CSE143 Lecture Questions for Wednesday, 4/7/21

Question	Answer
Can you explain a little more on why we are using LinkedList<E> for implementing Queue<E>? That makes sense! Thank you so much!	We need to use some implementation. That just happens to be the simplest one to grab. When we discuss linked lists next week, it will be a little more clear why a linked list provides fast access for adding at the back and removing at the front.
Is there ever a situation where you *don't* want to keep track of/update size in the constructor? For example, do it elsewhere or not do it at all? Is part of it also that taking care of that in the constructor is safer/more efficient? Ok, thanks!	You want to know that size will be set appropriately when an ArrayIntList is constructed. We do it in the constructor, but we could also have relied on the default initialization of the field. But you'd always want to know that it would be set properly no matter what. I just do it for readability.
I'm a little confused about why when we set size = 0, it clears the arraylist. I thought that we updated the size manually, so wouldn't the arraylist still have values? Oh that makes so much sense now. Thanks!	The size field is used to keep track of which elements of the array are currently part of the list. By resetting to 0, we say that nothing is in the list. There might be old values from previous calls on add that are stored in the array, but the client has no way to access those and they will be overwritten if the client makes new calls on add.
Hi Stuart: Homework 2 will be posted on this Friday is that correct? And will all homeworks in the future be posted on Fridays? Cause I remember you usually post the homework on Wednesdays during 142 last quarter. Just want to make clear about that. Ok thanks. I have not yet switched from the pattern of 142 yet. Haha :) Thanks!	Most will be posted on Fridays, including homework 2. Homework 8 sometimes goes out early.
Does HW8 sometimes go out early because it's harder than usual? On a similar vein, what does the homework look like in terms of difficulty/length as the quarter goes on? Does it match the difficulty of the content throughout the quarter? Hm ok. Thanks!	It's worth 30 points, so it's longer. I'm not sure how to answer that. The assignments in 143 tend to be more conceptual than detail oriented.

Why are interfaces favored over using classes by themselves? Ohhh that makes sense. Thanks!	Using interfaces leads to more flexible code. For example, suppose that you wrote 500 lines of code for manipulating a List<String> and in one of those lines you called new ArrayList<String>. If you wanted to switch to a LinkedList<String>, you would only need to change it in one place because if you use the interface, it can refer to either kind of list.
Could you clarify what "conceptual" means when you describe 143 in comparison to 142? I see so is it like we'll be looking at things from a more higher-level/big picture view rather than focusing on the grammar of it? Ok, thank you.	There are a lot of details associated with things covered in 142. Just think about classes, for example. There are a lot of syntax details to know about constructors, fields, the implicit parameter, etc. We're going to cover something called recursion that in a sense can be covered in 10 minutes. No details. But it takes a while to learn how to use recursion well. The concept of recursion takes time to learn, but it has few details associated with it. It's not really big picture. It's just different kinds of things to learn. Some things have a lot of details associated with them and other things are just challenging concepts.
Are reference semantics similar to variable scope?	No. Different concepts. Both important.
You mentioned earlier that a List can do everything that a stack/queue can do. What does that mean? I see. Can stacks and queues only have integers? Or any data type? Ok thank you.	We know that we can insert and remove values in the middle of a list. You can't do that with stacks and queues. There are many other things you can do with lists that you can't do with a stack or queue. No, they can store any type of data. It's Queue<E> and Stack<E>.
Does code efficiency and simplicity fall under the category of style? I see, like Boolean Zen. So should we prioritize code efficiency/simplicity outlined in the style guide and hw spec over simply trying to have overall efficiency? Ah ok interesting. Thank you!	Those are big topics potentially. We have various style issues that relate to efficiency and simplicity. For example, we take off points if you use extra data structures that you don't need or if you write loops or if/else structures in an overly complex way. We don't tend to worry about overall efficiency unless we have specifically mentioned it. For example, homework 2 will have a detail you have to get right for overall efficiency.

Is linkedList an implementation of Queue?	LinkedList<E> is a class that has many properties. One of those is that it implements the Queue<E> interface. We're going to talk about linked lists next week.