ADTs – Abstract data type

Goals for 241

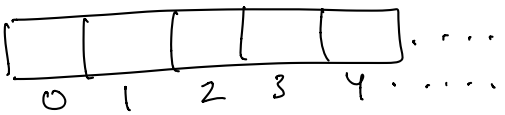
1. Introduce basic data structs & algs
2. Give practice in developing your own DS & Algs

→ Every ADT has 2 pieces
- A description of what the data type represents (abstract concept)
- List of the operations that the ADT can do.
  - ★ Tells us WHAT the ADT can do, not <u>how</u> it does it.

INTERFACE	IMPLEMENTATION
ADTs	Data structure

List ADT

Definition: A list consists of a collection of positions, each of which contains an integer. Each position is identified also by an integer starting at zero.



Operations

- Add to the list
  - Add @ beginning
  - Add @ end
- Iterate
- Remove
- Change elts
- Get next elt
- Sort
- get length
- determining if an item is in the list
- Remove duplicates

How could we implement this List?

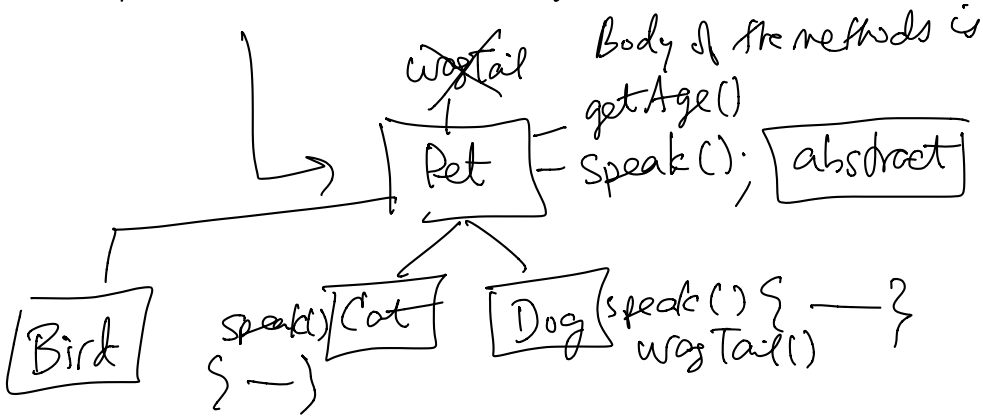
- Array ← obvious
- Linked list ← less obvious

Java Interfaces

Purpose — is to separate the interface of an ADT from its implementation

— "Regular" class — fields/methods

— Abstract class — you can have methods marked as ABSTRACT ↵




```
ArrayList<Pet> pets = new ArrayList<Pet>();
pets.add
```

```
pets.get(0).speak();
```

Pet

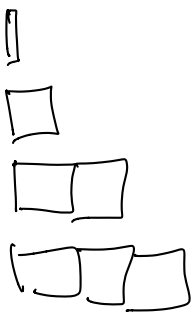
— Java interface — ALL methods are abstract

Java: class A extends class B ~~AND class C~~
class A extends class B implements C, D, E...
interfaces

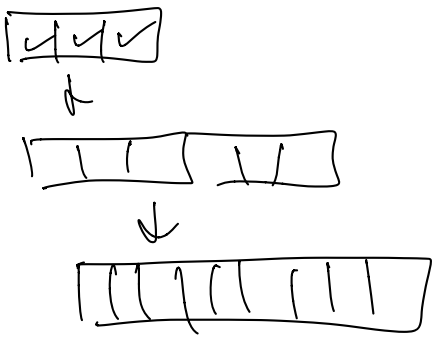
RList (interface)

RArrayList — grow on the fly

User



Creator



Interface

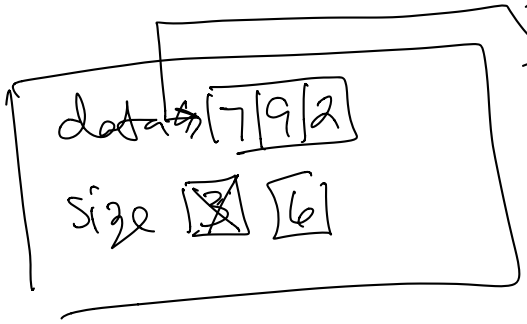
size()

Implementation

int data[]

int size — size from the user's perspective

(int capacity) — actual capacity of data[]



newdata