Lecture 12: Collections and Generics

CS 2110

March 3, 2026

Today's Learning Outcomes

52. Describe the differences between *data structures* and *abstract data types*.

53. Implement a generic class or method with one or more generic type parameters. Use generic classes in client code.

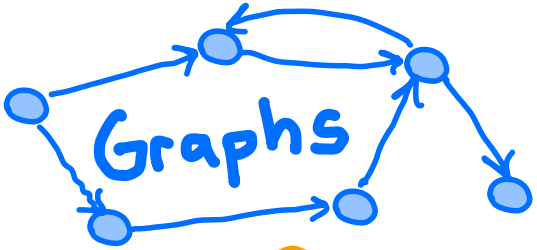
54. Describe the semantics of *auto-boxing* and *auto-unboxing* and identify where they happen in a code snippet.

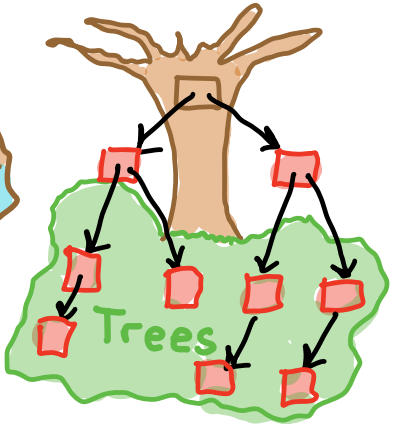
56. Compare and contrast the behaviors of ordinary Java arrays and dynamic array types such as Java's `ArrayList`.

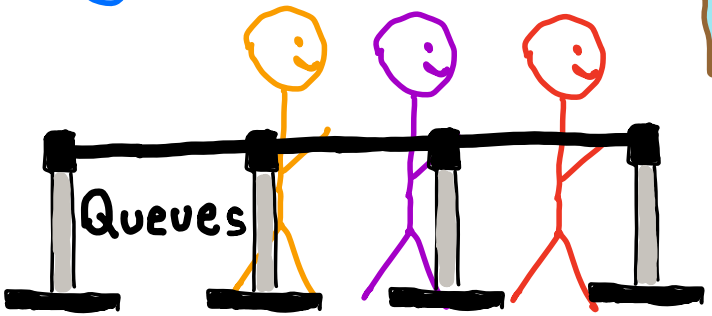
Arrays as Collections

Other Collections

Each has its own
- structure / state representation
- behaviors / invariants
- performance guarantees







Abstract Data Types vs. Data Structures

Client and implementer interact with collections in different ways. (abstraction barrier)

Client:

Implementer:

The List ADT

A list is an ordered collection that grows to accommodate an arbitrary number of elements. Its elements are accessible by index.

Coding Demo: `StringList` Interface

Using StringList as a Client

Use the `StringList` methods to complete the definition of the following method:

```
/** Replaces all instances of the given `word` with
  * "****" in these `lyrics`. */
static void censor(StringList lyrics, String word) {



}
```

StringList
add(String elem): void
insert(int index, String elem): void
size(): int
get(int index): String
contains(String elem): boolean
indexOf(String elem): int
set(int index, String elem): void
remove(int index): String
delete(String elem): void

Generic Classes

What if we want a list of something other than Strings?

Coding Demo: Generic `CS2110List`

Using Generic Types

Generic Types can take the place of a type name almost everywhere in a class.

- Fields

```
private T elem;
private T[] storage;
```

- Local variables
- Parameters

```
public void add(T elem) { }
```

- Return types

```
public T get(int index) { }
```

Brief Aside: Auto-(un)boxing

Generic type parameters can only be assigned reference types, not primitive types.

What if we want a list of ints?

The Dynamic Array Data Structure

How can we use arrays to model a list?

Need to handle unbounded size carefully.

Coding Demo: DynamicArrayList Design

invariantSatisfied() Methods

Data structures often rely on intricate class invariants to achieve good performance and ensure correctness.

Recall: Class invariant must hold at start/end of every public method call.

Coding Demo: `DynamicArrayList` Methods

Space Complexity of `DynamicArrayList`

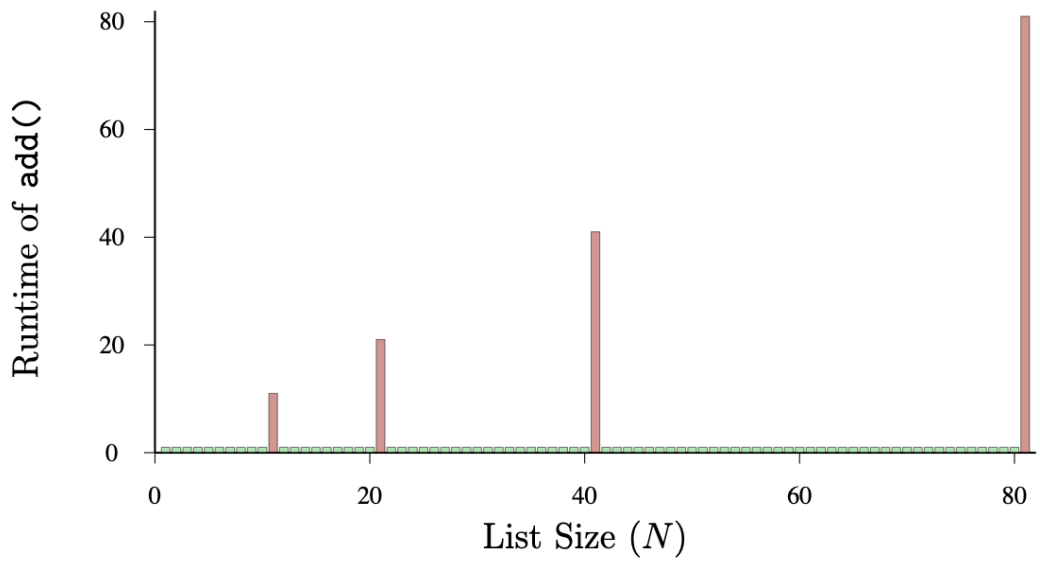
Time Complexity of DynamicArrayList

DynamicArrayList
+ insert(int index, T elem): void
+ remove(int index): T
+ size(): int
+ get(int index): T
+ set(int index, T elem): void
+ contains(T elem): boolean
+ indexOf(T elem): int
+ delete(T elem): void
+ add(T elem): void

Amortized Time Complexity

Let's think a bit more about runtime of add()

- Usually, just write to one array cell, update size
  O(1) operation
- Infrequently, resize and copy, O(N) operation



Amortized Time Complexity