to not use those logical operators again…

Nested `if` Statements:

Reading: Section 5.5

`if` statements can be written as part of the true or false block of another **`if`** statement.

Typically, nest **`if`** statements when more information is required beyond the results of the first **`if`** condition.



Nested **`if`** Statements (continued):

```
System.out.print("Enter number of items you have: ");
int numItems = keyboard.nextInt();
System.out.print("Do you have cold items: ");
boolean coldItems = keyboard.nextBoolean();
if ( numItems <= 15 ) {
  System.out.println("Use the express lane!");
} else if ( coldItems) {
  if ( numItems <= 17) {
    System.out.println("Get those cold things home! " +
                       "See if you can put a few things back.");
  }else {
    System.out.println("Consider putting the cold items back.");
  }
} else {
  System.out.println("Use the regular check out line.");
}
System.out.println("Thank you for coming!");
```

Nested **if** Statements (continued):

All of these if statements only have one line in the body, so let's remove the curly brackets!

There is that rule that says if there is only one statement in a true or false block, we can remove the curly brackets, so let's do that!

Well, it turns out, this code will work the way we want it to…

```
if ( numItems <= 15 )
  System.out.println("Use the express lane!");
else if ( coldItems)
  if ( numItems <= 17)
    System.out.println("Get those cold things home! " +
                       "See if you can put a few things back.");
  else
    System.out.println("Consider putting the cold items back.");

else
  System.out.println("Use the regular check out line.");

System.out.println("Thank you for coming!");
```


Nested **if** Statements (continued):

All of these if statements only have one line in the body, so let's remove the curly brackets!

There is that rule that says if there is only one statement in a true or false block, we can remove the curly brackets, so let's do that!

But this code will **not** work the way we want it to…

```
if ( numItems <= 15 )
  System.out.println("Use the express lane!");
else if ( coldItems)
  if ( numItems <= 17)
    System.out.println("Get those cold things home! " +
                       "See if you can put a few things back.");


else
  System.out.println("Use the regular check out line.");

System.out.println("Thank you for coming!");
```

Nested **if** Statements (continued):

When looking at an **else**, how does the compiler know what **if** it belongs to?

Recall: The compiler ignores indentation!

Each **else** is associated with the most recent **if** that does not already have an **else**.

```
if ( numItems <= 15 )
   System.out.println("Use the express lane!");
else if ( coldItems)
   if ( numItems <= 17)
      System.out.println("Get those cold things home! " +
                         "See if you can put a few things back.");


else
   System.out.println("Use the regular check out line.");

System.out.println("Thank you for coming!");
```


Nested **if** Statements (continued):

When looking at an **else**, how does the compiler know what **if** it belongs to?

Recall: The compiler ignores indentation!

Each **else** is associated with the most recent **if** that does not already have an **else**.

```
if ( numItems <= 15 )
   System.out.println("Use the express lane!");
else if ( coldItems)
   if ( numItems <= 17)
      System.out.println("Get those cold things home! " +
                         "See if you can put a few things back.");
   else
      System.out.println("Consider putting the cold items back.");

else
   System.out.println("Use the regular check out line.");

System.out.println("Thank you for coming!");
```

Nested **`if`** Statements (continued):

When looking at an **`else`**, how does the compiler know what **`if`** it belongs to?

Recall: The compiler ignores indentation!

Each **`else`** is associated with the most recent **`if`** that does not already have an **`else`**.

So, how do we fix this?

- Can fix this by adding an **`else`** clause for the ice skating **`if`** statement.
  - This works.
  - But, can be awkward when we did not have anything to put in the **`else`**.
- Use **`{ }`**'s to surround the **`if`** statement that does not have an **`else`**.
  - When **`{ }`**'s are present, the **`if`** statement has to fit entirely inside the **`{ }`**'s.
  - This tells the compiler that the **`if`** statement does not have an **`else`** clause.

Block Scope:

Reading: page 225.

The *scope* of a variable is the region of code within a program where the variable can be referenced (or used).

Scope is determined by the *block* of code containing the variable declaration.

Code blocks:

- The **`main`** method is a code block.
- Code in the *true* clause of an **`if`** statement is a block.
- Code in the *false* clause of an **`if`** statement is a block.
- Code inside **`{ }`**'s is a block.

Block Scope (continued):

```
public static void main(String[] args)
{
  Scanner inputScan = new Scanner(System.in);
  int waterTemp;
  System.out.print("Enter the water temperature: ");
  waterTemp = inputScan.nextInt();
  if ( waterTemp <= 0 ) {
    System.out.println("Ice skating time!");
  } else {
    System.out.println("Go for a swim!");
    System.out.println("Might need a wet suit...");
  }
  System.out.println("Have a good time!");
} // end of method main
```

main's block

true clause

false clause

Block Scope (continued):

```
public static void main(String[] args)
{
  Scanner inputScan = new Scanner(System.in);
  int waterTemp;
  System.out.print("Enter the water temperature: ");
  waterTemp = inputScan.nextInt();
  if ( waterTemp <= 0 ) {
    System.out.println("Ice skating time!");
  } else {
    System.out.println("Go for a swim!");
    System.out.println("Might need a wet suit...");
  }
  System.out.println("Have a good time!");
} // end of method main
```

inputScan's Scope

waterTemp's Scope

Block Scope (continued):

This code works. There are two distinct variables, both named **area**. Each exists in a different scope.

```
public static void main(String[] args)
{
  Scanner inputScan = new Scanner( System.in );
  System.out.print("Enter the width and height: ");
  int width = inputScan.nextInt();
  int height = inputScan.nextInt();
  if ( width == height ) {
    int area = width * width;
    System.out.println("Area of square = " + area);
  } else {
    int area = width * height;
    System.out.println("Area of rectangle = " + area);
  }
} // end of method main
```

area

area

Block Scope (continued):

This code does not work. There are (still) two distinct variables, both named **area**. Each exists in a different scope. The reference to **area** after the **if** is invalid, since it lies outside the scope of both **area** variables.

```
//to save room I removed the lines of code that get
//width and height as use input
if ( width == height ) {
  int area = width * width;
  System.out.println("Area of square = " + area);
} else {
  int area = width * height;
  System.out.println("Area of rectangle = " + area);
}
int inchesArea;
inchesArea = area * 144;  // 144 sq inches in 1 sq foot
System.out.println("Area in square inches = " +
                   inchesArea);
} // end of method main
```

Compiler Error: cannot find symbol. How can this be fixed?

area

area

Testing Techniques:

Reading: Section 5.6

Execution Path Testing:

Develop a test plan that includes:

Running the program multiple times with data values that cause all **`true`** blocks to be executed,

AND all **`false`** blocks to be executed.

Check results against the program specifications.

Black Box Testing:

Treat the program like a black box:

Assume you do not know how the code is written.

Develop test data on program specifications.

Testing Techniques (continued):

Consider the flowchart.

What values of (**`xray`**, **`zap`**) are needed to test each path?