Coming up:

Due Fri, Jan 27: (end of the day)
Read (or finish reading) Section 2.(2, 4, & 5)
Take data structures quiz

Also:
Do "basic data structures" practice problems (suggested by Mon, Jan 30)
Do **"Implementing ADTs"** *project (suggested by Wed, Feb 1)*

Due Wed, Feb 1: (class time)
Read Section 3.1
Do Exercises 2.(22–24)
Take sorting quiz

This week and next week (Chapters 2 and 3):

- Abstract data types (Wednesday)
- Data Structures (**today** and Monday)
- Programming practices (Monday)
- Linear time sorting (next week Wednesday and Friday)

Today:

- Ex 1.11
- ADT review
- Data structure categories
- List vs array
- Abstractions
- Adapter pattern

```
def is_palindrome(str) :
    palindromic = True
    n = len(str)
    i = 0
    while palindromic and i < n // 2 :
        palindromic = str[i] == str[n-i-1]
        i += 1
    return palindromic
```

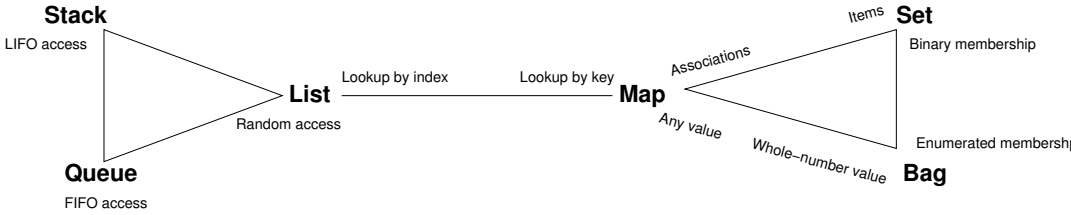
Invariant (Loop of `is_palindrome`)

1. $\forall\, j \in [0, i-1), \texttt{str}[j] = \texttt{str}[n-j-1]$
2. *palindromic iff* ($i = 0$ *or* $\texttt{str}[i-1] = \texttt{str}[n-i]$)
3. *i is the number of iterations completed*

	best case	worst case	expected case
binary search			
bounded linear search			
selection sort			
merge sort			
quick sort			

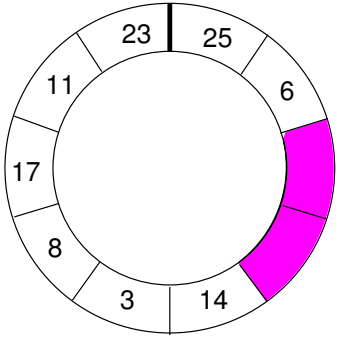
The “canonical ADTs”:

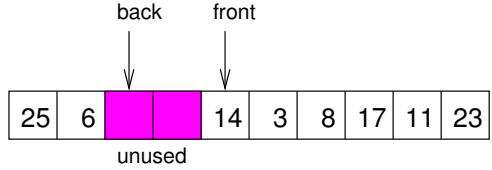
List.	Linear collection with sequential and random access.
Stack.	Linear collection with LIFO access.
Queue.	Linear collection with FIFO access.
Set.	Unordered collection with binary membership.
Bag.	Unordered collection with enumerated membership.
Map.	Unordered collection of associations between keys and values.

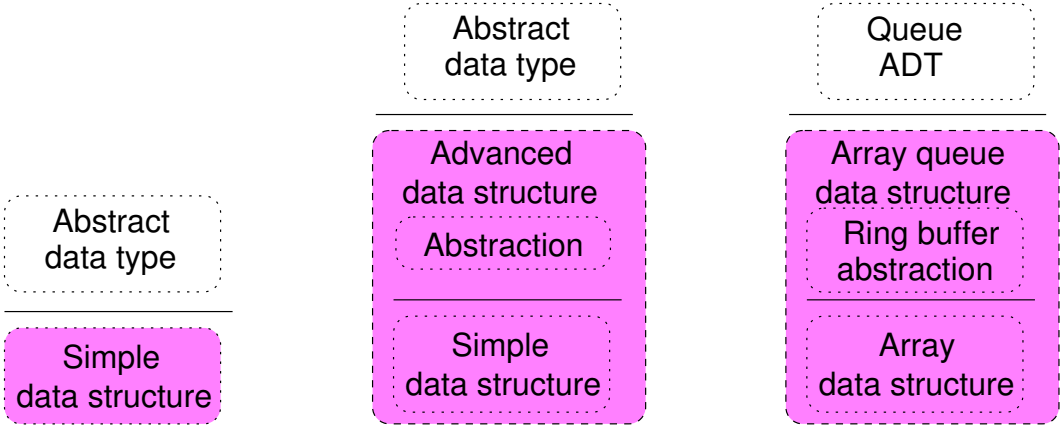

Stack
LIFO access
Queue
FIFO access
List
Random access
Lookup by index
Lookup by key
Map
Associations
Any value
Whole−number value
Items
Set
Binary membership
Enumerated membershp
Bag

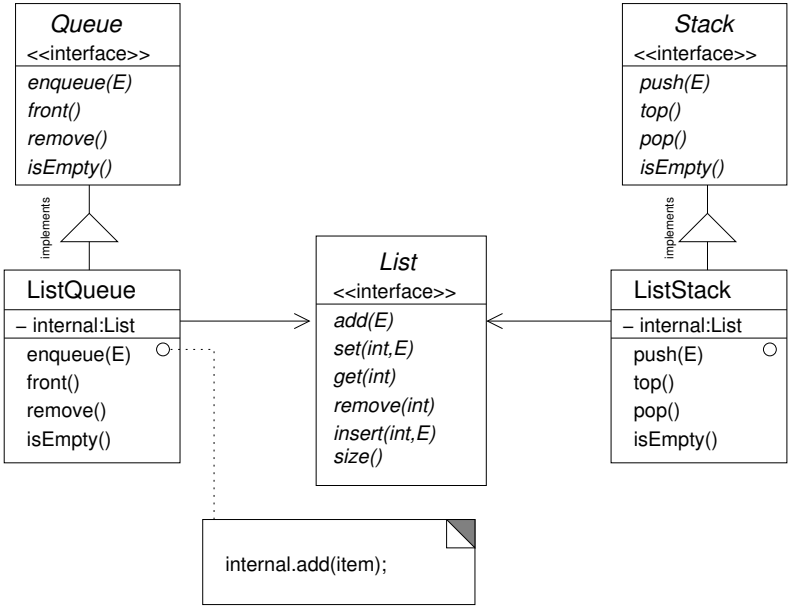
The four basic ways to implement an ADT:

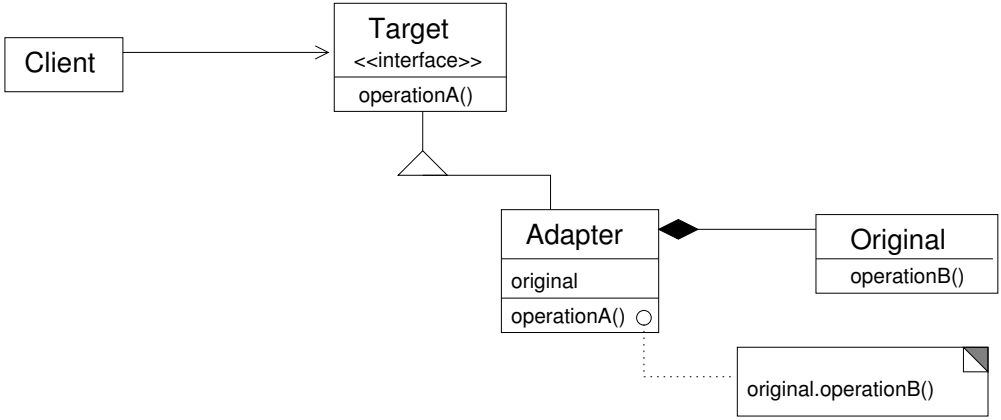
- Use an array
- Use a linked structure
- Use an "advanced" data structure, varying and/or hybridizing linked structures and arrays
- Adapt an existing implementation of another ADT.

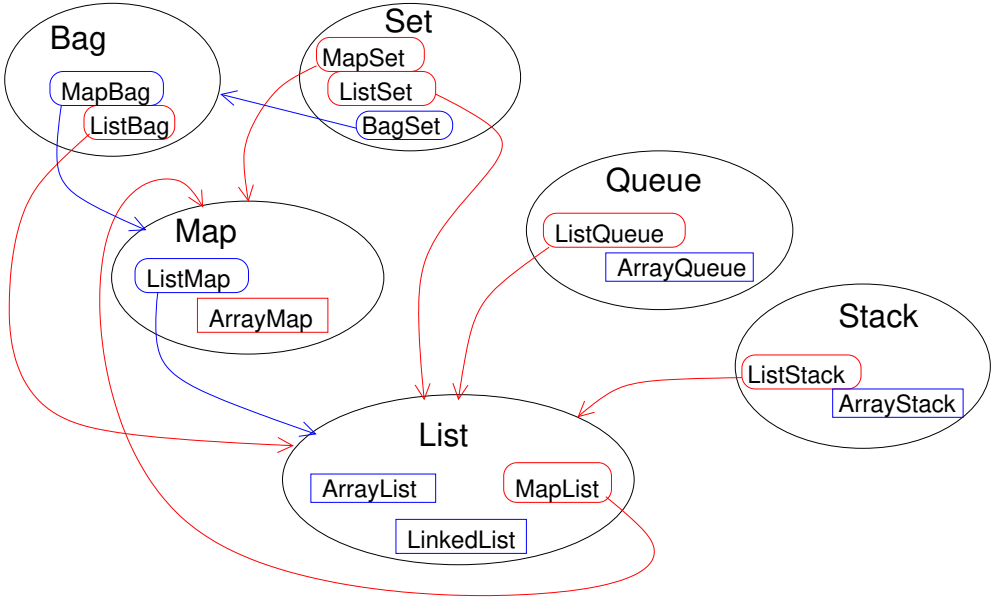

23
25
11
6
17
8
3
14


back
front
25
6
14
3
8
17
11
23
unused


Abstract
data type
Simple
data structure
Abstract
data type
Advanced
data structure
Abstraction
Simple
data structure
Queue
ADT
Array queue
data structure
Ring buffer
abstraction
Array
data structure


Queue
<<interface>>
enqueue(E)
front()
remove()
isEmpty()
implements
ListQueue
– internal:List
enqueue(E)
front()
remove()
isEmpty()
List
<<interface>>
add(E)
set(int,E)
get(int)
remove(int)
insert(int,E)
size()
Stack
<<interface>>
push(E)
top()
pop()
isEmpty()
implements
ListStack
– internal:List
push(E)
top()
pop()
isEmpty()
internal.add(item);


Client
Target
<<interface>>
operationA()
Adapter
original
operationA()
Original
operationB()
original.operationB()


Bag
MapBag
ListBag
Set
MapSet
ListSet
BagSet
Map
ListMap
ArrayMap
Queue
ListQueue
ArrayQueue
Stack
ListStack
ArrayStack
List
ArrayList
MapList
LinkedList

Coming up:

Due Fri, Jan 27: (end of the day)
Read (or finish reading) Section 2.(2, 4, & 5)
Take data structures quiz

Also:
Do "basic data structures" practice problems (suggested by Mon, Jan 30)
Do **"Implementing ADTs"** *project (suggested by Wed, Feb 1)*

Due Wed, Feb 1: (class time)
Read Section 3.1
Do Exercises 2.(22–24)
Take sorting quiz