**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

For each of the below questions, write a short sentence or two to express (in your own words) your answer. Keep the answers short, but use complete, correct, English sentences.

If it helps to clarify the questions, feel free to mentally prefix all the questions with the phrase "According to the video…"

1. After you’ve watched all the videos, please answer this question:  
   Of all the videos that you watched, if you could pick one video to be re-recorded by the instructor outside of class which would you choose? Why?  
   (Keep in mind the recording outside of class will omit any pauses from the instructor answering student questions, have less hemming and hawing, etc, and generally be more concise)

|  |
| --- |
| < Write your answer here > |

**VIDEO: OOP Basics**

1. The basic idea behind object oriented programming is to combine what two things together, in one place?

|  |
| --- |
|  |

1. What is the first “part” of the rectangle class?

|  |
| --- |
|  |

1. Data should always be declared as what?  
   (Remember that you can always create a public method to access these data fields later.)

|  |
| --- |
|  |

1. Even though the behavior (the methods) are defined for all instances of this class, calling the calculateArea method will cause the area to be calculated based on what?

|  |
| --- |
|  |

1. What is the “constructor” used for?

|  |
| --- |
|  |

1. How does the constructor copy a starting value for the width (or height) into the rectangle’s “permanent” memory?

|  |
| --- |
|  |

1. Given a choice between making a variable a permanent (“instance”) variable or a temporary variable (typically a local variable), which type should you prefer to use? Under what circumstances would you use the other one?

|  |
| --- |
|  |

**VIDEO: OOP Encapsulation**

1. In C# you can use either a getHeight() method or a C# property. Why will we be using the getHeight() method in this course?

|  |
| --- |
|  |

1. In addition to allowing us to make basic changes to objects (such as changing **r1**’s height to be 10), what else does the accessor method make it easy for us to do?  
   (Hint: What if we try to change **r1**’s height to be **-**10)

|  |
| --- |
|  |

1. If one makes the height field public what implications will that have for the program (in terms of who can access that field)?

|  |
| --- |
|  |

1. Is the public/private/protected feature that we’re looking at a security mechanism?

|  |
| --- |
|  |

1. Why are the constructor and calculateArea methods allowed to access the private **height** field?

|  |
| --- |
|  |

1. Can the **main**  method access the private **height** field of the **r1** object?

|  |
| --- |
|  |

1. Why is it ok to call the calculateArea method outside the Rectangle class (say, in **main**)?

|  |
| --- |
|  |

1. Let’s say that you wanted to ensure that the height field of all rectangle objects are not negative. Prior to the public/private/protected access-control mechanism, what was your only option for ensuring that this would be true? Why would this be difficult?

|  |
| --- |
|  |

1. Fill in the blank: “Just to recap, encapsulation is:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”

|  |
| --- |
|  |

**VIDEO: Object Oriented Programming (OOP): Background / Motivation**

1. Prior to object oriented programming, how were programs (typically) divided up?

|  |
| --- |
| < Etc. > |

1. How does object oriented programming (typically) change the way programs are divided up?

|  |
| --- |
|  |

1. Fill in the blanks: “The basic idea is that in object oriented programming we’ll have a [Car] **class, which describes \_\_\_\_\_\_\_\_\_\_\_\_\_\_**, and we’ll have an individual **object which describes \_\_\_\_\_\_\_\_\_\_\_\_\_\_**”

|  |
| --- |
|  |

1. Fill in the blank: “An object is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_” ( about 6:30)

|  |
| --- |
|  |

**VIDEO: OOP: (Instance) Methods, Object Allocation Demo**

1. What is a good type of name for a class (and what are a couple of examples of bad names for a class)?

|  |
| --- |
|  |

1. If you wanted to add a “Print” command to a class named Car, where would you put the Print method’s definition? Describe the location in English, then copy the example code from the video (at about the 1:50 mark)

|  |
| --- |
|  |

(After 2:15 the video answers a number of tangential questions –skim this if you’re interested, but don’t worry about recalling the details of this part of the video)

**VIDEO: OOP: Instance Variables**

1. Fill in the blank: “Unless you’ve got a really good reason [to do otherwise], you should always make all your instance variables \_\_\_\_\_\_\_\_\_\_\_\_\_”

|  |
| --- |
|  |

1. When you create a new Car object, what value will be put into all of the instance variables for that particular object?

|  |
| --- |
|  |

1. Describe in your own words how the SetMaxSpeed method allows one to change the maximum speed for a car object.

|  |
| --- |
|  |

1. Why is it ok to set maxSpeed in the SetMaxSpeed method, and then use it in the Print method?

|  |
| --- |
|  |

1. How does writing out as your variable as **this.**maxSpeed help to make your program more clear?

|  |
| --- |
|  |

(Starting at about 9:30 there’s a very visual explanation of how memory is managed when creating and using objects. Even though there aren’t a ton of questions about this, definitely watch this in order to develop an intuition about how object creation & usage works!!)

1. What is the purpose of the invisible parameter named **this**? What information does it pass from main to the SetMaxSpeed method?

|  |
| --- |
|  |

1. Inside the SetMaxSpeed method, how does C# figure out what the (variable) name maxSpeed refers to?

|  |
| --- |
|  |

1. Because of the procedure you outlined in your answer to the prior question, is it possible to create a local variable that has the same name as an instance variable?

|  |
| --- |
|  |

1. Should you ever create a local variable that has the same name as an instance variable?   
   ( Hint: NO!!!!!!!!!!!!!! ☺ )

|  |
| --- |
|  |

1. Does the method’s name cause the action/effect of the method?   
   If not, why is it still a good idea to choose method names that describe the each method does?

|  |
| --- |
|  |

1. The “Stack” is very, very efficient at doing what?

|  |
| --- |
|  |

**VIDEO: OOP: Access control (public/private), getter/setter methods**

1. The primary purpose of access control is prevent my co-workers from doing what?

|  |
| --- |
|  |

1. What can the SetMaxSpeed method do that will help prevent errors for being made in the overall program?

|  |
| --- |
|  |

1. What does the keyword “private” mean / what does it do?

|  |
| --- |
|  |

1. If you try to access a private data member (say, in **main**), what error message will you get?

|  |
| --- |
|  |

1. Why is **main** allowed to call SetMaxSpeed? Why is SetMaxSpeed allowed to change **maxSpeed**?

|  |
| --- |
|  |

1. Another advantage to forcing everyone to use accessor methods is that you can change what part of the class/program and yet NOT have to change the rest of the program?

|  |
| --- |
|  |

(Around the 9 minute mark there’s a blurb about exception handling – it is interesting to know about, but you’re not required to know it (unless there’s other material in this class that specifically requires you to know it) ).

1. Does access control protect you from intentionally malevolent actions by your co-workers? If not, then where is it useful?

|  |
| --- |
|  |

**VIDEO: Overloading**

1. Are over**load**ing a method and over**riding** a method the same thing? If not, give a really quick, intuitive explanation of each one (you haven’t seen overriding yet so don’t worry about providing a detailed/entirely correct explanation – the main thing is that you know these are different)

|  |
| --- |
|  |

1. There’s an example of overloading in the video involving a printBox method (right around the 3 minute mark) – briefly summarize what we were trying to accomplish with the printBox methods, and why it’s useful to have multiple methods that all have the same name.

|  |
| --- |
|  |

1. The basic rule for figuring out if you’ve legally overloaded a method is if you’ve got enough information at what point in the program? What information do you need, specifically?

|  |
| --- |
|  |

1. Fill in the blank: “Overloading makes sense when you have one \_\_\_\_\_\_\_\_\_\_\_, but a whole bunch of different \_\_\_\_\_\_\_\_\_\_\_\_”

|  |
| --- |
|  |

1. Can C# differentiate which version of the method to call based on the return value?  
   If not, why not?

|  |
| --- |
|  |

1. Is it less efficient to call one overloaded version of the function, and then have it immediately call the ‘real’ version? If so, why do we do it?

|  |
| --- |
|  |

1. Give an example of an overloaded method that’s built into the .Net Framework Class Library:

|  |
| --- |
|  |

**VIDEO: OOP: Constructors**

1. Implementing a constructor for your class will help you avoid making what sort of error?

|  |
| --- |
|  |

1. When does the constructor method run/execute?

|  |
| --- |
|  |

1. When the Car class has only 1 constructor, that takes a single parameter, what sort of compile-time can occur? Why is this error good / useful?

|  |
| --- |
|  |

1. What two things identify a method as being a constructor?

|  |
| --- |
|  |

1. How can you have one constructor call a different constructor? Give a brief snippet of C# code that demonstrates this, and then provide a brief explanation (in English, in your own words) about how this works:

|  |
| --- |
|  |

**VIDEO: Basic Array of Objects**(You can jump to the 1:30 mark & watch from there – the first minute is not useful ☺ )

1. In your own words BRIEFLY outline what the **new** operator does when asked to create an array of Demo objects. Make sure that you’re clear what value (if any) that gets placed into each array slot.

|  |
| --- |
|  |

1. What does the keyword **null** mean/represent in C#?

|  |
| --- |
|  |

1. What is a simple, intuitive rule for remembering what default value will be filled into a new array?

|  |
| --- |
|  |

1. In the below snippet of C# code, what does line **(1)** do? What does line **(3)** do? Do you really need both? How are they different?

Demo [] ds = new Demo[ 5 ]; **(1)**  
Console.WriteLine( ds[ 0 ] );  
ds[1] = new Demo(); **(3)**

|  |
| --- |
|  |

1. How do you call the printBox() method on the object at array slot #1 in the array named ds?

|  |
| --- |
|  |

**VIDEO: Null References Within An Array**

1. What happens when you attempt to call the printBox() method on the third slot in the ds array (ds[2]) ? Why does that happen?

|  |
| --- |
|  |

1. What sort of exception will see in when you attempt to call the printBox() method on the third slot in the ds array (ds[2]) ?

|  |
| --- |
|  |

1. How can you check (using C# code) if an array slot is **not** null? Provide a short snippet of C# code, and a brief, intuitive explanation of how it works:

|  |
| --- |
|  |

1. How can you check (using C# code) if an array slot **is** null? Provide a short snippet of C# code, and a brief, intuitive explanation of how it works:

|  |
| --- |
|  |