ECE 2574: Data Structures and Algorithms - Sorted List ADT

C. L. Wyatt

Today we will look at the sorted list ADT and different ways it can be implemented.

- Sorted List ADT
- AbstractSortedList
- Adapting List ADT
- Reusing ArrayList via inheritance
- Reusing ArrayList via composition

Sorted List ADT

A number of objects, not necessarily distinct but of the same type, sorted by thier value.

```
+isEmpty(): boolean
+getLength(): integer
+insertSorted(newEntry: ItemType): void
+removeSorted(entry: ItemType): boolean
+remove(position: integer): boolean
+clear(): void
+getEntry(position: integer): ItemType
+getPosition(entry: ItemType): integer
```

These methods differ from the List ADT

- `+insertSorted(newEntry: ItemType): void`: insert the entry in order
- `+removeSorted(entry: ItemType): boolean`: remove first occurance
- `+getPosition(entry: ItemType): integer`: get position of first occurance or the negated position where it would be

Lets define an interface

See code `abstract_sorted_list.h`

Adapting List ADT

Note that SortedList ADT is very similar to the List ADT.
Can we just reuse the List ADT?

- Using inheritance: a SortedList is-a List
- Using composition: a SortedList has-a List

Reusing ArrayList via inheritance

We adapt DynamicArrayList by private inheritance.
See code `array_sorted_isa.h`

- Modifications to the Abstract List

See example code.

Reusing ArrayList via composition

We use DynamicArrayList as a private member.
See code `array_sorted_hasa.h`

Implementing insert

See code

Implementing getPosition

See code

Implementing remove

See code

Next Actions and Reminders

- Read CH CH pp. 352-357
- P3 is due Tuesday 10/31.