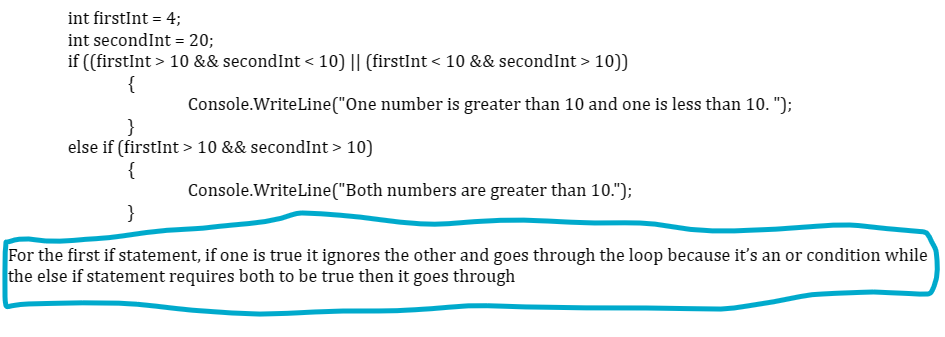
1. For question 1 on PE4 I checked that my code was correct but I was wondering if you could see the statement below and check if my way of thinking about how it worked was ok?



**Yes, that will satisfy the boolean tests required for the question.**

1. For number 2 on PE4 I had implemented the condition in the else if statement into a while loop instead and asked for new input if the numbers were greater than 10. Would that be ok or did you prefer us having conditionals instead?

**That approach is fine.**

1. Are enumerated types considered objects since they are lists?

**The unumerated type is a data type for creating objects.**

**So EColors { red, blue, green }**

**Defines a brand new data type that can be used to create variables of this type.**

**EColors enumeratedColor;  
enumeratedColor is an object which can only be set to EColors.red, EColors.blue or EColors.green. By default C# uses an int to store an enumerated type variable, and it will always be stored as a number.**

1. You were talking about how classes and structures use “public” so we can access variables outside the class. Can you go over how that would work exactly? Also, can we even access anything other than the variables outside the class with the use of public?

**We’ll go much deeper into this in Unit 2, but a class can contain variables, properties, methods, constructors and boolean operator overloads. Everything contained in a class has an accessibility attribute, which can be private, protected or public. Members which are public can be accessed outside of the class.**

**As an example, we can have a Student class:**

**public class Student { public string name; private string password; }**

**If I create a new student object: Student kashaf = new Student();  
I can access kashaf.name because it is public, but I cannot access kashaf.password because it is private and only accessible within the class’s code block.**

1. If we have multiple parameters that we want to make using “ref” is it ok to just put ref in front of the first parameter and not the others or do we need it in front of each parameter?

**The parameters are separate, so ref needs to prefix each parameter that is to be treated as a reference variable.**

1. Is it necessary to define the data type before creating a parameter?

**I’m not sure exactly what you are asking. Parameters must have a data type:**

**static void MyMethod(int parameter1, string parameter2, double parameter3, EColors parameter4)  
The parameters cannot be specified without a datatype.**

**But the Object datatype can reference ALL datatypes, so if you aren’t sure what you want to pass to a method, you can have it receive an object and pass anything that you want!  
static void MyMethod(Object parameter1, Object parameter2, Object parameter3, Object parameter4)**

1. I was trying to print “null” to the console but it was not working when I did a regular Console.WriteLine(varThatContainsNull) so I was wondering if it’s possible to even print that to the console?

**Printing null looks the same as an empty string. Note that you cannot compare a null variable against another data type. For example, you cannot write: if varThatContainsNull < “yellow”. That will generate a runtime error.**

1. Should we still set Console.ReadLine() equal to a variable if we just want to print it to the console or would it be just ok to do Console.WriteLine(Console.ReadLine())?

**This will repeat whatever the user typed in, but it won’t store it anywhere. You can do that, but you won’t be able to use their input anywhere in the program.**

1. Try and catch only works for one specific issue or problem that may occur, so we can’t put in 2 different possible errors that might happen within one try and catch?

**try/catch will catch any runtime exceptions that can occur in the try code block, so you could have one code block that might parse incorrectly and divide by zero and not be able to convert. For example:**

**try**

**{**

**// this will cause a runtime error if myString is not all numeric**

**int myInt = int.Parse(myString);**

**// this will cause a runtime error if myInt does not fit into myByte**

**byte myByte = Convert.ToByte(myInt);**

**// this will cause a runtime error because it is dividing by zero**

**int myQuotient = myByte / (myByte - myByte);**

**}**

**catch**

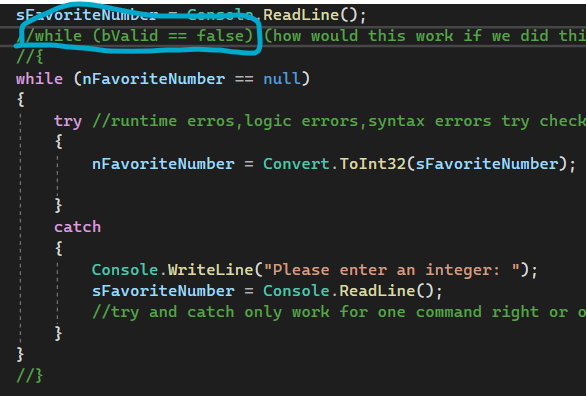
**{**

**// a runtime error occurred!**

**}**

1. Usually if there is a red line under our code is it safe to assume that would be a compile time error? And if we get an error in the console it’s usually a run time error?

**That’s right!**



1. For this screenshot I was just wondering if we made the while condition the boxed part, would the same process still hold but instead if the try works it would skip the catch and right after the catch we would put bValid = true so then it could exit the loop after it checks the while condition again?

**If you want to use while(bValid == false), you could do the following:**

**bValid = false;**

**while(bValid == false)**

**{**

**try**

**{**

**nFavoriteNumber = Convert.ToInt32(sFavoriteNumber);**

**bValid = true;**

**}**

**catch**

**{**

**Console.WriteLine(“Please enter an integer.”);**

**sFavoriteNumber = Console.ReadLine();**

**}**

**}**

**Notice bValid is set to true if Convert.ToInt32() was successful and it didn’t jump to the catch code block.**

1. For the user input question in PE3, you said that I would have to do int product = (int)(first\_conversion \* second\_conversion \* third\_conversion \* fourth\_conversion); for the code to work but I was wondering why we could not do int product = (int(first\_conversion \* second\_conversion \* third\_conversion \* fourth\_conversion)); because I thought that the second way would make all of the numbers ints before multiplying.

**Int is not a method, it’s a data type. To cast a variable as an int you use (int)variable. You could do: (int)first\_conversion \* (int)second\_conversion \* (int)third\_conversion \* (int)fourth\_conversion));**

1. I was still a bit confused about int? product = (first\_conversion \* second\_conversion \* third\_conversion \* fourth\_conversion); would we put the “?” after the int because our data types were initially set to int?. I thought that since we did Convert.ToInt32, it would have made our variables ints already?

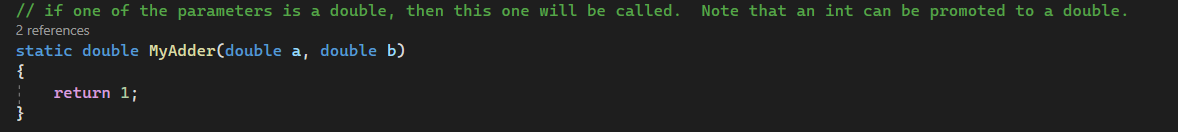
**int? Is more precise than int, since it stores the int and can store null. Therefore you cannot set an int = int? without casting it as an int.**

**int myInt = (int)myIntQuestionMark;**

**Convert.ToInt32() returns an int, not an int?, but you are storing the int in an int? Variable.**

1. For the steps for making a new source file would we have to put “partial” before every single class program, even our main file we started with? Also, would we have to put our name spaces as the same name as the parent file so we can call things within the parent from another file and the other way around?

**If you want to access everything in all of your source files without specifying the namespace (or using the “using” statement), then you want them all to use the same namespace. To have the same class span multiple source files, you must use “partial” everywhere you define the class, even the main file you started with. The code from Session 2-2 documents that.**



1. For this screenshot here for the comment you put, if the int can be promoted to the double does it not add the “.0” after it? Also I just wanted to be sure that we can convert an int to a double but not a double to an int?

**When you write a line of code and you include “.0”, C# treats that number as a double.**

**So myInt \* 5.0 will return a double, and myInt \* 5 will return an int.**

**But when you look at the contents of a double, it doesn’t add .0 to it so you can’t tell what type of variable it is simply by looking at its value.**

**double myDouble = myInt; // you can do this implicitly and by passing myInt to a double method parameter**

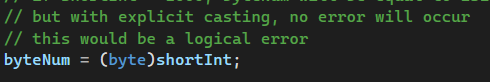
**int myInt = myDouble; // you cannot do this implicitly or by passing myDouble to an int method parameter**

1. When you said that double is prone to rounding errors did you mean that it rounds down no matter what even if the number may be 4.9?

**No, the rounding errors occur when adding small increments, like 0.1. If a double equals 4.9, then it will always equal 4.9.**

**But if you add 0.1 to 4.9, then it might be equal to 0.4999999999**

1. I was still confused about explicit and implicit casting. So basically for implicit we can convert from a more precise to a less precise data type and it would retain all of its information, but for explicit casting, in the screenshot below would it be a logic error because the data is lost but it still displays? Also, for checked() does it not return anything like a bool. or something like that or would it only be the error if data is lost? Is the main use of it just throwing an error if data is lost?



d

**You can implicitly copy a less precise data type into a more precise datatype: myInt = myByte;**

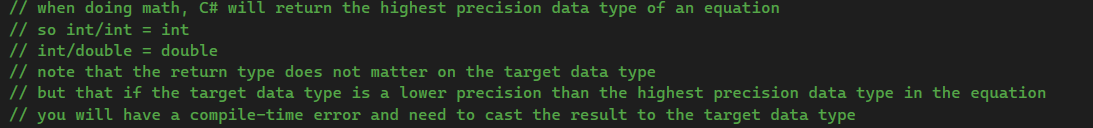
**You must explicitly cast if you go the other way: myByte = (byte)myInt;**

**checked() does not return a value, it simply generates a runtime error if data would be lost.**

1. For the same topic on explicit casting, why would we even need a try catch if it converts without one like above? Both ways would involve data being lost so why is it more beneficial?

**The explicit cast example above would be a logical error and you would never know that the data was lost, and you may not care because you only care about the lowest 8 bits. But if you are depositing $1,000 into your bank account and your code used a byte instead of an int to store how much you were depositing, then you would lose a lot of money, and you would want your program to alert you. You would would use check() or the Convert class with a try/catch to ensure you catch the conversion error and not lose your money!**

1. For the screenshot below what did you mean by the last 3 comments?



**If you have the code: double myDouble = myInt1 / myInt2**

**just because you are setting the quotient to a double doesn’t mean that C# will use doubles to do the division. int/int = int regardless of what kind of variable you are storing it in. so double myDouble = 5 / 2 will be equal to 2, not 2.5 even though I am storing the result in a double. You need to cast one of the operands into a double to get a double result: myDouble = (double)5/2**

**And if you try myInt = myDouble1/myDouble2, you will get a compile-time error because a double cannot be implicitly copied into an int. So you need to cast the answer: myInt = (int)(myDouble1/myDouble2)**

1. Unsigned means that the variable of that data type can’t be negative right?

**Correct**

1. Does short just round down to the nearest whole number usually? Is that the case for double as well? If so, would the alternative be round then convert it back to a double to use the math functions?



**truncated means that any decimal points are simply chopped off. Which is the same as rounding down for positive numbers, but rounding up for negative numbers! This is because 3.94 will be truncated to 3 (rounded down). -3.94 will be truncated to -3 (rounded up). Doubles aren’t truncated.**

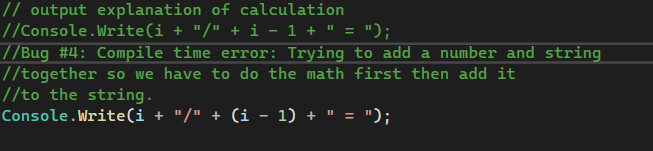
1. When you mentioned: //when we call methods they are specific to the datatypes being passed into them. Does that mean if we have a parameter set to an int it will only accept an int.?

**If the parameter is an int then it will accept any data type as precise or less than int. So it will accept int, short, byte and char.**

1. With making another source file, is the parent the only one with the main method and the other source files usually don’t have it?

**You can only have one Main() method per application, which usually goes in first created source file.**

1. For this screenshot below I was just confused if this was a compile time error because I thought it would have been run time?



**It’s compile time because you cannot subtract in Console.Write(). My comment was not quite right! You can add a number and a string, but you cannot subtract.**

**If the original code was: Console.Write(i + “/” + i + 1 + “ = “);   
the it would be a logical error and not a compile time error, because the concatenation would work, but the output would be incorrect. For i = 1, it would output: “1 / 11 = ”**

1. For our loop conditions, the last number was always exclusive right? (Ex. we wanted to go up to i = 10, then we would have to do i<11 or i<=10).

**Using less than is exclusive. Using <= is inclusive.**

1. TryParse does not accept a variable that was initialized with a “?” right?

**Correct. It specifically only accepts its own type:**

**int.TryParse(string, out int) // cannot use out int?**

**double.TryParse(string, out double) // cannot use out double?**

1. Should we know all 3 ways of converting or is one good?

**You should know all 3 since they are not interchangeable. Parse() only converts from strings for example, whereas Convert can convert from most other data types.**

1. For my code for “Flow Control” that I had submitted I was wondering if there was a way it could be simplified because it works, but I feel like it’s too repetitive.

**You can move the initial Console.ReadLine()’s into the while() loops and delete lines 39-49.**

**while(startImagValNum == null)**

**{**

**Console.WriteLine("Enter an actual number for the starting imagniary number (a good starting point would be at 1.2): ");**

**string startImagVal = Console.ReadLine();**

**try**

**{**

**startImagValNum = Convert.ToDouble(startImagVal);**

**}**

**catch**

**{**

**Console.WriteLine("That is not a valid number.");**

**}**

**}**

**while (endImagValNum == null)**

**{**

**Console.WriteLine("Enter an acutal number less than " + startImagValNum + " for the ending imagniary number (a good ending point would be at -1.2): ");**

**string endImagVal = Console.ReadLine();**

**try**

**{**

**endImagValNum = Convert.ToDouble(endImagVal);**

**}**

**catch**

**{**

**Console.WriteLine("That is not a valid number.")**

**continue;**

**}**

**// c# will generate a runtime error if you try to compare an int with a null value, so we need the continue above**

**if (endImagValNum > startImagValNum)**

**{**

**endImagValNum = null;**

**}**

**}**

**while (startRealNum == null)**

**{**

**Console.WriteLine("Enter an actual number for the starting real number (a good starting point would be -0.6): ");**

**string startReal = Console.ReadLine();**

**try**

**{**

**startRealNum = Convert.ToDouble(startReal);**

**}**

**catch**

**{**

**Console.WriteLine("That is not a valid number.");**

**}**

**}**

**while (endRealNum == null)**

**{**

**Console.WriteLine("Enter an actual number greater than " + startRealNum + " for the ending real number (a good ending point would be 1.77): ");**

**string endReal = Console.ReadLine();**

**try**

**{**

**endRealNum = Convert.ToDouble(endReal);**

**}**

**catch**

**{**

**Console.WriteLine("That is not a valid number.");**

**continue;**

**}**

**if (endRealNum < startRealNum)**

**{**

**endRealNum = null;**

**}**

**}**

1. When you had answered my question about writing to the console with a “+” rather than a ”,”, if we put a “,” it basically reads the first thing and ignores the rest?

**Everything is ignored after the “,” because it is used to substitute the tokens {0}, {1}, {2}, etc. So if you don’t have those in your string, then it ignores everything after the first “,”.**

1. For the “out” keyword if we tried to access a variable with “out” before changing it would it give us the value we initialized the variable with or does it not care about that and it would return an error?

**static void MyMethod(out int intParm )**

**{**

**// if you try to access intParm in the method before it is set, you will have a compile-time error**

**Console.WriteLine(intParm); // because it is an “out” parameter, it is assumed to be undefined and we cannot access its value**

**intParm = 42; // now we have set it**

**Console.WriteLine(intParm); // so now we can output it**

**}**

**static void Main()**

**{**

**// if we have a variable in the parent method**

**// we can do anything we want with it**

**int myInt = 0;**

**myInt += 42;**

**myInt /= 7;**

**Console.WriteLine(“My favorite number is “ + myInt);**

**// but when we call MyMethod() with it**

**MyMethod(out myInt); // we cannot access its value in the method. The method can only set it to a value**

**}**

1. Can goto only be used with switch statements only or could it be used with other things?

**goto can be used anywhere as you will see this week**

1. We had talked about string interpolation but I was confused on how to use it.

**The “$” symbol is the interpolation operator.**

**int myInt = 42;**

**double myDouble = 6;**

**string myString = $”{myInt} / {myDouble} = {myInt/myDouble}. I like turtles and math!”;**

**Interpolation allows us to insert code blocks into the string and they will be evaluated and inserted into the string. Code blocks start with “{“ and end with “}” and can be any valid C# expression. In the example above, myString will be equal to: “42 / 6 = 7. I like turtles and math!”**

1. These are the right definitions for break and return right?



**Correct**

1. For the questions you had answered about PE3 I was just wondering how your code worked and wanted to confirm I was thinking correctly because my question about my code had been answered only not about how your code had worked. Since we put while true, it keeps trying to convert that specific entry we are on from the for loop to an int? After would we keep getting input from the user until it converts to an int., then multiply the number into nNumber then keeps going until we reach 4 entries?

**Correct. In the try code block, if the string is able to be converted to the int, then the next line will execute, which is the “break” and will break from the infinite loop to move on to the next number.**

1. Also I was reading through the arrays code and I needed a walk through for the code you had included with the 2D/3D arrays so is it ok if we go over that after class on Tuesday or on Wednesday during office hours?  
   **Yes, I can be available for up to 15 minutes minutes after class and definitely during Wednesday’s office hours.**