**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: Inheritance: Conceptual overview**

* What is the purpose behind inheritance? (Give a very brief, intuitive explanation)

|  |
| --- |
| You have an existing class that literally inherits the base traits of the original class, but different. So a 'specialized class'. |

* It is bad to copy-and-paste the Car code and then tweak it in order to create a more specialized subclass/subcategory (such as, for example, a SpyCar). Briefly, intuitively explain one reason why this is bad / will introduce problems later on.

|  |
| --- |
| Copy and pasting the code for the Car code could become a pain later if you discover that the code you copy-and-pasted everywhere has a bug in it. Suddenly you're forced to play catch-around. |

* In order to tell C# that a given class extends another, existing class, what do add to the class declaration?

|  |
| --- |
| You use a colon. For the SpyCar class, for example.  class SpyCar : Car  {  } |

**VIDEO: Inheritance: vocabulary**

* A **SpyCar** has a copy of what?   
  (This will be covered in more detail in the next video)

|  |
| --- |
| A SpyCar is a copy of Car. |

* What are the two pairs of words we use to talk about the Car & SpyCar classes? Make sure that you’re clear which word describes Car and which word describes SpyCar, and which two words typically go together in which pair:

|  |
| --- |
| Superclass, subclass.  Base Class, Derived Class.  The Car is the base class, because it is the foundation of the classes they derive from.  The derived class is deriving from the base class, which is SpyCar from the base class. |

* If you have a class, then create a subclass of it, and you then create a further subclass of the firsdt subclass?   
  What advantage(s) would this give you?

|  |
| --- |
| There are many less variables to micromanage in your code if one class goes awry. So each consecutive class inherits all the previous classes' features 1:1, with changes as each sub-classification having specializing traits that distinguish it from the class they've derived from. |

**VIDEO: Inheritance: Inheriting data**

* What data fields does an instance of the SpyCar class have? For each field, explain where it came from / why the SpyCar has it.

|  |
| --- |
| The spy car class has a tankSize data-variable from the Car class. |

* Can a derived class directly access a data field that was declared to be private in a base class?  
  If not, then how does the derived class make use of it (explain this briefly and intuitively)?

|  |
| --- |
| The derived class cannot access a private base class.  It has a copy of the class from the class its deriving from, but not the real deal. |

* What does the C# keyword **protected** do? How is this different from **private**?

|  |
| --- |
| It's technically private but is able to be manipulated by anything that is inheriting data from the class itself, with specifications as such with the colon. |

* Can an instance of the **Car** class ever access the fields of the derived **SpyCar** class? Explain why this does (or does not) make sense, briefly and intuitively.

|  |
| --- |
| No. The car class knows it is running as a car class, but does not know the SpyCar class is running on the car class. |

**VIDEO: Inheritance: Constructors**

* In C#, how do you call a base-class constructor? If there are multiple (possibly overloaded) constructors in the base class, how do you indicate which one you want to use?

|  |
| --- |
| For the example in the video  public SpyCar(double ts, int sum) : base( ts )  If you want to overload a constructor with multiple constructor, just make the method with the parameters in the order you expect them to enter. |

* How can you set things up so that all instances of the **SpyCar** class always has, say, zero backseats.

|  |
| --- |
| You set up the constructor and call the constructor with just a bunch of 'throwaway values' to feed the parameters with.  So for SpyCars, when you'd only want smoke screens, they'd go like this  public SpyCar(int sm) : base( 20.0, 0 )  {  xxxxxx  }  Where the 20.0 represents the tankSize  The 0 represents the number of backseats  And the 'int sm' parameter in the SpyCar is the derivative of the base class that wishes to feed its own unique parameters. |

* If you don’t specify a base-class constructor (to use in a derived class’s constructor), which one will C# use?

|  |
| --- |
| If you don't specify a constructor, it will use the default constructor. |

* How can you provide a default constructor in the base class without duplicating code amongst the constructors?

|  |
| --- |
| You would need to create it like this after the constructor  Constructor : this(xxx, xxx)  {  } |

* Building off your answer for the previous question, what is a very typical pattern when a class provides multiple constructors?

|  |
| --- |
| One real constructor with a handful other constructors for convenience. |

* Why do you have to copy-and-paste the parameters for the derived class’s constructor(s)?

|  |
| --- |
| It's a completely separate constructor method. |

**VIDEO: Polymorphism: Overriding Methods**

* When you declare a method to be **virtual**, what does that mean?

|  |
| --- |
| When you declare something as virtual, you're enabling a 'specializable' object, and basically making it replaceable. |

* In order to override a method in a subclass, what do you need to copy from the superclass?

|  |
| --- |
| You need to copy the class' signature. |

* In addition to what was listed in the prior question, what other keyword do you need to add to the subclass’s method?

|  |
| --- |
| 'override'. |

* When **playerCar** is set to refer to the **c1** object, and you call **Print**, what happens? Why?

|  |
| --- |
| It will read the reference type, and then check for the Car it's subclasses. Checks for a Print(), and goes for the most specialized version of it; which is the base Print(); |

* When **playerCar** is set to refer to the **sp** object, and you call **Print**, what happens? Why?

|  |
| --- |
| It uses the specialized spyCar version of Print() because the playerCar refers to the spyCar. |

* What is the name for this technique?

|  |
| --- |
| Polymorphism. |

* In C#, if you wanted the **SpyCar** to invoke (to call) the base-class version of **Print** (i.e., **Car.Print**), what snippet of C# code would you use?

|  |
| --- |
| base.Print(); |

* What is the name of the technology that allows (parts of) your C# program to run in parallel?

|  |
| --- |
| Threads... the act is called concurrency... |

**VIDEO: Polymorphism: An Array Of Objects With An Overridden Method**

* What output will be produced by the loop that iterates through the **garage** array and calls the Print method on all the non-null slots? Please summarize briefly in English what will be printed, and then make sure to explain WHY this output is produced.

|  |
| --- |
| Car tank size: 200  Car tank size:30  thisis a SPYCAR, the tank size is 400  Car tank size:400  This happens because the objects are being polymorphed from the class they originate from.  For the garage[i].Print(), they call the object's most 'specialized' Print() method; for lack of better phrasing.  The first one is the Car object's tank size being printed from the 200  The second is the Car object's default tank size being printed, which is 30  The last one is the polymorphed Car object that is SPYCAR, and it prints its' unique Print() |

* When adding objects to an array, are you required to create a new object for the array? If not, how would you add, say, the **c2** car object into the array

|  |
| --- |
| Not necessarily. You can use a pre-existing car object like c2 into the array.  Car[] garage = new Car[xxx];  garage[i] = new Car(xxx, xxx);  Car c2 = new Car(xxx, xxx);  garage[i] = c2;  garage[i] = null;  These are all valid. |

**VIDEO: Named Constants**

* What is a “magic number”?

|  |
| --- |
| A secret magic number that takes 3 tries to be guessed before the game is guessed. |

* In terms of printing out a magic number, what is a better alternative to simply printing the number directly?

|  |
| --- |
| Replace the number like this  Console.WriteLine("Haha, this is a {0}", 4);  pretend the 4 is a variable. |

* What modifier can be used to mark an integer (or double, or short, or char, etc) as a named constant?

|  |
| --- |
| const |

* What is the “readonly” modifier used for?

|  |
| --- |
| This is used for reference variables. |

* What happens if I try to change the value of a named constant (after it’s been created)?

|  |
| --- |
| You will get an error. |

* How do people (traditionally) capitalize named constants?

|  |
| --- |
| all of it is in caps  LIKE\_THIS\_VARIABLE |

* Where do you normally put named constant declarations?

|  |
| --- |
| You put them before the methods. |

* Copy over the C# source code that declares a new class named ComputerScreen, including it’s two named constants:

|  |
| --- |
| class ComputerScreen  {  const public double WIDTH = 10.5;  const public double HEIGHT = 14.0;  } |

* Copy over a snippet of C# source code that will make use of the two constants in a class that’s separate from the ComputerScreen class?

|  |
| --- |
| static void Main(string[] args)  {  Console.WriteLine("The width of hte screen is {0}", ComputerScreen.WIDTH);  Console.WriteLine("The height of the screen is {0}", ComputerScreen.HEIGHT);  } |