**Name: JASON TRUONG**

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…"

* 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**

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

|  |
| --- |
| Data, and behaviour. |

* What is the first “part” of the rectangle class?

|  |
| --- |
| Set of private data fields, width and height. |

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

|  |
| --- |
| suggested to always be private |

* 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?

|  |
| --- |
| Based on the private data declared prior, width and height |

* What is the “constructor” used for?

|  |
| --- |
| To initialize and define behaviours of the width and height |

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

|  |
| --- |
| By supertransposing the numbers into the private data values in the rectacle class |

* 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?

|  |
| --- |
| When remembering reusable information for a constructor, the permanent data is prefferable  If not, just use the temporary data until otherwise |

**VIDEO: OOP Encapsulation**

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

|  |
| --- |
| An accessor method is more common than a property, found in other OOP languages like Java or C++ |

* 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)

|  |
| --- |
| They minimize bad values the program will allow |

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

|  |
| --- |
| Anyone anywhere in the code can access and edit the field if its public |

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

|  |
| --- |
| Its not a security mechanism, it prevents value modifying |

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

|  |
| --- |
| Because they are all blocked in the same defined class, in this case, 'Rectangle' |

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

|  |
| --- |
| No, it can't |

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

|  |
| --- |
| calculateArea() is a public method so it can be called outside its class and into another class; say, in main. |

* 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?

|  |
| --- |
| You can only just be cautious and mindful when writing code so have it intentionally not accept negative values. When working on a collaborative project, its not much of what you can handle when your collab-mates make the mistake that you're trying to avoid. |

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

|  |
| --- |
| you marking your data fields and methods, private or public, [and later one we're going to see protected]. Basically marking them as either accessible to the outside world, which is public. Or only accessible through whats within the class, which is private. |

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

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

|  |
| --- |
| A bad name would be something that overlaps with pre-existing classes, are too simplified, or nearly meaningless. Bad names include Bob and Mary.  Good names are typically meaningful unique categorical names. |

* 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)

|  |
| --- |
| class Car {  public void Print()  {  Console.WriteLine("Car object");  }  } |

(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**

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

|  |
| --- |
| private |

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

|  |
| --- |
| 0 will be in place of the instance |

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

|  |
| --- |
| When SetMaxSpeed() is called in main, it passes a valid value through the parameter. Then takes the constructed integer and assigns itself to the private object maxSpeed, and changes established private integer for the whole class. |

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

|  |
| --- |
| Because the maxSpeed variable is available for all methods inside the class Car, that includes Print(). |

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

|  |
| --- |
| this is an invisible parameter and command that is clarified by us to make the changes in the object maxSpeed clearer when used in other methods with the new value. |

(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!!)

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

|  |
| --- |
| this, in this case, is in reference to the Car object. The information passed through the SetMaxSpeed() method can and will be stored into the Car() call object |

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

|  |
| --- |
| C# would need to conclude there isn't a local or parameter named maxSpeed, so it goes ahead and attaches the this parameter and checks for all instance variables named maxSpeed in its stead. |

* 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?

|  |
| --- |
| Its possible to create a local and instance variable with same names. |

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

|  |
| --- |
| No. The local variable takes precedence over the instance variable. The program will be right if otherwise. |

* 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?

|  |
| --- |
| The name of the method does nothing to the code, its always useful to have a fitting name for accessibility. |

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

|  |
| --- |
| Deallocating and allocating |

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

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

|  |
| --- |
| From editing the variables of objects in classes |

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

|  |
| --- |
| The SetMaxSpeed() stops bad values; such as negatives from being passed and assigned in maxSpeed. |

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

|  |
| --- |
| when the keyword 'private' is attached to a variable, only objects within the class are able to access the variable freely; all access outside are denied. |

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

|  |
| --- |
| the maxSpeed variable is in its own level |

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

|  |
| --- |
| SetMaxSpeed() is a public method so it can be pulled outside the class. SetMaxSpeed is in the Car class alongside maxSpeed, so changes can be made. |

* 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?

|  |
| --- |
| Using accessor methods prevents the codes from facing errors when handling cross-class actions. |

(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) ).

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

|  |
| --- |
| miscommunication between co-workers is common, and without full participation in dialogue, accidental errors will occur, access controls acts as a net for these errors. |

**VIDEO: Overloading**

* 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)

|  |
| --- |
| overloading is in reference to calling commands and overriding is in reference to inheritance |

* 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.

|  |
| --- |
| The situational factor is considered witht he 3 different printBox() method, consideration with scenario to scenario |

* 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?

|  |
| --- |
| The parameter's passing type would required a matching method and matching variable. |

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

|  |
| --- |
| Overloading makes sense when you have one [action], but a whole bunch of different [ways to call the action] |

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

|  |
| --- |
| The compiler can't distinguish the methods from each other on return value alone, so no. |

* 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?

|  |
| --- |
| In cases if the method can get complex, just call only one. |

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

|  |
| --- |
| Console.WriteLine, 19 different versions of it accepting different parameters, its a great example of an overlaoded method. |

**VIDEO: OOP: Constructors**

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

|  |
| --- |
| It will prevent the absence of an important instance variable |

* When does the constructor method run/execute?

|  |
| --- |
| The constructor method runs/executes when an object's created |

* 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?

|  |
| --- |
| The Car object obligates the user to pass a value in the absent parameter. |

* What two things identify a method as being a constructor?

|  |
| --- |
| When there are no return values made, and the name of the class and constructor match. |

* 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:

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  >>>>  }    class Car(int mS)  {  maxSpeed = mS;  }  public Car() :this(60)  {  }  }  Constructors that include the 'this' and a parameter of a separate constructor follows the respective parameters. |

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

* 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.

|  |
| --- |
| The new operator created is placed within an array stored in the heap that exists separately against eh call stack.  Each array slot is null by default. |

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

|  |
| --- |
| Null is a fancy way of saying its an empty value |

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

|  |
| --- |
| If a datatype has traces of a number, it wil be defaulted to value of 0; if elsewise, null. |

* 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)**

|  |
| --- |
| Line 1 creates an array of the Demo object, length of 5.  Line 3 assigns a new Demo object to position 1. |

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

|  |
| --- |
| ds[1].printbox(); calls the printbox() method with the ds object in the designated slot, in this case 1, of the ds array object. |

**VIDEO: Null References Within An Array**

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

|  |
| --- |
| An error will occur when you run the program on the call of ds[2]. ds[2] however is null in this space of the array. |

* 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]) ?

|  |
| --- |
| Null Pointer/Reference Exception |

* 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:

|  |
| --- |
| It will check if slot 2 and null are the same 'value'  if (ds[2] != null)  ds[2].printBox(); |

* 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:

|  |
| --- |
| It will check if slot 2 and null are the same 'value'  if(ds[2] == null)  ds[2].printBox(); |