ISTA-320, Cohort 3

C# Step by Step, Chapter 24 (to page 575)

October 22, 2017

Asynchronicity

1. What is an *asynchronous method*? When the book talks about a contract, what is the contract and who is it with?
2. What can be the problem with decomposing a series of discrete method calls into a set of tasks, such as we saw in chapter 23?
3. What can be the problem with decomposing a series of discrete method calls into a set of continuations? When does the last continuation "complete" as compared to the previous continuations? What problem might this cause?
4. What might be the problem with implementing te previous solution as a continuation passing a delegate? What would be your interpretation with this error message: "The application called an interface that was marshaled for a different thread."?
5. The book suggests a solution using a continuation delegate calling another continuation delegate via an anonymous function. What does the book ientify as a problem with this suggested solution?
6. What does the *async* modifier do? What does the *await* operator do?
7. What is an *awaitable* object? Be specific.
8. In a method definition, how do you create and run a *Task* and return a reference to the *Task*? What is the type of such a method? What does the method return?
9. How do you define method calls in the implementation of an *async* method? Specifically, you must define a task, you must run the task, you must implement the task, and you must await the task. What is the syntax for doing this?
10. What is the difference between decomposing a series of method calls that do not return values, and a series of method calls that return values? What is the *Result* property of a method that returns a value? How do you use the *await* operator in this circumstance?
11. What is the difference between the *await* operator and the *Wait* method?