CS241, Jan 23, 11:00 AM

ADT = abstract data type

interface vs implementation

WHAT something does — HOW something is done

→ 2 components

- → Short description of what the data type represents (abstractly)
- → List of operations that the data type is capable of. These operations don't give low-level details about how ~~they~~ work. They just specify **WHAT** operations the data type can do.

Ex: Cars

| interface | | implementation | |
|---|---|---|---|
| - steering wheel | → | ? cables? | different for gas vs elec cars |
| - gas pedal | → | engine? | |
| - brake ~~pad~~ pedal | → | activate brakes | |

---

ADT = interface

In order to actually write a program, ADTs must be paired w/ an implementation.

"Data Structure" → the implementation part that is paired w/ the ADT
→ implementation + interface

# LIST ADT

→ description: A list consists of a collection of positions, each of which contains a single element of the list. Each position has a unique index, which is an integer in the range from 0...n-1, where n = # of elements in the list.

This description says nothing about how the list is stored in memory.

→ Operations

- → get the length/size of the list
- → add things into our list
- → remove things from our list
- → retrieve an item of the list at a specific index (GET)
- → sort the list
- → modify/change an element @ a given index (SET)
- → merge 2 lists together
- → create a new (blank) list

---

## How to implement this RList interface?

What is a data structure we could use to implement RList?

→ Java arrays

Big diff b/w java arrays vs ArrayLists

| | |
|---|---|
| int array[10]; | ArrayList<Integer> list = new ArrayList<>(); |
| FIXED SIZE | GROW + SHRINK |

NOTE: Programming langs need to know for each function how much memory the function will need (total size of all vars)

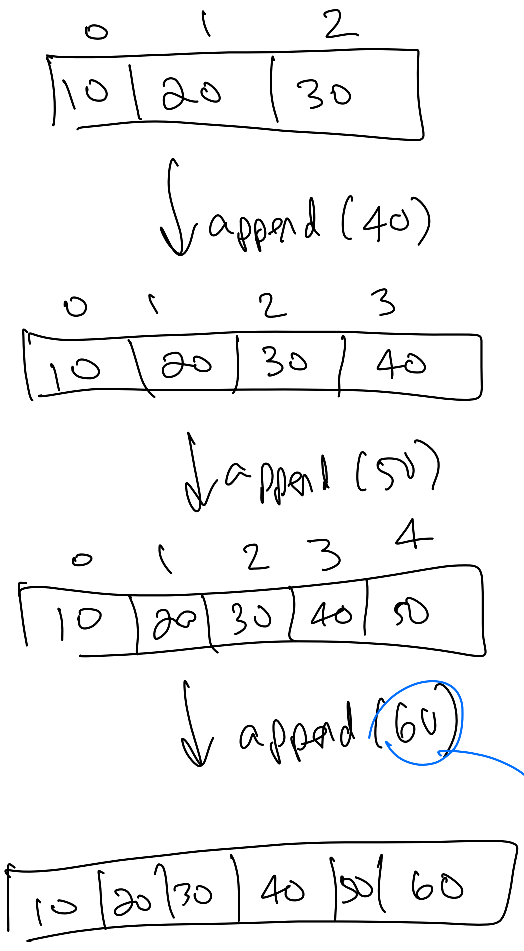
Java

int x; → 4 bytes

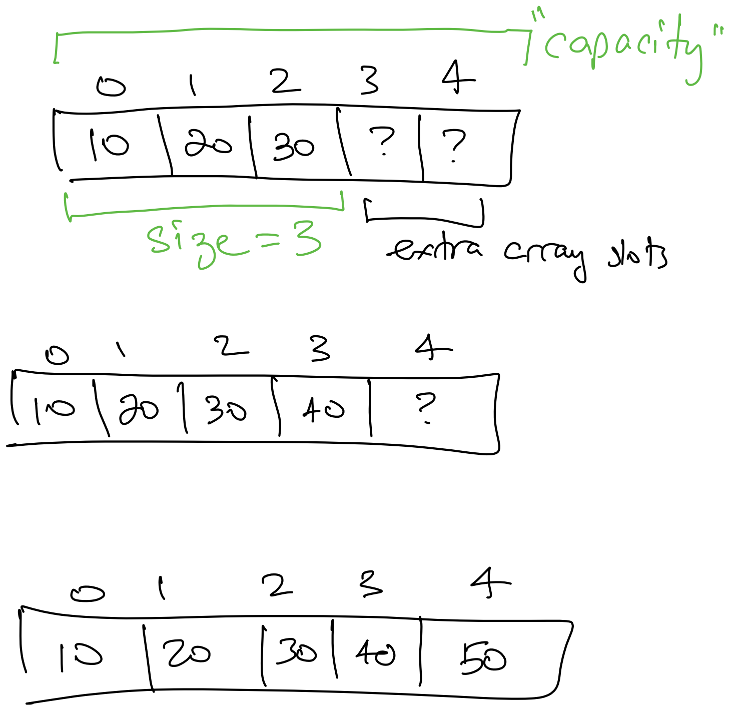
long y; → 8 bytes

int array[10]; → 4 bytes × 10 = 40 bytes

User's perspective (interface)



Programmer's perspective (implementation)

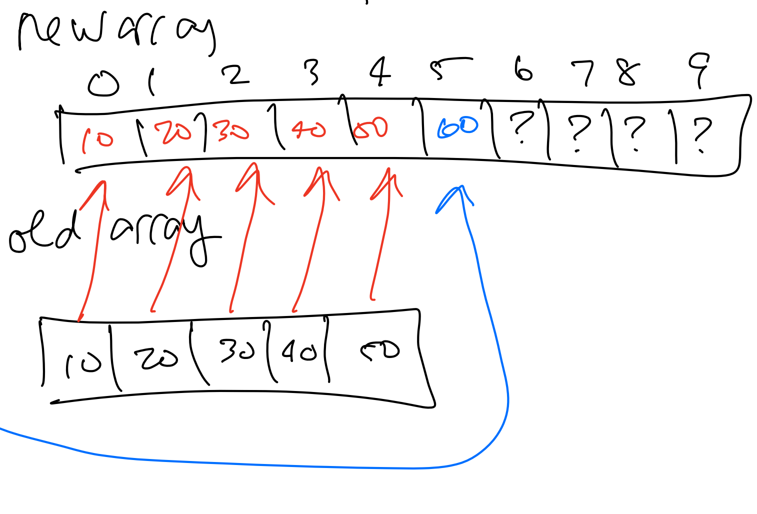


out of space ☹

"expand" the array

→ create a new array w/ extra spots

→ copy the old array into the new array



## Our Implementation : class RArraylist

int[] data; → hold the elements of the list
→ can't grow!

int size → size from the user's perspective

★ How do we detect when we are out of space?

size vs data.length

" == " " → out of space