CSE143 Lecture Questions for Friday, 1/7/22

Question	Answer
Over the last few classes, we have gone over building the class code. How to build it, how to comment, and throwing out exceptions. There has been some somewhat new content. Would you mind explaining why we created an ArrayIntList when we could have just used the ArrayList<element> class?	There are two different issues here. Obviously we want to be able to use built-in structures so that we don't have to write all of the code ourselves. Software engineers do that often. But we also want to understand how to build such a structure because we might be asked to do that. So we want to understand both sides of this. In today's lecture I talked about how to use ArrayList<E>. You will do that often as a Java programmer. But you also want to know how to create such a structure, which is why we were studying ArrayIntList as a way to explore how such a structure works.
Could you give some examples where you use interface types in the 4 ways thanks!	I talked about the List<E> interface. If we write a program that uses a list and we declare all of our variables, parameters, etc, to be of type List<E>, then we could have that code work with either an ArrayList<E> or a LinkedList<E>. If we had written it to use an ArrayList and then found out later that LinkedList is a better structure to use, then we have to change just the call on new where we construct the object. The four ways would be for a variable: List<E> list = …; As a parameter: public void doSomething(List<E> list)... As a return type: public List<E> doSomethingElse()... And as a field: private List<E> data;
Does the "E" in List<E> stand for something?	Java doesn't say anywhere what it stands for, but I'm convinced it's meant to be short for "Element" or "Element Type." The one thing you will never use inside the <> is "E" itself. It's a placeholder that is meant to be substituted with an actual type like String.

How are we going to test? Are there going to be multiple choice questions or are we going to build classes/client code to run? If we are to do the second, are we expected to comment out code? Thank you!	We have some questions that are mechanical in that you fill in a blank ("What output is produced…"). Most of the questions will be short programming questions. The section problems we go over are a great example of the kind of question we'll ask. Many of them are old exam questions. We'll also give you several sample exams. There shouldn't be any surprises about exactly what will be on the test.

In this code excerpt why is the interface not used/should it be used?
 	
Ah, I see. I will finish it then.
 	
`// wrapper classes: Integer, Boolean, Double`
`int[] data = {18, 4, 97, 3, 4, 18, 72, 4, 42, 42, -3};`
`ArrayList<int> numbers1 = new ArrayList<>();`
`numbers1.add(18);`
`numbers1.add(42);`
`int product = numbers1.get(0) * numbers1.get(1);`
	You have a sharp eye. Congrats! It should be used. I believe I fix that a few minutes later in the lecture.
	
	If we had a live lecture, you probably would have raised your hand to ask, which would have been great.

Would calling the ArrayList "list" also be bad style since it's not an abstraction of what the list represents? To follow up, would listOfNumbers then include unessential information?	It's good to have a variable name that indicates what an object is used for. You don't want to abstract away everything. For example, I said that a method called drawBase might have been helpful for the Space Needle program from 142. We wouldn't have wanted to call it "drawSomething." I guess the key is to figure out whether something is an essential property versus an unimportant detail. Knowing that something is a list seems like useful information. listOfNumbers might be a good name. It's somewhat of a judgement call.
Why exactly is declaring a class constant as public better than private? Will we lose style points for writing it as "private"?	The issue has more to do with what we want the client interface to be. I think that for ArrayIntList, a client might want to know the default capacity. In other cases we might have a constant that we wouldn't want to share with the client. We will be clear whether a constant should be public or not.

Can sets store ArrayLists/other forms of lists? If so, what does the 'default' treeSet ordering look like? And HashSet does not provide a 'natural' ordering, correct?	Sets can store any kind of data, but the TreeSet is limited in what kind of data it stores. If you wanted to have a set of lists, you would have to use the version known as HashSet. That's right. HashSet does not put things into any specific order.
After declaring a field, is it okay to initialize it in the constructor? I read in the course website that its generally not done?	We prefer that you initialize fields in a constructor. The thing we want you to avoid is initializing the field outside of a constructor.
In the HW 1 spec sheet, it is not mentioned whether we need to use public or private class constant… so what should we do?	The constant should be declared public.
Should all variables in the constructor be declared at the start of a class?	Some programmers put fields at the end of the class (that's more of a C++ convention), but you should generally group all of the fields together. Java programmers tend to put them at the top of the class.
In the lecture you mention that the length of an array is a field and that it is a constant. Is it a private field? Then what kind of field is it? So would it be implemented to be something like this: public static void final LENGTH; And initialized in the constructor? Got it, thank you.	No, it's not private because we refer to it often when writing array code, as in: for (int i = 0; i < array.length; i++) … It's a field that is declared to be "final" like what we have seen with the declaration of a constant. It's a field, so it would look like this: public final int length; And it would be initialized in the constructor, as you have suggested. But the array class is a special case because it is a built-in part of Java.
is it okay to put parentheses around size == 0 to fix boolean zen... like: return (size == 0)	Yes, that's a matter of personal choice whether to have some extra parentheses.
can we use Character.isLetter() in HW1?	You can use anything in the Character class that is not listed as forbidden on the general style sheet.
Will the Array -> ArrayList translations be posted?	They are already posted to the calendar for today (ListSetExample.java).
I'm guessing we don't need to study / memorize the history of Java and C#, right?	That's correct.

You mentioned that Java updated its code to "Auto-wrap integers" when creating them. If this is true, why is it necessary to Still use Integer when initializing an object (List<Integer> = new ArrayList<>();) So java 8 can unwrap but not wrap?	The answer is rather complicated. The short version is that they weren't planning for generics when they designed Java. What they did doesn't allow them to support ArrayList<int>. So it's a compromise. They want ArrayList<Integer> to almost act like ArrayList<int>. Java 8 includes autoboxing and autounboxing, so it does both.
Hi,Stuart. Some questions about the HW1: are we allowed to use private constructor? And I was a little confused about the relationship between private method/field and class, Is that true that private class is not accessible to the client, but can be used by the class we wrote. Thanks!	You can introduce whatever private constructors and methods that you want to. Private fields and methods can't be accessed from the outside by a client, but they can be accessed by objects of the same type. So one instance of the ArrayIntList class can access the private fields and methods of another instance of the ArrayIntList class.
In homework 1, we used a class constant to construct the count array. When we traverse through this count array, should we use the class constant, or count.length?	I tend to use the array length, but either is okay because they will match.
I am aware this class puts an emphasis on commenting without details about implementation. However, in cases where a method returns a void, and instead changes a field (an implementation) how do we comment on the behavior of these methods. (example count[] getting changed in homework 1 in the set method]	You have to keep in mind the abstraction that the client is asked to work with. In the case of ArrayIntList, we talk about changing the list of numbers and we don't talk about changing elementData and size. For LetterInventory, the client knows that we have a structure that is keeping track of counts for various characters, so you can talk about it that way without making a specific reference to how those counts are stored internally.
You say Iterator<String> = list.iterator() without an identifier. Was this a typo? Nevermind, you corrected it.	Haha…yes…I made a silly mistake. Too bad we weren't in a live lecture because then you could have raised your hand and helped me to fix it earlier.

Hi, Stuart, can you explain the interface type a little and what is the key difference with normal types? Thanks! If I understand correctly, interface types is a more general thing that can include any specific type. For example the idea of accountant can include all different types of accountant that have different skills, and they are all called accountant? **Thanks! Get it!Have a good day! Thanks**	We're going to talk about this more next week, but I'll say a bit. An interface is a list of behaviors that an object must implement in order to be considered of that type. For example, you might be someone who has studied enough to be a certified accountant. That means you're of both types. Are you a student or an accountant? Both. That's sort of right. Let me bring it back to Java. There are many different ways to store a list of values. We've seen the array version and in a week we'll start studying another approach known as a linked list. So that's two different kind of list objects, but they're somewhat the same in that they perform the same operations. This may sound like a weird analogy, but it might be more like an accountant who uses an adding machine versus an accountant who uses a spreadsheet. In some sense you don't care as long as they get the work done. Both kinds of list are lists and both kinds of accountants qualify as an accountant.
I'm wondering how can I know the abstraction of a class, like List is the abstraction of ArrayList. Got it. Thank u!	An interface is a way to abstract away unimportant details, but we don't always use an interface. This is something that you learn to do with practice over time. It takes a while to learn how to do it well.
Are we online or in person next week?	I will email the class about this later today, but I expect to be in person but also working hard to support those who want to (or need to) be online.
For the lecture because we are using Set<E> interface can we assume that or would it be a style thing we don't used the SortedSet<E> interface when using a treeset because treeset guartness ordering and the SortSet<E> provided by java can be a interface for TreeSet	I will talk about SortedSet later in the quarter. For now we'll use Set<E>.
When we use the for each loop based on lecture doesn't java call the iterator because it implements Iterable<E> so does doing a for each loop same thing as doing a iterator	It turns out that the foreach loop is implemented using an iterator, but they are still different approaches. You can't change a structure using a foreach loop, but you can with an iterator.

Also wanted to clarify because TreeSet used ordering can we assume everything we work with has some sort of Comparable to it like we can't just says Set<News> news = TreeSet<>() or does java know how to do the ordering	For now we will be working with types that have ordering. We'll eventually talk about working with other types.
If I am working with generic types and I get the following error or warning is this a style thing. I also try to get around by attempting to cast Note: LetterInventory.java uses unchecked or unsafe operations. Note: Recompile with -Xlint:unchecked for details. **Wanted to clarify more if runs and passes test is okay to ignore or can I suppress those warnings or style deducation**	That indicates that you have used generics incorrectly somewhere. There is no reason to be working with generic types for homework 1. If you have warnings like that, then you likely have a style issue, but you have a bigger issue that you shouldn't need generics for homework 1.
It's a question from our lecture on Wednesday. Does it mean that there's a 'pre' only when a specific method needs to throw some kinds of exception? Ok, and if there's no 'if' in a method, I can definitely say this method has no precondition So the add and subtract method in HW1 also have preconditions since they cite a 'LetterInventory other'? Okay. Thanks!	No, there might be a precondition even if no exception is thrown. For example, there is a method Arrays.binarySearch that has a precondition that the array is sorted, but it doesn't throw an exception because it would be too costly to verify the precondition. No, the precondition is something that you reason about. For binary search, it needs a sorted array. It doesn't matter what code you write for it in terms of whether it has the precondition. You have to reason about whether a given method we have asked you to write has a precondition. You should be thinking about whether there any assumptions that the method needs to make in order to complete its task.
Okay thanks got it so you told me to to not use generic types so doos this mean it's safe to make an assumption that LetterInventory only works with one type and the client would only use it for one data type or should I make other assumptions these were some assumptions I made along with others that spec does not clearly label	LetterInventory only works with strings and characters.