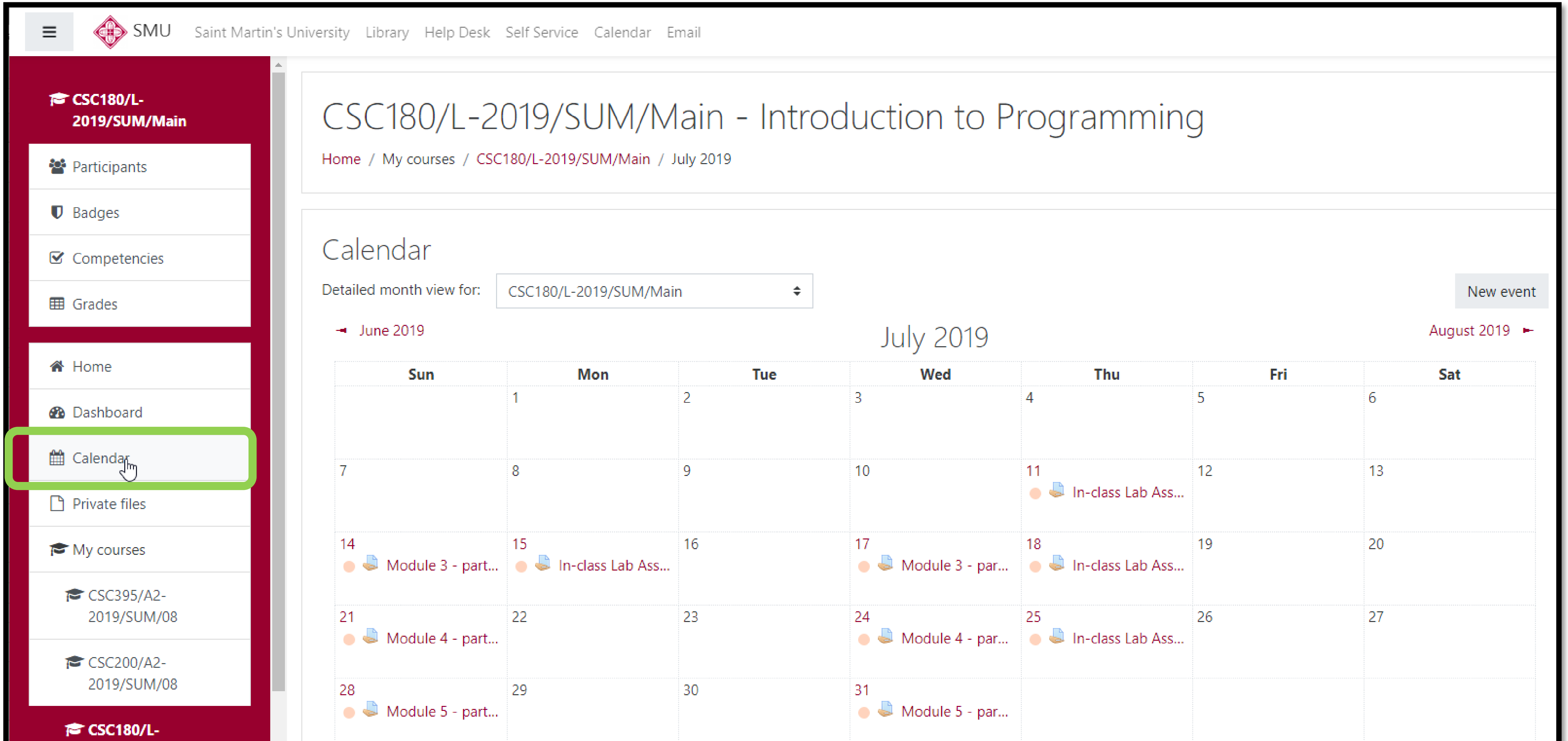
CHAPTER 4: CONDITIONALS AND RECURSION

Fall 2019 – CSC 180 – Introduction to Programming

REMINDER - DEADLINES

- In Moodle, remember to look into “Calendar” to see upcoming deadlines:



CSC180/L-2019/SUM/Main - Introduction to Programming

Home / My courses / CSC180/L-2019/SUM/Main / July 2019

Calendar

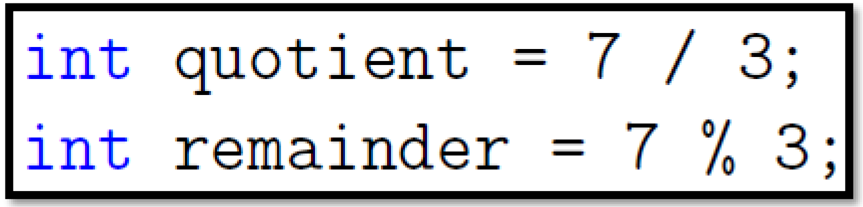
Detailed month view for: CSC180/L-2019/SUM/Main

July 2019

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11 In-class Lab Ass...	12	13
14 Module 3 - part...	15 In-class Lab Ass...	16	17 Module 3 - par...	18 In-class Lab Ass...	19	20
21 Module 4 - part...	22	23	24 Module 4 - par...	25 In-class Lab Ass...	26	27
28 Module 5 - part...	29	30	31 Module 5 - par...			

4.1 THE MODULUS OPERATOR

- The modulus operator works on integers (and integer expressions) and **yields the remainder** when the first operand is divided by the second. The modulus operator is a **percent sign** %



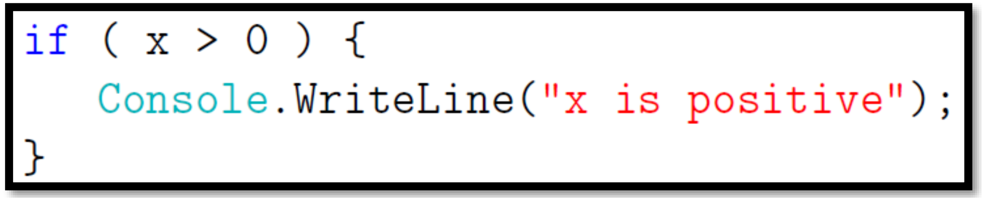
```
int quotient = 7 / 3;
int remainder = 7 % 3;
```

- The modulus operator turns out to be surprisingly useful. Very important to know:
  - How would you check if a number is even?
  - One can check whether a number is divisible by another: if x % y is zero, then x is divisible by y.
  - x % 10 yields the rightmost digit of x (in base 10). Similarly x % 100 yields the last two digits.

4.2 CONDITIONAL EXECUTION

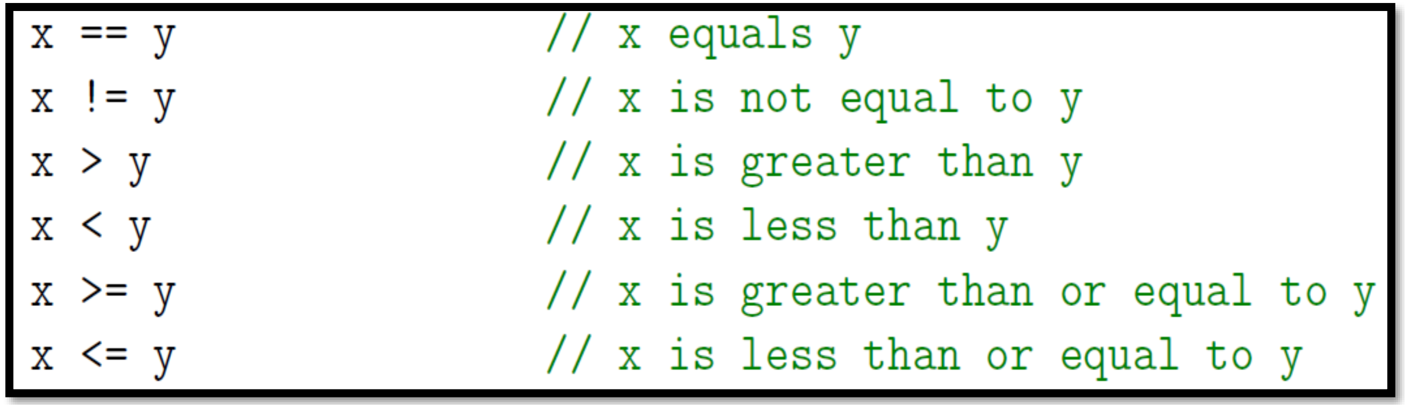
- To write useful programs, we almost always need to check conditions and change the behavior of the program accordingly. Conditional statements give us this ability.

- The simplest form of the if statement:
  - The expression in parentheses is called the condition.
    - If it is true, then the statements in braces get executed.
    - If the condition is not true, nothing happens.



```
if ( x > 0 ) {
   Console.WriteLine("x is positive");
}
```

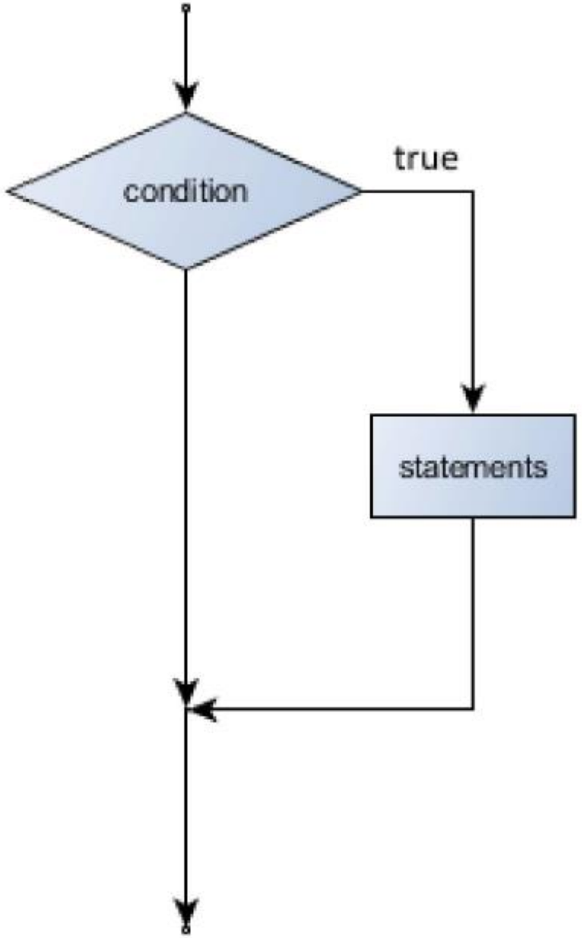
  - The condition can contain any comparison operators, sometimes called relational operators:
    - The condition can contain any Boolean expression (any expression that evaluates to either true of false)
    - Note: only == and != works with strings



```
x == y                   // x equals 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
```

- Remember that = is the assignment operator, and == is a comparison operator.

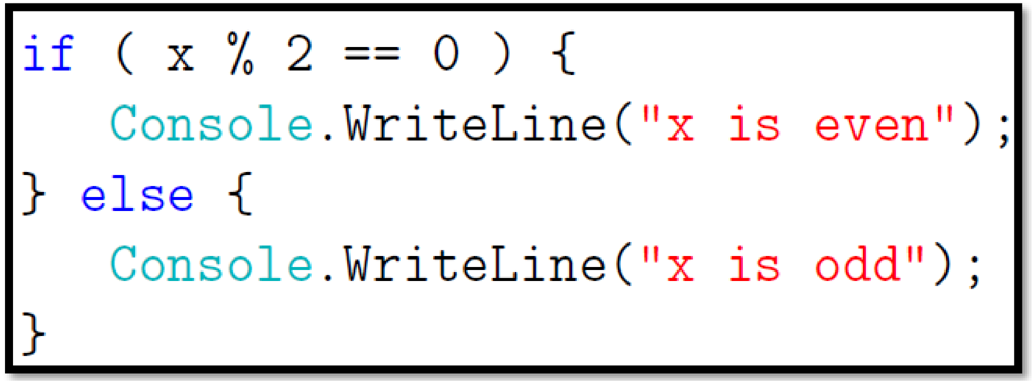
MORE TO KNOW …



- Typically we use { } to create a **block** of statements that you want to be executed together.
    - In this class I will require you to use them even if you only have one statement

- Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
    - Variables defined inside a set of braces have **local scope** or **block scope**.
    - They may only be used in the part of the program between their definition and the block's closing brace.

4.3 ALTERNATIVE EXECUTION

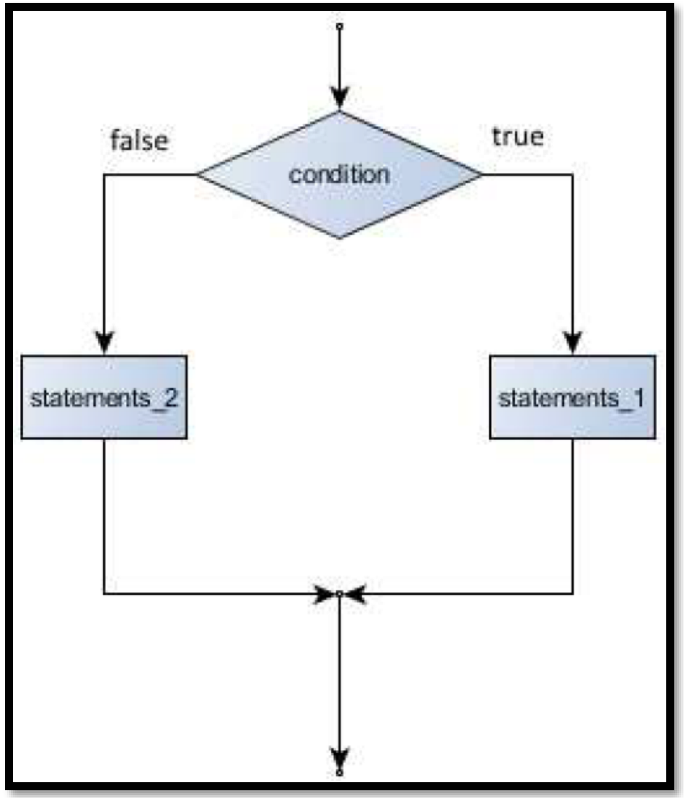


```
if ( x % 2 == 0 ) {
   Console.WriteLine("x is even");
} else {
   Console.WriteLine("x is odd");
}
```

- A second form of conditional execution is alternative execution, in which there are two possibilities, and the condition determines which one gets executed.

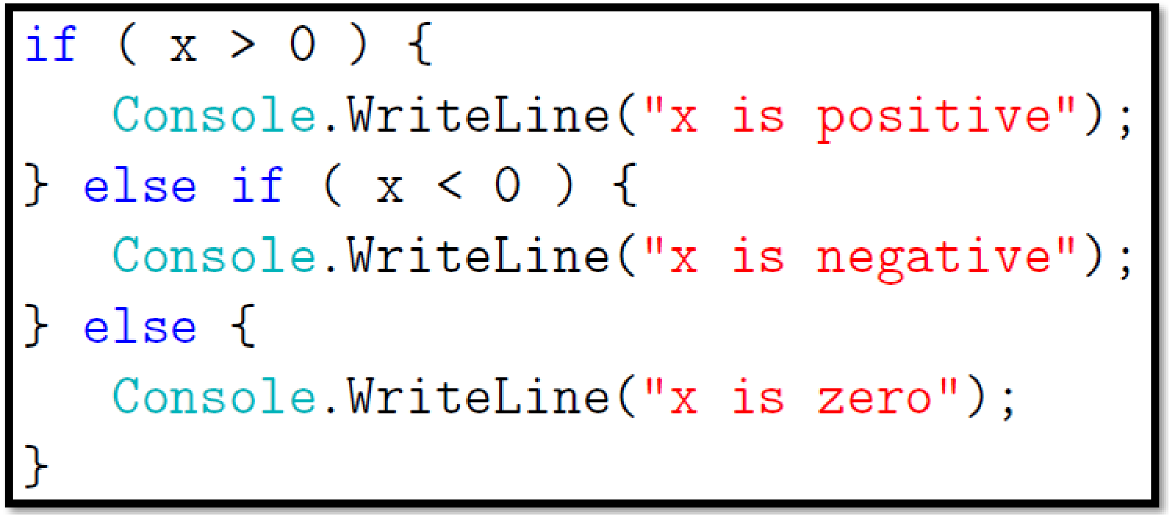
- The expression in parentheses is called the **condition**.
  - If it is true, then statement1 gets executed.
  - If the condition is not true (is false), then statement2 gets executed.

- Exercise:
  - Create a method definition that would display whether or not a number is even
  - Invoke the method for numbers 2019, 17, and 20.
    - In main show how to ask the user to enter three values instead …



4.4 CHAINED CONDITIONALS

- Sometimes you want to check for a number of related conditions and choose one of several actions.
- One way to do this is by chaining a series of **if**s and **else**s (another way is using switch statements)



```
if ( x > 0 ) {
   Console.WriteLine("x is positive");
} else if ( x < 0 ) {
   Console.WriteLine("x is negative");
} else {
   Console.WriteLine("x is zero");
}
```

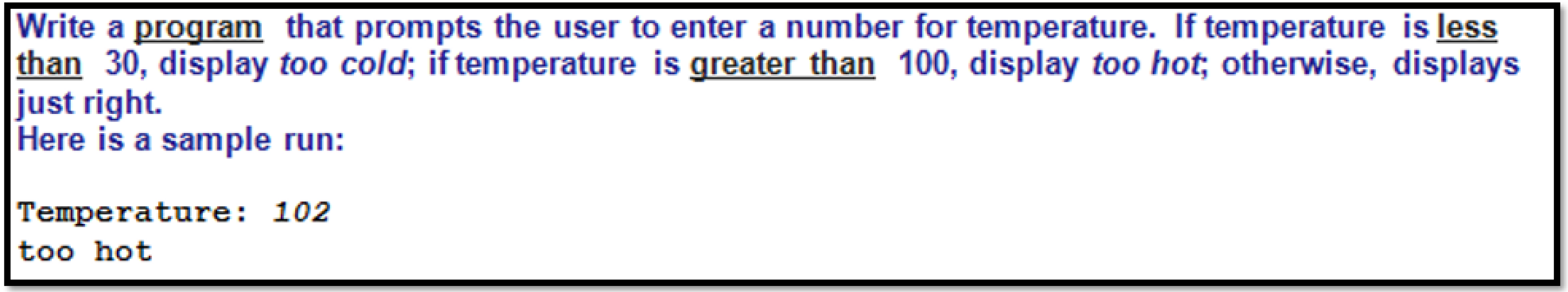
- Important discussion (take notes!):
  (if … if   vs if … else if …)
  - You may lose points on the homework if you use if … if where if … else if should be used.

- Exercise:
  - Write a program that asks the user to enter a score and it will display the corresponding letter-grade.
  - Use the following conversion table:

Score	Corresponding letter grade
Less than 60	F
[60,70)	D
[70,80)	C
[80,90)	B
At least 90	A

4.4 CHAINED CONDITIONALS - EXAMPLE

- Source: google C# programs



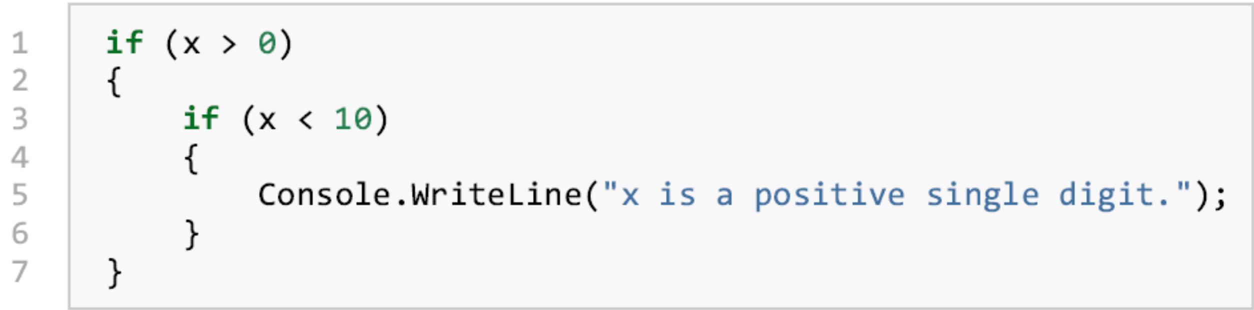
Write a program that prompts the user to enter a number for temperature. If temperature is less than 30, display *too cold*; if temperature is greater than 100, display *too hot*; otherwise, displays just right.

Here is a sample run:

```
Temperature: 102
too hot
```

4.5 NESTED CONDITIONALS

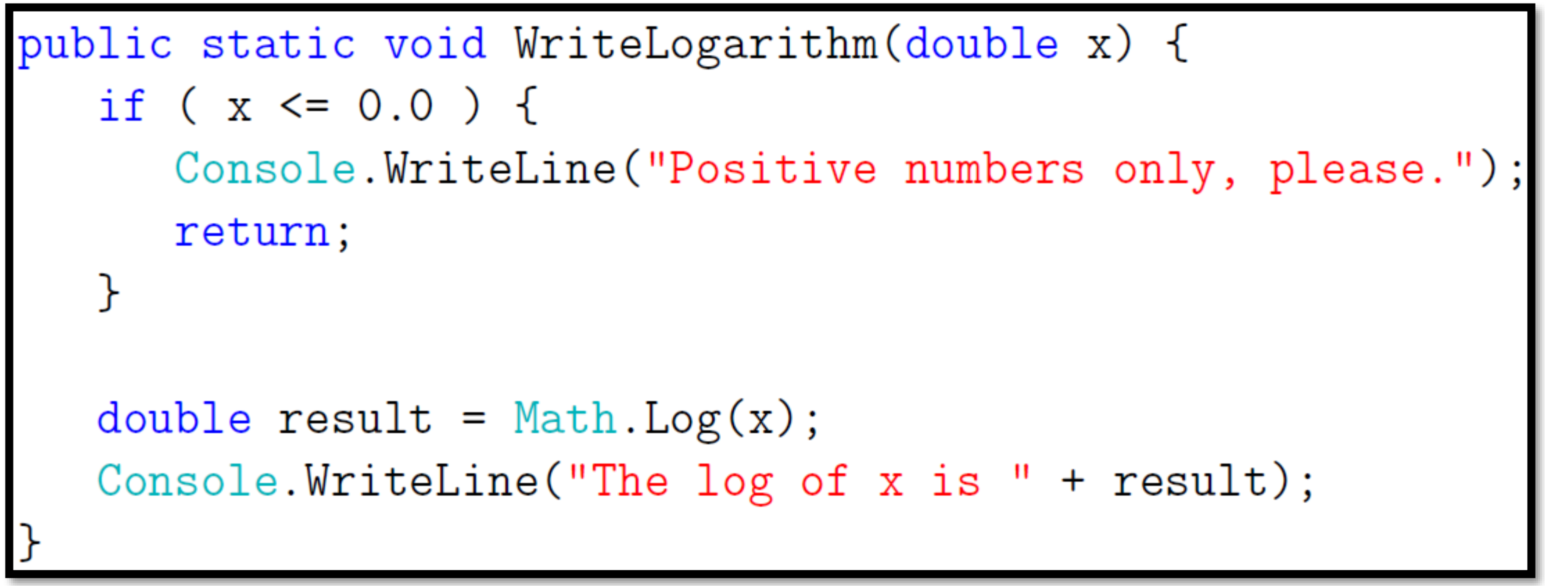
- you can also nest one conditional within another.



```
if (x > 0)
{
    if (x < 10)
    {
        Console.WriteLine("x is a positive single digit.");
    }
}
```

4.6 THE RETURN STATEMENT

- The return statement allows you to terminate the execution of a method before you reach the end.
  - One reason to use it is if you detect an error condition



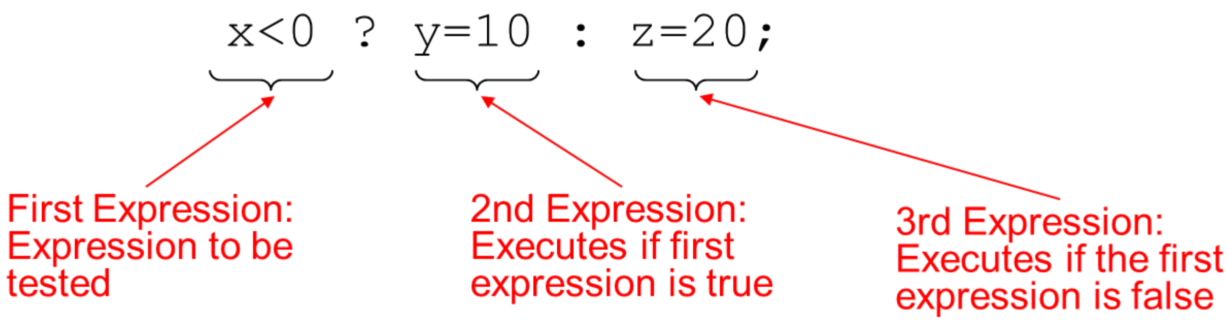
```
public static void WriteLogarithm(double x) {
    if ( x <= 0.0 ) {
        Console.WriteLine("Positive numbers only, please.");
        return;
    }

    double result = Math.Log(x);
    Console.WriteLine("The log of x is " + result);
}
```

- Exercise:
  - Write a program that asks the user to enter two **positive** values and it will display the largest of the two.
  - Input validation: do not accept values that are not positive!

THE CONDITIONAL OPERATOR – IF TIME

- Note: Do not confuse **conditional** operator with **conditional** statement
  - This is optional, you don't have to use this but please be aware of it in case you encounter it later
  - Format: cond ? expr1 : expr2;

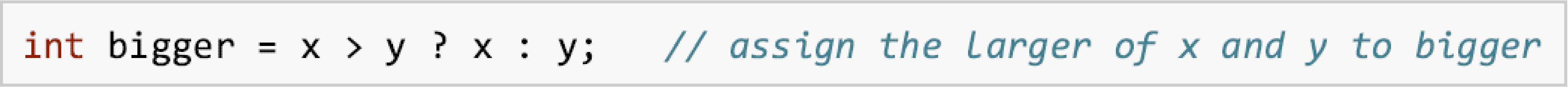


```
x<0 ? y=10 : z=20;
```

First Expression: Expression to be tested

2nd Expression: Executes if first expression is true

3rd Expression: Executes if the first expression is false

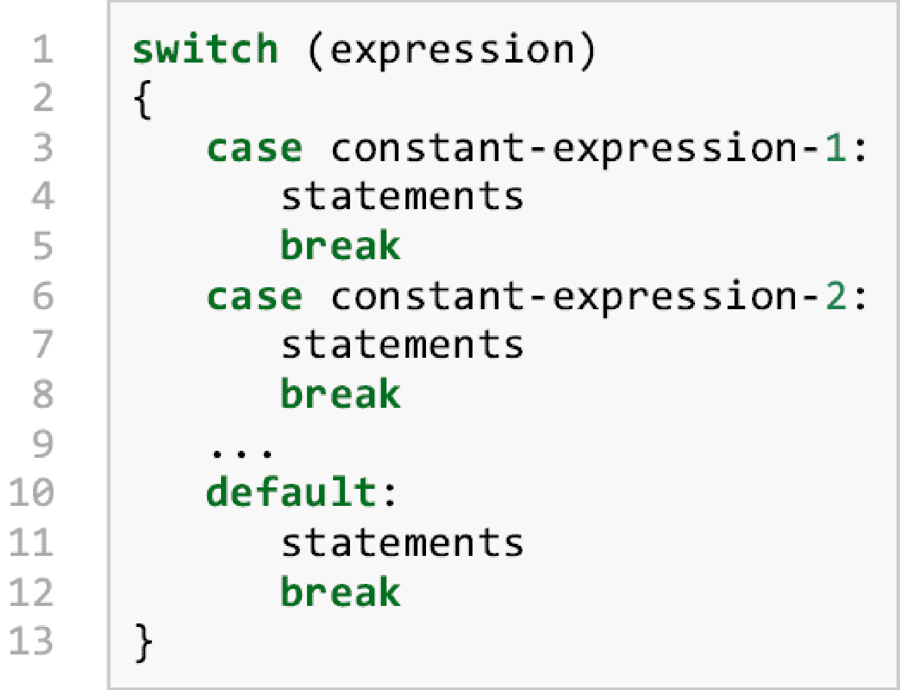


```
int bigger = x > y ? x : y;   // assign the larger of x and y to bigger
```

- Not recommended: these operators can be nested too …

SWITCH STATEMENT – IF TIME

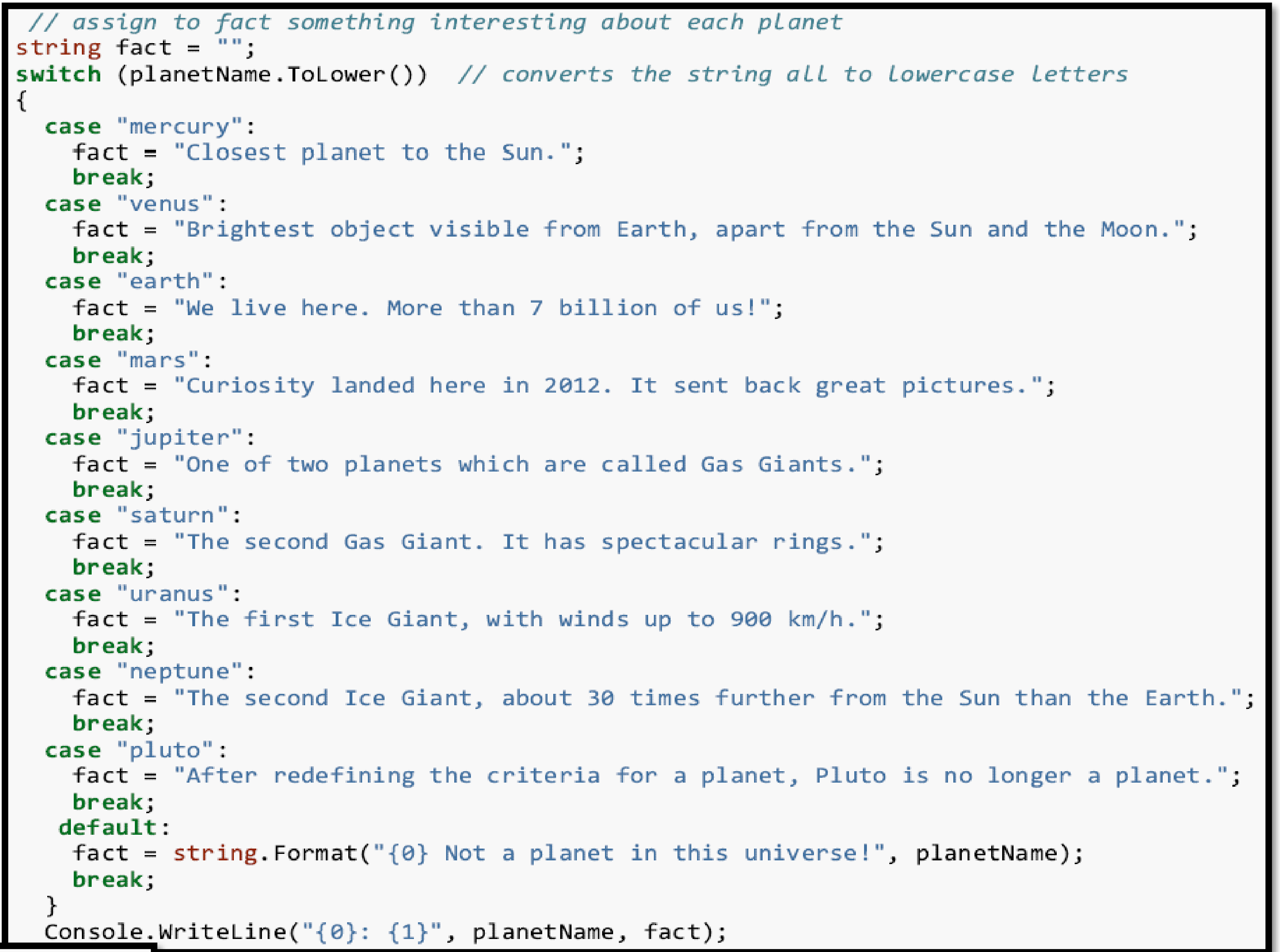
- An **if** statement can be used to choose between two alternatives.
- By contrast, the **switch** statement handles multiple selections by passing control to one of the case statements within its body.
  - Feel free to use if…else if instead of switch statements if you prefer
- A **break** statement is required after each case block
- If the switch expression doesn't match any case, the default block will be executed
- It is possible to have multiple case labels apply to a single block of statements



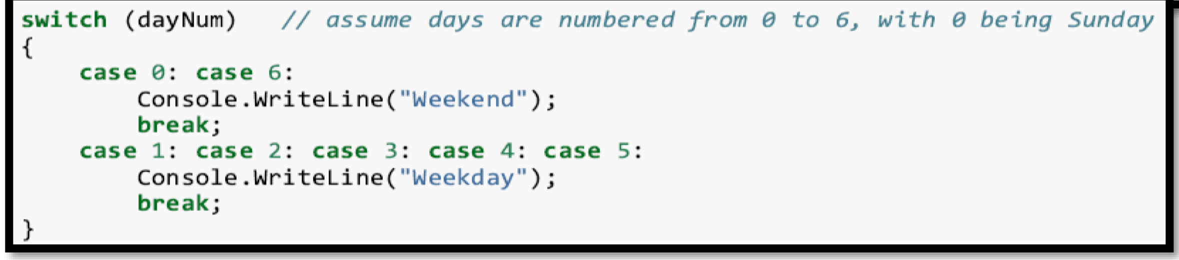
```
switch (expression)
{
    case constant-expression-1:
        statements
        break
    case constant-expression-2:
        statements
        break
    ...
    default:
        statements
        break
}
```

SWITCH – SKIP

- Examples:



```
 // assign to fact something interesting about each planet
string fact = "";
switch (planetName.ToLower())  // converts the string all to lowercase letters
{
  case "mercury":
    fact = "Closest planet to the Sun.";
    break;
  case "venus":
    fact = "Brightest object visible from Earth, apart from the Sun and the Moon.";
    break;
  case "earth":
    fact = "We live here. More than 7 billion of us!";
    break;
  case "mars":
    fact = "Curiosity landed here in 2012. It sent back great pictures.";
    break;
  case "jupiter":
    fact = "One of two planets which are called Gas Giants.";
    break;
  case "saturn":
    fact = "The second Gas Giant. It has spectacular rings.";
    break;
  case "uranus":
    fact = "The first Ice Giant, with winds up to 900 km/h.";
    break;
  case "neptune":
    fact = "The second Ice Giant, about 30 times further from the Sun than the Earth.";
    break;
  case "pluto":
    fact = "After redefining the criteria for a planet, Pluto is no longer a planet.";
    break;
   default:
    fact = string.Format("{0} Not a planet in this universe!", planetName);
    break;
  }
  Console.WriteLine("{0}: {1}", planetName, fact);
```



```
switch (dayNum)   // assume days are numbered from 0 to 6, with 0 being Sunday
{
    case 0: case 6:
        Console.WriteLine("Weekend");
        break;
    case 1: case 2: case 3: case 4: case 5:
        Console.WriteLine("Weekday");
        break;
}
```

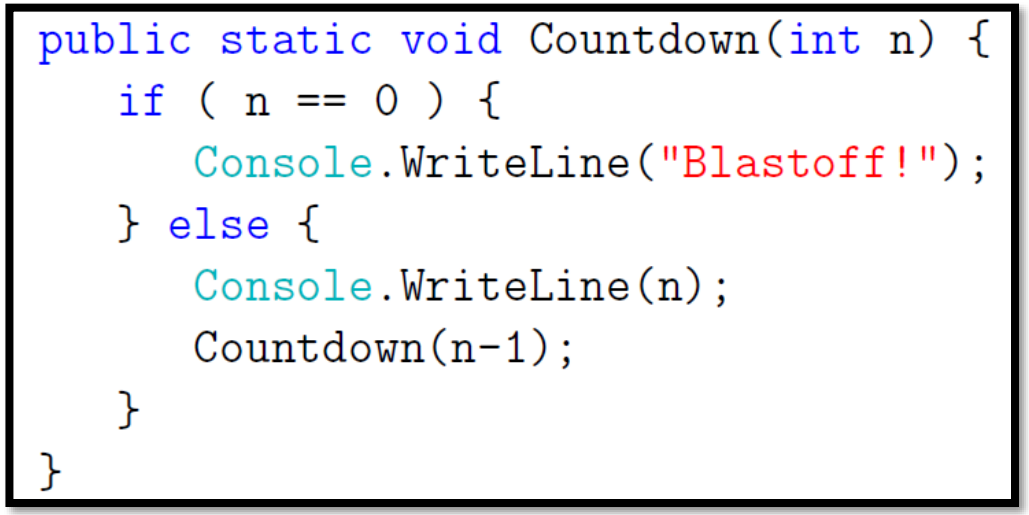
4.7 TYPE CONVERSION

- Discussion, take notes!
  - Casting
  - Convert.ToXXX

- val = Console.ReadLine(); //reads an entire line from the console and saves it into **val**
- int a = Convert.ToInt32(val); //converts a value (from variable **val**) to an integer and saves it into **a**
- double b = Convert.ToDouble(val); //converts a value (from variable **val**) to a double and saves it into **b**

4.8 RECURSION

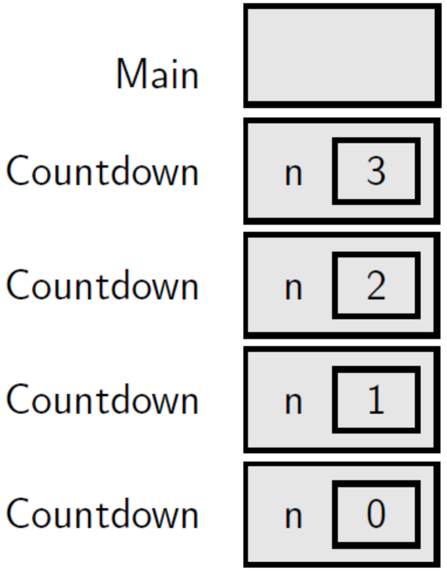
- it is legal for one method to invoke another
- it is also legal for a method to invoke itself.
  - It may not be obvious why that is a good thing, but it turns out to be one of the most magical and interesting things a program can do.



```
public static void Countdown(int n) {
    if ( n == 0 ) {
        Console.WriteLine("Blastoff!");
    } else {
        Console.WriteLine(n);
        Countdown(n-1);
    }
}
```

- A method that calls itself is called a recursive method
  - Beware of the infinite recursion

- Stack diagrams for recursive methods
  - every time a method gets called it creates a new frame that contains a new version of the method's parameters and variables.



IN-CLASS PRACTICE EXAMPLE – IF TIME

- A mobile phone service provider has three different subscription packages for its customers:
  - Package A: For $39.99 per month 450 minutes are provided. Additional minutes are $0.45 per minute.
  - Package B: For $59.99 per month 900 minutes are provided. Additional minutes are $0.40 per minute.
  - Package C: For $69.99 per month unlimited minutes are provided.

- Write a C# program that calculates a customer’s monthly bill. It should ask which package the customer has purchased and how many minutes were used. It should then display the total amount due.

IN-CLASS PRACTICE EXAMPLE – IF TIME

- To create a random integer between 10 and 100 (100 not included):
  - Random random = new Random(); //creates a new Random object
  - int randomNumber = random.Next(10, 100); //uses the Random object to generate a random integer
    double randomDouble = random.NextDouble();//generates a random num between 0 and 1.0 (not included)

- Create a program that "rolls two dice" and displays them.

- Generate two random numbers and ask the user to guess the correct sum. Then display whether or not the user answered correctly

- Generalize the previous problem so the program also generates a random operation (+,-,*,/)

HOMEWORK FOR CHAPTER 4

- Requirements: see moodle for details
- Deadline: see moodle

- **Reminder: If your code does not compile, crashes at start, or contains no meaningful comments, it will automatically be graded with 0!**