**1) What is a class? According to the book, what does a class “arrange?”**

- A class is a blueprint that is used to create instances or objects at run time. According to the book, a class “arranges” information and behavior into a meaningful entity.

**2) What are the two purposes of encapsulation?**

-The two purposes of encapsulation are to combine methods and data within a class and to control the accessibility of the methods and data.

**3) How do you instantiate an instance of a class? How do you access that instance?**

-You instantiate an instance of a class by using the *new* operator. You access that instance using the *public* and *private* keywords. YOU USE THE DOT ACCESSOR

**4) What is the default access of the fields and methods of a class? How do you change the default?**

-The default access of the fields and methods of a class are *private*. You can change the default by using the *public* keyword in place of the *private* keyword.

**5) What is the syntax for writing a constructor?**

-The syntax for writing a constructor starts with the accessibility keyword *public*, followed by the name of the method. The name of the method needs to be the same name as the class that it is located in. You can include a parameter list in the method signature and the method does not include a return type.

**6) What is the difference between class fields and methods, and instance fields and methods? How do you create class fields and methods?**

-The difference between class fields and methods is that class fields are the variables that are stored inside a class. They can be accessed by creating an object of the class or by using the dot syntax. Methods are class members that are used to perform certain actions and can define how an object of a class behaves. Methods can also be accessed with the dot syntax, but the method must be declared as *public.* Instance fields are specific to an instance of a type. An instance field (or Instance variable) represents the data of a class that enables an object to maintain its state. It is used to encapsulate the data that must be accessed by all the methods of a class and allow the data to be stored throughout the lifetime of the instance of the class. Instance methods are declared without the *static* keyword and are used for polymorphism. Instance methods in a class can make use of any of the public or private static data that belongs to that class. They can also make use of static data from other classes, provided that it is accessible. You can create class fields and methods by declaring them within the class to which they belong.

**7) How do you bring a static class in scope? Why would you want to bring a static class in scope?**

-You can bring a static class into scope with the *using* statement. You would want to bring a static class into scope to save on allocated stack memory. In a static class, no memory is allocated, it is only done so when it brought into scope to be used and the memory is freed up once brought back out of scope.

**8) Can you think of a good reason to create an anonymous class? What is it?**

-A reason to create an anonymous class is for a one-time operation. They are designed to be used from within the scope in which they are created. No other scope should know about the anonymous type’s definition. An application for an anonymous class usage that I found on the internet is when you need to process two separate collections. You can zip them together and protect the result over an anonymous type that only exists within the scope of the LINQ query.

**9) What is polymorphism as this term is used in computer science? This is not in the book.**

-Polymorphism is the third pillar of object-oriented programming after encapsulation and inheritance. Polymorphism has two distinct aspects:

-At run time, objects of a derived class may be treated as objects of a base class in places such as method parameters and collections or arrays. When this polymorphism occurs, the object's declared type is no longer identical to its run-time type.

-Base classes may define and implement virtual *methods*, and derived classes can override them, which means they provide their own definition and implementation. At run-time, when client code calls the method, the CLR looks up the run-time type of the object and invokes that override of the virtual method. In your source code you can call a method on a base class and cause a derived class's version of the method to be executed.

**10) What is message passing as this term is used in computer science? This is not in the book.**

-Message passing is a way that parts of your program can communicate with each other. With message passing, each object acts as an independent cell. The only thing an object can do to communicate with other objects is to send a message.

**11) What was the first object-oriented programming language?**

-The first OOP was called Simula.

**12) Who is Alexander Stephanov? What do you think about this quote?**

-Alexander Stephanov is a Russian-American computer programmer, best known as an advocate of generic programming and as the primary designer and implementer of the C++ Standard Template Library. I would be inclined to agree with the quote because he is basically stating that with OOP you are breaking down entities to exist with a single type. This leads to the belief that things can only exist in a binary application; either it is, or it is not. In reality things are multifaceted. For instance, what is a cell phone? By definition it is a phone, but it has evolved to include the implementation of several internal radios that handle the various communication requirements. Since things exist that are too complex to be broken down to a classification of “ones and zeroes”, you do need multi-sorted algebras. The rest of the quote that is not displayed on the homework page discusses the use of axioms and their use in mathematics. Mathematicians start with proofs and when you have related proofs only then can you create axioms. OOP is backwards from this. It starts with the axioms and is asking you for the proofs.