**Oriette Ramos**

**Lesson Plan 17**

**ISTA-220**

1. What is the difference between a managed resource and an unmanaged resource?  
   **Managed resources are those controlled by the CLR. Unmanaged resources need to be managed by the programmer.**
2. When is memory for an object (reference type) allocated? When is the memory deallocated?  
   **The memory for an object is allocated when the object is initialized. The memory is deallocated when all references to the object are no longer in scope (although the exact time the memory is freed is up to the garbage collector).**
3. What is a destructor?  
   **A destructor is a special method, a little like a constructor, except that the CLR calls it after the reference to an object has disappeared.**
4. What is the difference in syntax between a constructor and a destructor?  
   **The constructor for a class uses an access modifier, the class name, and parameters, whereas the destructor uses the tilde and the class name. For example:**
5. class Bob
6. {
7. public Bob()
8. {
9. //this is the constructor
10. }
11. ~Bob()
12. {
13. //this is the destructor
14. }

}

1. Give some examples of scarce resources. Why would you want to manage scarce resources?  
   **Examples of scarce resources include file streams, network connections, database connections, and memory. You want to manage resources to increase performance and to prevent errors caused by a resource not being available. If the resource is a file, not releasing it could prevent other users from being able to access that file; if the resource is a database connection, your application could prevent other users from being able to connect to the same database.**
2. What is exception-safe disposal?  
   **Exception-safe disposal releases resources even if an error occurs.**
3. How do you think that the using statement works for resource management? Give an informal, English language, explanation of how it works.  
   **The using statement sets a boundary in which the resource can be used. Once the code leaves the using block, it disposes of the resource.**
4. What ill effects could result from attempting to dispose of a resource more than once?  
   **You will not compromise the security of your computer, but you might affect the logical integrity of your application if you attempt to dispose of a managed object that no longer exists.**
5. We will look at threads later in the term. For now, what is your understanding of how threads interact with resource management? A good guess is a sufficient answer to this question.  
   **Threads allow more a multithreaded computer to process more than one thing at a time. As far as resource management is concerned, this could lead to two threads attempting to access the same resource (to read, update, or delete it) simultaneously. Without proper protection in the code, this could lead to inaccurate data, data corruption, or other errors.**
6. Why does the book recommend not attempting to force the garbage collector? Are their any exceptions to this recommendation?  
   **The GC.Collect method starts the garbage collector, but the process runs asynchronously—the GC.Collect method does not wait for garbage collection to be complete before it returns, so you still don’t know whether your objects have been destroyed. Let the CLR decide when it is best to collect garbage. Possible exceptions could include when you know a large number of objects have come out of scope and a small performance hit would be less noticeable. For example, when a large form is closed.**