CS151 Intro to Data Structures

Arrays, Generics

Announcements

- Lab02 today - due Friday Feb 7th
- HW0 and Lab1 due Friday Jan 31 (this friday!)

Outline

- Java Review
- ExpandableArray
- Generics

Java Review?

- if/else statements
- Loops and nested loops
- Static methods
- Arrays and 2D arrays
- Inheritance
- Compiling and running java code
- ....

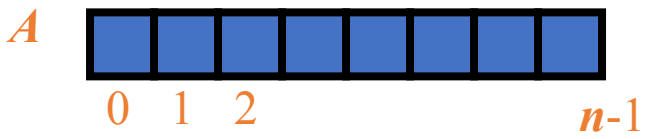
ARRAYS

Arrays

- Homogeneous types
- Contiguous memory
- Fast access
- Fixed size

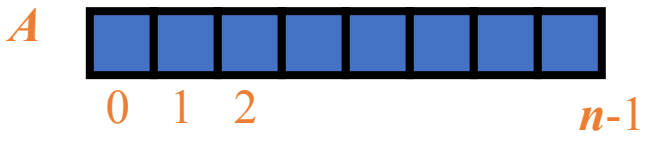
Let’s design an array that can change size!

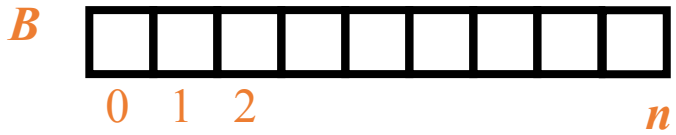
Imagine we have n items in our array



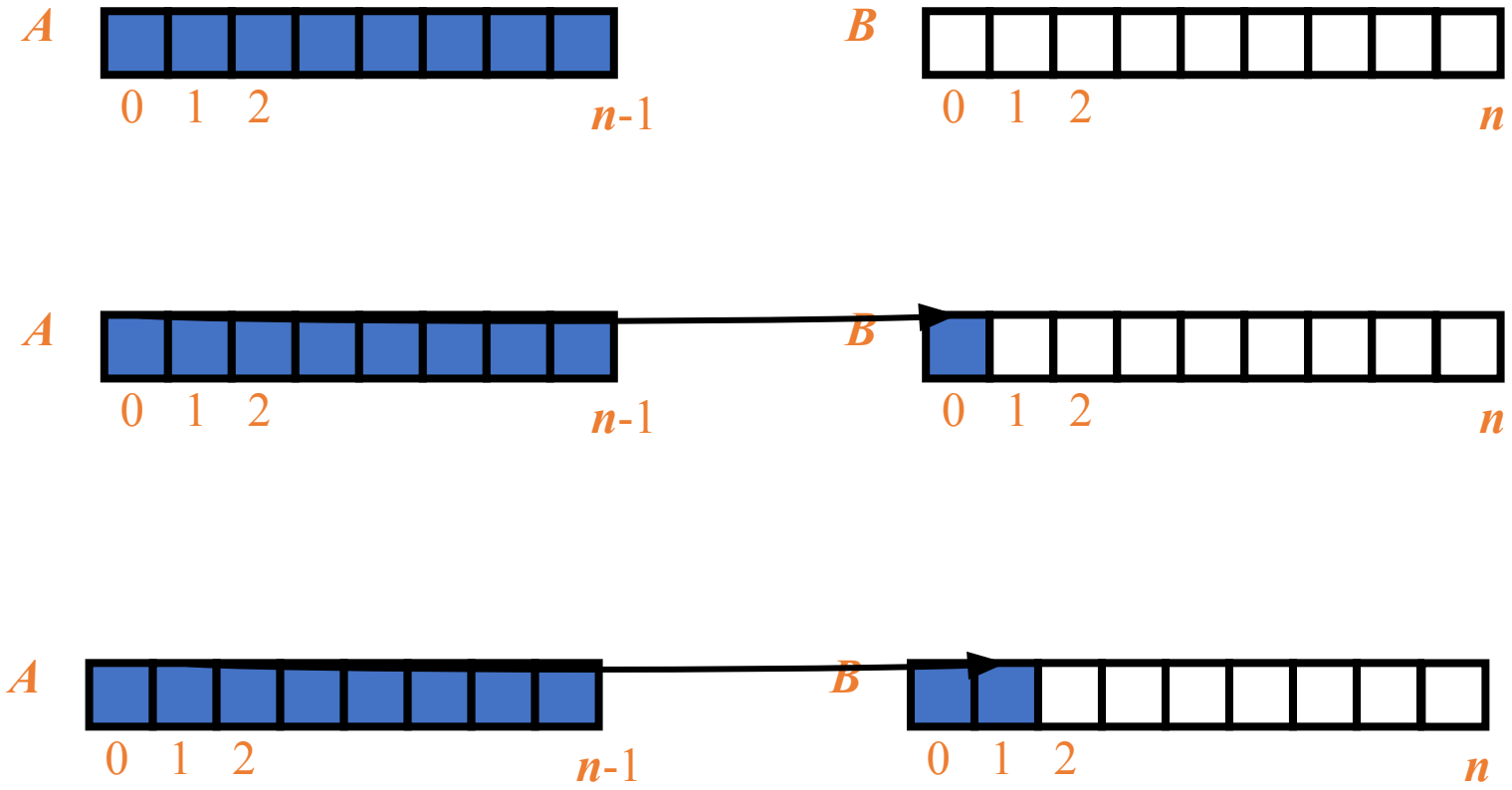
Say we want to add another item, are we stuck?

- No, make a new array and copy all the items over

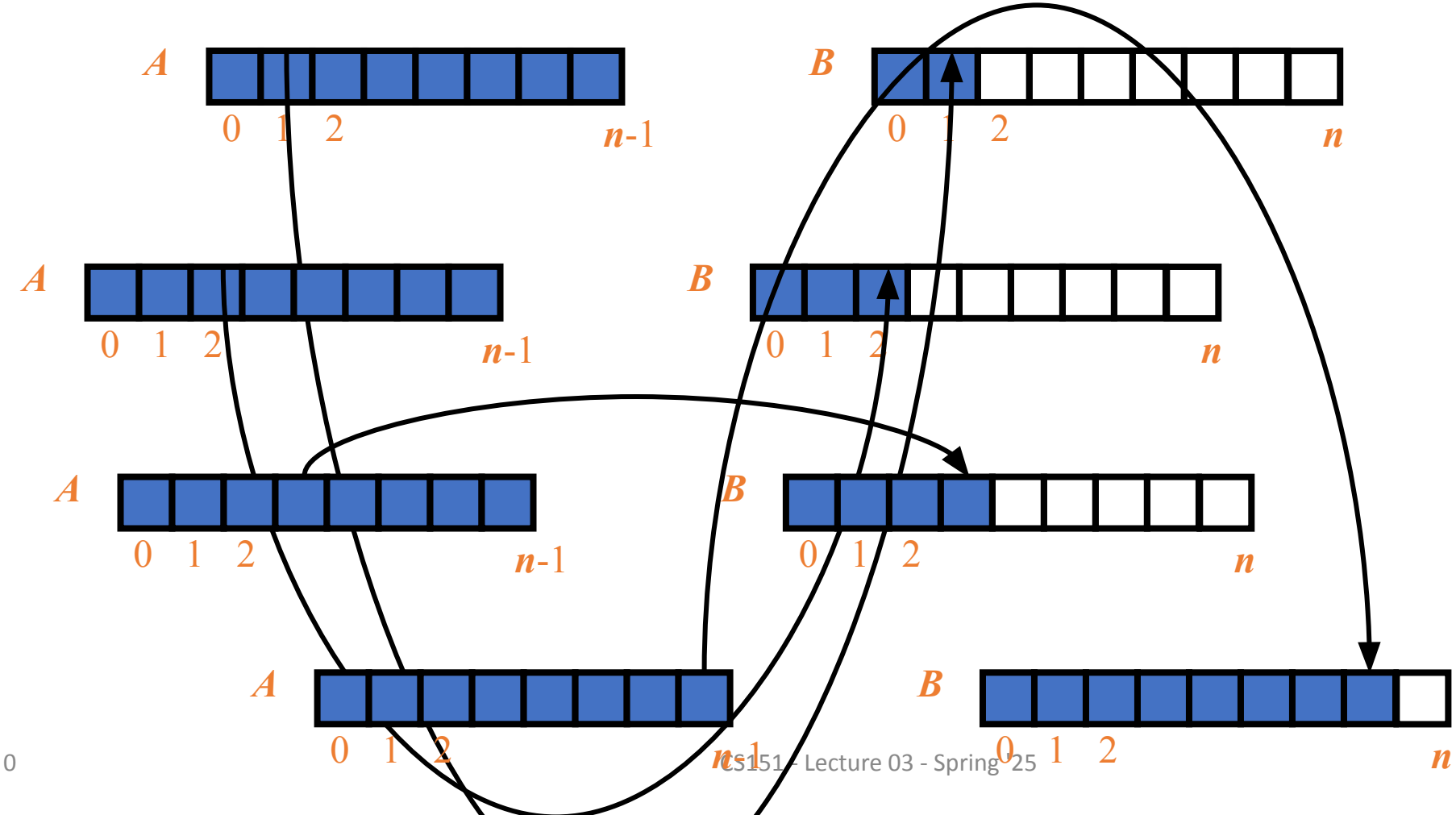




Array – Copying items over



Array – Copying items over



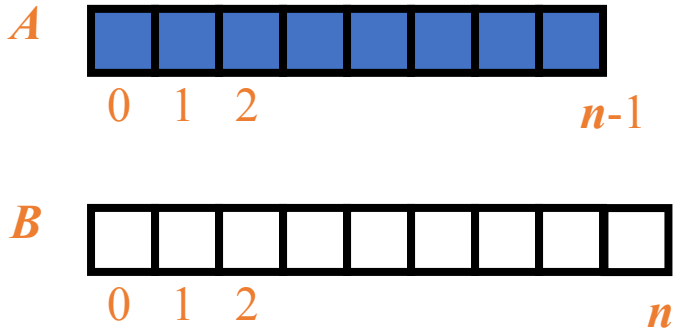
Array Copying

Computational complexity?

O(n)

How big should the new array be?

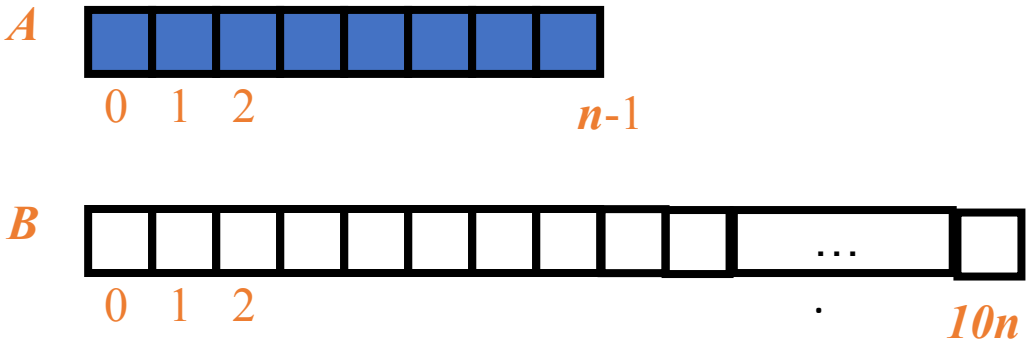
Just one more slot?



Pro: only use much space needed

Con: can lead to lots of copying over

10x the amount of slots?

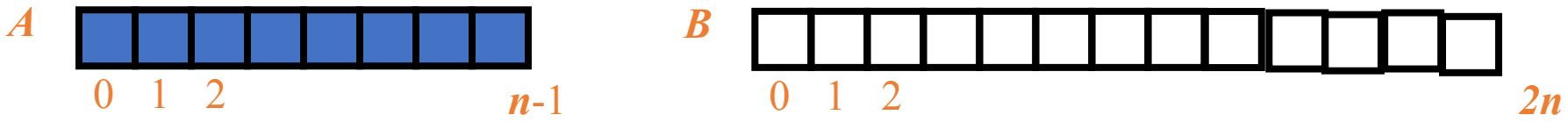


Pro: don’t need to copy lots of times

Con: lots of unused space

How big should the new array be?

- 2 times the length of the full array



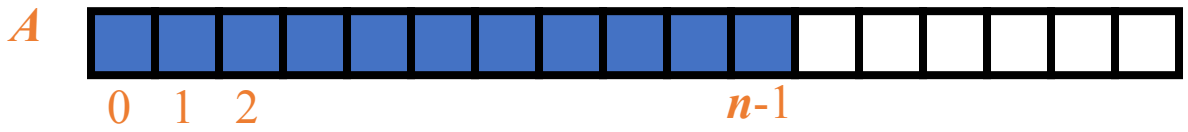
- Compromise between creating too much unnecessary space and having to expand the array too many times

- Runtime complexity?

Array Operations

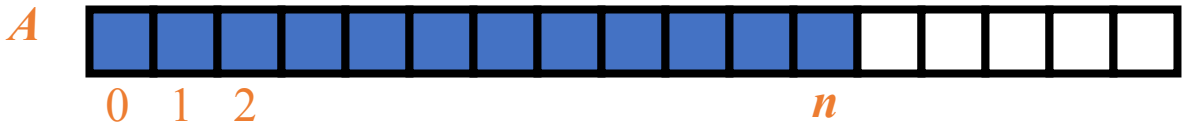
- Insertion
- Removal

Insertion



Where would be the easiest place to insert a new item?

The first open spot?

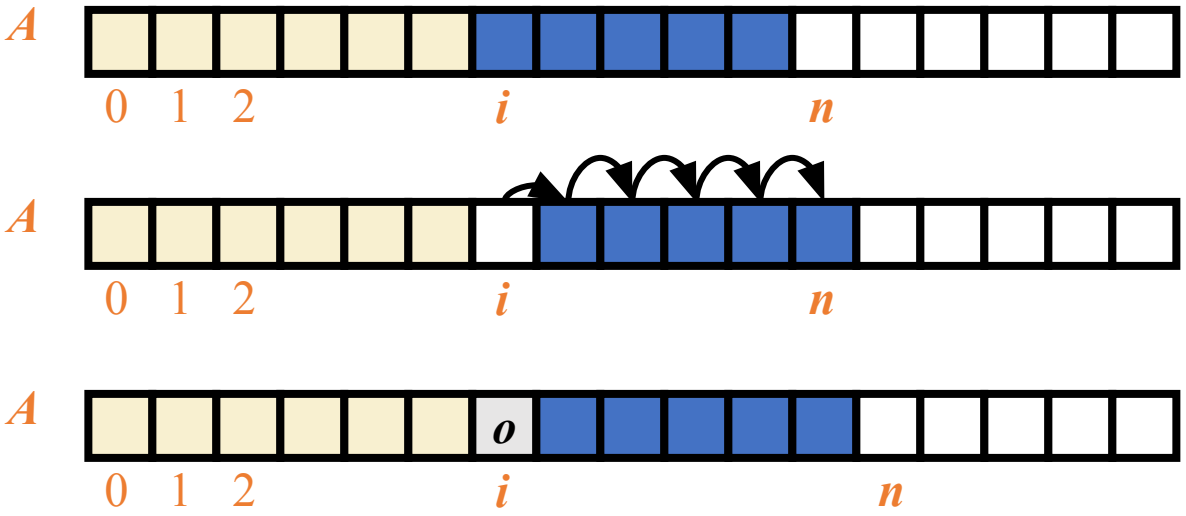


beginning of the array?

If we are going to search for that item a bunch

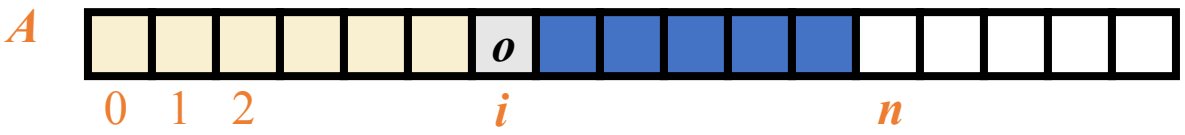
Insertion

- In an operation `insert(i, o)`, we make room for the new element `o` by shifting forward the elements `A[i], …, A[n - 1]`



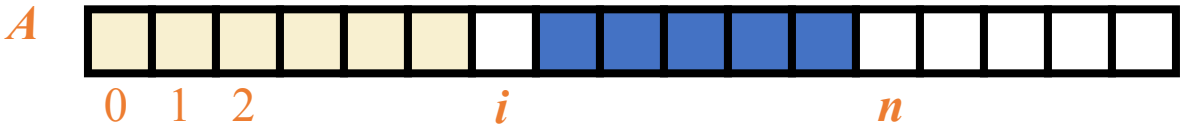
Removal

Say we want to remove the item at index `i`?

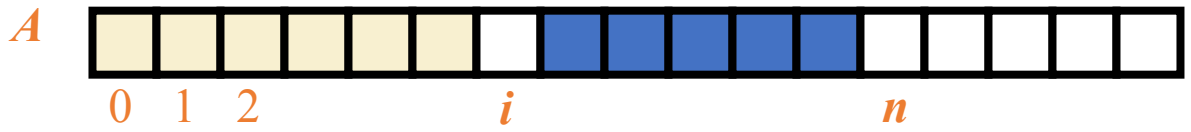


What's the simplest approach?

Just remove it, leaving an empty index



What is wrong with this setup?



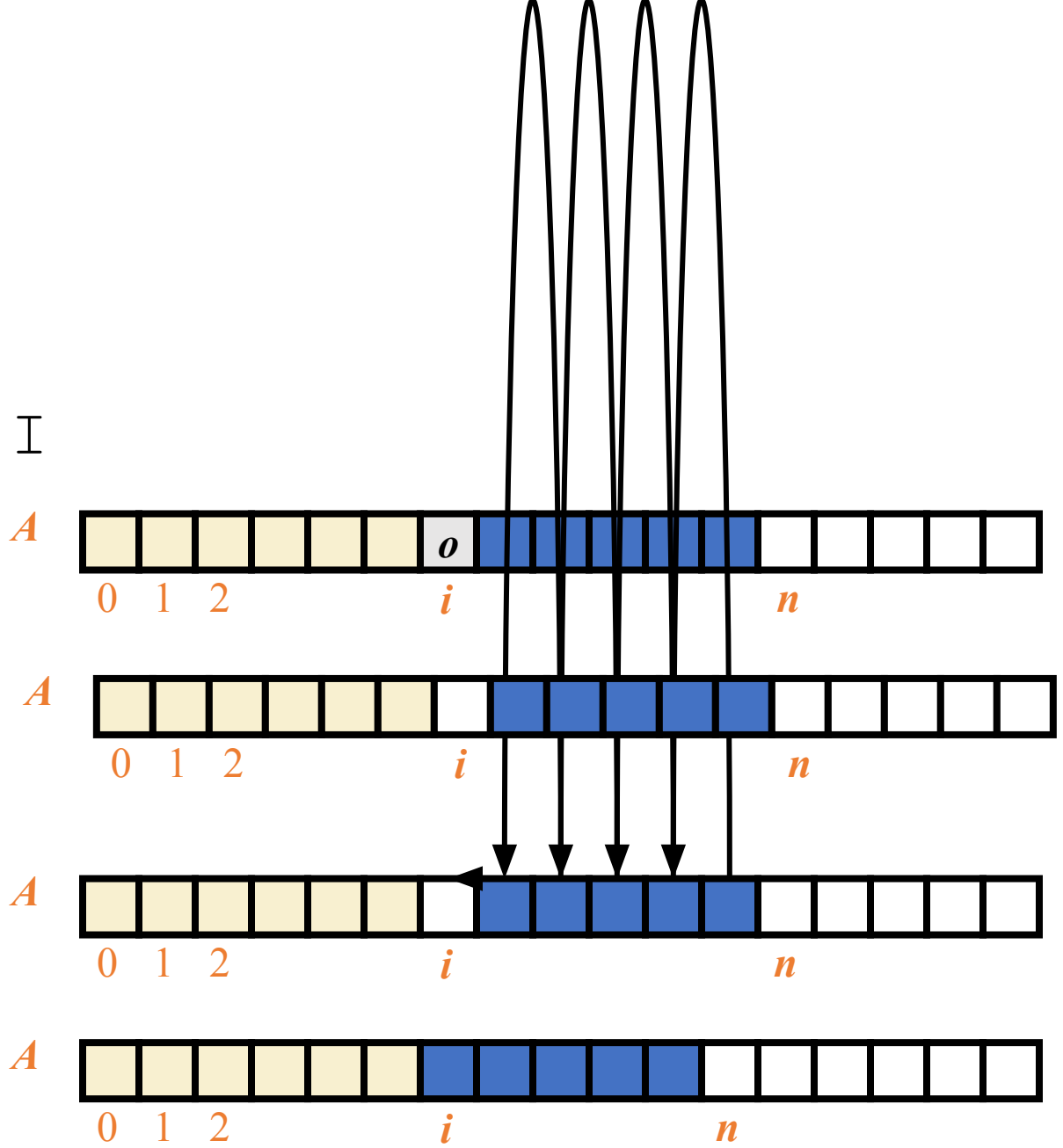
Why is having an empty slot in the middle of the array not ideal? What issues might arise?

- Makes inserting complicated
  - Where would we put a new item? At the end, or fill the spot?
- Makes looping through the array complicated
  - Need to check for null spots

Removing

In an operation `remove(i)`, we

- remove the element at location `I`
- then fill the hole by shifting backwards elements `A[i+1], …, A[n-1]`



Array Review

We designed an array that can change size

Insertion:

- Expand array 2x each time it’s full
- copy all elements over
- Complexity?

Deletion:

- Fill the hole by shifting everything backwards
- Complexity?

ExpandableArray

We designed an expandable array

Now we will Implement it

In this course, we will use simple Java data structures as an underlying tool to build off.

Let’s start coding it! :)

Questions?

ExpandableArray

What did we just do? Create an ExpandableArray for int types

```
private int[] data; //underlying array
```

- What if we want an ExpandableArray for doubles, Strings, Students, etc.
- We could create an ExpandableArray for each type... but now this violates our goal of **reusability**
- This brings us to **Generics**

Generics

Generics

- First, let's look at some code.
- A way to write classes or methods that can operate on a variety of data types without being locked into specific types at the time of definition
- Write definitions with type parameters

```
public <T> void print(T x) {
    System.out.println(x);
}
```

Generic Classes

- We just implemented a generic print method

- Let’s see how to make our ExpandableArray generic
  - **Code!**

Generics Arrays

Can not create arrays of parameterized types!

```
private T[] array = new T[10];
```
is not valid

- Casting to the rescue!
  - `T[] array = (T[]) new Object[10];`

Other Generic Restrictions

Can not declare static instance variables of a parameterized type

```
private static T MAX_SIZE;

//compiler error: non-static type variable T
cannot be referenced from a static context
```

Generic Static Methods

These are allowed!

```
//Static class method
public static <T> void getMax(T t){
    System.out.println(t);
}
```

```
//Class instance Method:
public T get(int index) {
     return this.data[index];
}
```

Summary

- Java arrays are *fixed-size,* contiguous, and sequential
- We started building our own ExpandableArray data structure
  - You will finish this in Lab / HW
- We made our ExpandableArray generic
  - What does this mean?