CSE143 Lecture Questions for Friday, 4/9/21

| Question | Answer |
|---|---|
| That was an awesome piano arrangement!<br>The one you played! But I guess both! | Haha...thanks. Or are you referring to what Kevin did to make it easier for piano players? If so, I agree. |
| Thinking back to our earlier classes, how did the ArrayIntLists end up shifting their sizes so dynamically? If we removed an element in the middle for example, all the other elements would get shifted back by a value of one. I'm confused because I don't remember ever defining an instance method for that in lecture. I only remember methods like add, set, and remove. Am I missing something?<br><br>OK so it was already implemented in the class but we just didn't cover it in lecture? Because i thought we were building the ArrayIntList from scratch. Okay that makes a lot more sense. I thought I missed an entire portion of the lecture or something. Thanks! | If you're talking about this quarter, I covered the appending add in lecture but I didn't discuss adding at an index (in the middle) or the remove method, both of which require shifting. I had that as part of the first section, although I think most TAs didn't have time to cover it because they were doing ice breaker activities. I briefly mentioned it in the beginning of Wednesday's lecture last week, including the fact that add is harder to implement because shifting right is harder than shifting left. Chapter 7 discusses both in detail.<br><br>Yes, I didn't have time to cover it in lecture. I referred people to chapter 7 for a more detailed discussion. |
| Is there a reason to use interfaces besides having flexibility in parameters?<br>Got it, thank you! | It's flexibility for everything, not just parameters. There are other good reasons. For example, sometimes you want to optimize a bit of code and you find that you can do that by replacing some class with a different version of it. If someone has written code using a specific class like ArrayIntList, it is harder for the optimizer to replace it. If you have used the less specific IntList, then it is more likely that you can replace it with some more efficient version. |

| | |
|---|---|
| I have another question about one of the earlier concepts about the iterator. Specifically, I'm having trouble understanding this statement: Iterator<Integer> itr = list.iterator();<br><br>Since the iterator's constructor is defined under the iterator object class, how does list.iterator() work since technically the object 'list' can't call for this constructor and only the iterator object can? Moreover, we defined the iterator constructor to take in a parameter Arrayintlist list and we didn't pass anything into the above statement. How does this even compile?<br><br>Ok so essentially, since the iterator is a subclass of the ArrayIntList, this works? Alright I thought that since every object came with an iterator, it would be a subset. Okay, understood. I think I'll get a better understanding after going through the notes again. Thanks for answering! | The line of code you have pointed out is from the client code. The client asks the ArrayIntList to give it an iterator. It's the ArrayIntList class that constructs the iterator. It does so with this line of code:<br><br>return new ArrayIntListIterator(this);<br><br>I mentioned that this is an interesting use of the "this" keyword to indicate which ArrayIntList the iterator should be iterating over.<br><br>No. The ArrayIntListIterator class is a separate class from ArrayIntList. Anyone can construct an ArrayIntListIterator when it is done this way, including an ArrayIntList.<br><br>It's not true that objects come with iterators. They have to be implemented, as we were doing with ArrayIntListIterator. It can be confusing because the built-in classes like ArrayList<E> and LinkedList<E> have iterators, but that's because programmers wrote the code to make that work (code like our ArrayIntListIterator). |
| At this point in time, should we know the characteristics of ArrayIntList vs LinkedIntList? Or should we just be aware that they're somewhat different? Like HashSet vs TreeSet etc<br>Cool. Thank you! | You don't have to understand the differences between these structures. We will spend all of next week talking about how LinkedIntList is implemented. |
| Is this homework assignment some form of a sound synthesizer? I'd really like to see the code behind the conversion from the numbers to sound, that sounds like something pretty cool to see.<br>Yeah. It's shockingly similar to actual acoustic guitar sound.<br>Oops. HarpsichordLite doesn't have the same ring to it. | It simulates a musical instrument. I guess that counts as a synthesizer. It is sending simple signals to the sound card. I include some links if you want to read a bit more about all of this. I find it fascinating that such a simple approach produces such beautiful sounding music. Kevin Wayne admitted during his presentation that this is really more of a harpsichord than a guitar.<br><br>Yeah, I don't think HarpsichordLite will sell well. :-) |

| | |
|---|---|
| When we use an interface to create an object and assign it a variable, and then later assign that variable to another object, does the information automatically transfer over? For example, when we changed list1 to equal LinkedList after being an ArrayIntList, is the new LinkedList empty or filled with the same info? Ok, thank you! | We called new on a LinkedIntList, which returns a brand new (and therefore empty) list. |
| I'm confused as to what the goal of the homework is. Are we making a class of 37 notes or are we making an interface that will be able to like accept other classes that code for notes? | In the second part of the assignment you are writing a class called Guitar37 that is an implementation of the Guitar interface. It keeps track of 37 different GuitarString objects. |
| I'm confused on how you ran the Guitar Hero program. Is it possible for us to run it because I am getting errors when I compile the program? Oh, right. Thank you | You have to finish the first part of the homework first, which is to implement a GuitarString class. Then you will be able to run GuitarHero with GuitarLite. |
| How are random numbers able to produce specific notes? Like, shouldn't a specific note have specific frequencies and sound waves? I'm confused about the part of the spec where it says that a queue of random numbers are able to produce specific sounds…? Does my question make sense?<br><br>I'm starting to make sense of the homework spec now! Thank you! | Your question makes perfect sense. I had the same confusion and, to some extent, I still find it confusing. That's why I said that at first I didn't think I understood what I was supposed to do. But I followed the instructions and it worked. I think that the physics of sound and how the ear perceives it is not a simple thing. You need to have the white noise pattern for the ear to be able to recognize it. The frequency with which you play that short randomized sequence determines the pitch that the ear perceives (high pitch for short sequences that are produced at a higher frequency and lower pitch for long sequences that are produced at a lower frequency). Make the leap of faith and just do what the writeup says and you will be pleasantly surprised (as I was) to find that it works even though I don't entirely understand why. |
| What is the main purpose of using Point[]? | I gave that as an example to review some issues related to working with an array of objects. It's different than working with an array of primitive values like an int[]. You will be working with an array of objects in the homework, so it seemed useful to review these issues. |

| | |
|---|---|
| What's the difference between ArrayIntList and LinkedIntList interfaces? | They aren't interfaces. Those are classes. In the lecture we defined an interface called IntList that they both implement. We have talked about how ArrayIntList is implemented in these first two weeks. Next week we will discuss how LinkedIntList is implemented. |
| Love the analogies! Really helpful and makes me chuckle. | thanks |
| WOAH! That was so cool??? Never would have expected that I would hear sounds from JGRASP??? How does that even work!! | I'm with you...it's still a bit of a mystery to me that it sounds so good. |
| At 10:08 isn't it possible to add both the ArrayIntList and the LinkedIntList as parameters in the processList() method?<br><br>Yeah, I thought by having two parameters it will allow the method to be compatible with both types of lists. Not true?<br><br>Makes sense. Thanks! | Are you suggesting two parameters? That would be odd. We're trying to write a method that acts on one list. And what if you had two LinkedIntList objects?<br><br>Not true. How then would you know which list to use? And how can you call it passing it just one list? If you knew you had exactly one ArrayIntList and exactly one LinkedIntList, then two parameters of those types would work. But that's not what we have here. |
| Wow that piano result is so cool! | Yes, it's a pretty cool program once you finish it. |