Maps + Sets

└ Almost every program you write in the real world has either a List or a Map or both.

MAP – Dictionary, Symbol Table, Associative Array, Hashtable

- is a collection of associations between pairs of data.
- Maps are used when you want to save a piece of information (a value) & later retrieve that value based on another piece of info called a key

Sometimes called a dictionary b/c a map is used to look up values based on their keys.

	KEY		VALUE
Dictionary	word	→	definition
Bank:	acct#	→	balance
Airline:	flight#	→	start city, end city
Event:	Conf #	→	date, time, loc'n...
Address Book:	name	→	tel #, email, address
Symbol table:	variable	→	~~value~~ memory loc'n
└→	functions	→	code

- Purpose of a map is to organize these key-value pairs in a way to make them fast to search.

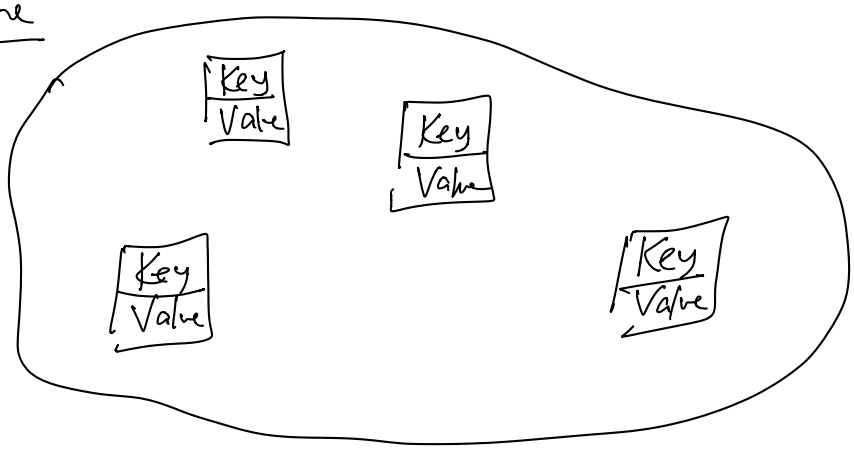
- Usually for arrays/lists, the user controls the order of items.
- Usually for maps " programmer " " key-value pairs.

Operations?

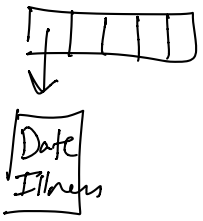
Put(key, value) — stores this key-value pair in the map.
↳ add, insert

Get(key) — return the value associated w/ ~~that~~ key.
↳ lookup, find, search

Picture



Dog → Vet Record



Regular ARRAYS: lookup an item based on index

Map (assoc array): lookup a value based on a key

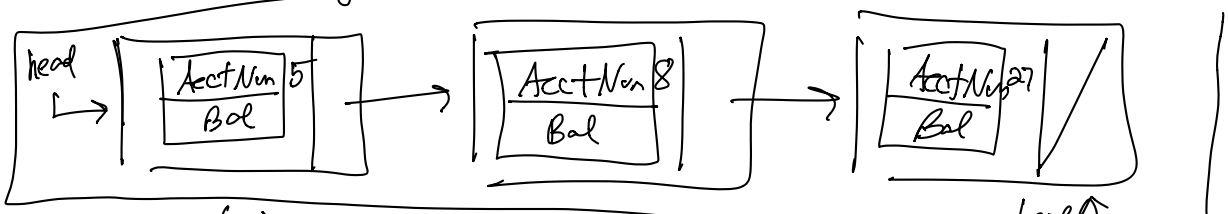
Implementation

Bank : Key = int (acct#)
Value = doubles (balance)

```
class KVPair {
    int acctNum;
    double balance;
}
```

public toString()
↓
String

① Linked List of KVPairs → numerical order by acct



Put? O(n)

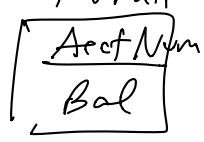
Get? O(n)

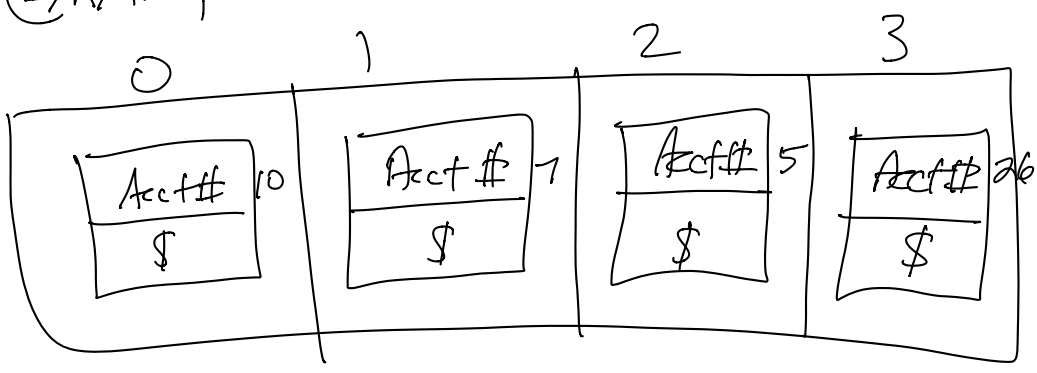
```
Node {
    KVpair data;
    Node next;
}

Node n = new Node()
n.data.acctNum = 10
n.data.balance = 1,000
```

★ ② RArrayList of KVPairs — (not sorted)

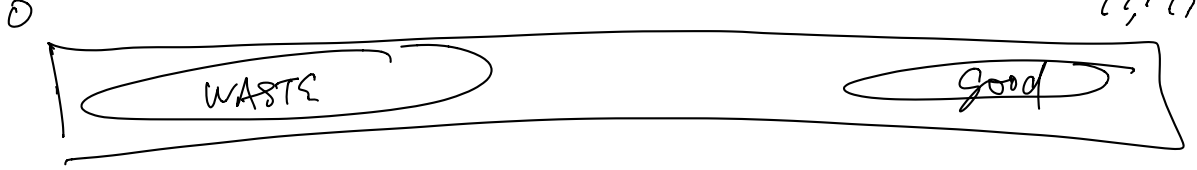
KVPair



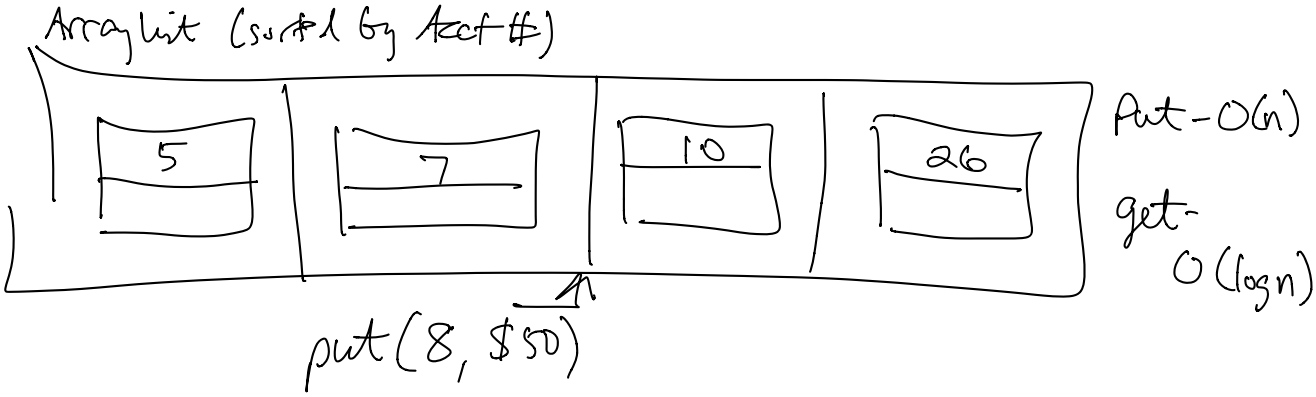


Put – O(n) (expand)

Get – O(n)



(3)



Pseudocode

```
ArrayList<KVPair> list;

void put(int acctNum, double balance)
{
    KVPair newpair = new KVPair();
    newpair.acctNum = acctNum;
    newpair.balance = balance
    list.add(newPair)      // list.append(newPair)
}
```

O(n)

↑ expand