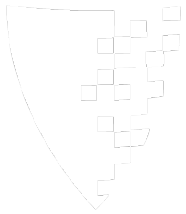
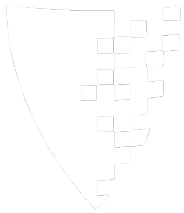
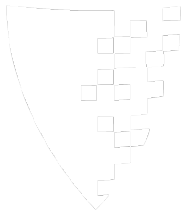


Sets and Maps



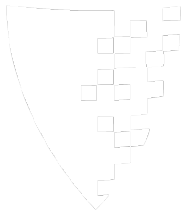
Announcements

- APT set 3 – due Feb 12
- Exam – Feb 15
- Homework submissions
  - Check them!
  - Link on assignment page
  - Incorrect submissions will no longer be accepted



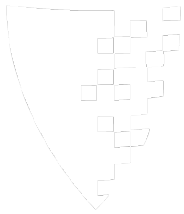
Last time

	ArrayList	LinkedList
add/remove (element at *k*)	O(N)	O(k)
add/remove (element at end)	O(1)*	O(1)
get (element at *k*)	O(1)	O(k)



Today

- Set
- Map
- Hashing
- Markov Models (The next assignment)

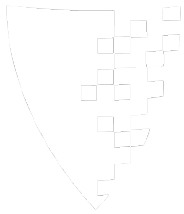


- Set – unordered collection of unique values

java.util

Interface Set<E>

Method Summary	
boolean	**add**(E e) Adds the specified element to this set if it is not alrea
boolean	**addAll**(Collection<? extends E> c) Adds all of the elements in the specified collection to
void	**clear**() Removes all of the elements from this set (optional o
boolean	**contains**(Object o) Returns `true` if this set contains the specified elemen
boolean	**containsAll**(Collection<?> c) Returns `true` if this set contains all of the elements o
boolean	**equals**(Object o) Compares the specified object with this set for equali
int	**hashCode**() Returns the hash code value for this set.



- Map – Collection of values mapped to keys
  - dictionary

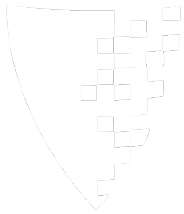
java.util

Interface Map<K,V>

Type Parameters:

K - the type of keys maintained by this map
V - the type of mapped values

Method Summary	
void	**clear**() Removes all of the mappings from this map (opti
boolean	**containsKey**(Object key) Returns true if this map contains a mapping for t
boolean	**containsValue**(Object value) Returns true if this map maps one or more keys t
Set<Map.Entry<K,V>>	**entrySet**() Returns a Set view of the mappings contained in
boolean	**equals**(Object o) Compares the specified object with this map for e
V	**get**(Object key) Returns the value to which the specified key is m
int	**hashCode**() Returns the hash code value for this map.



- Implementation of Set
  - We have seen two
    - HashSet
    - TreeSet

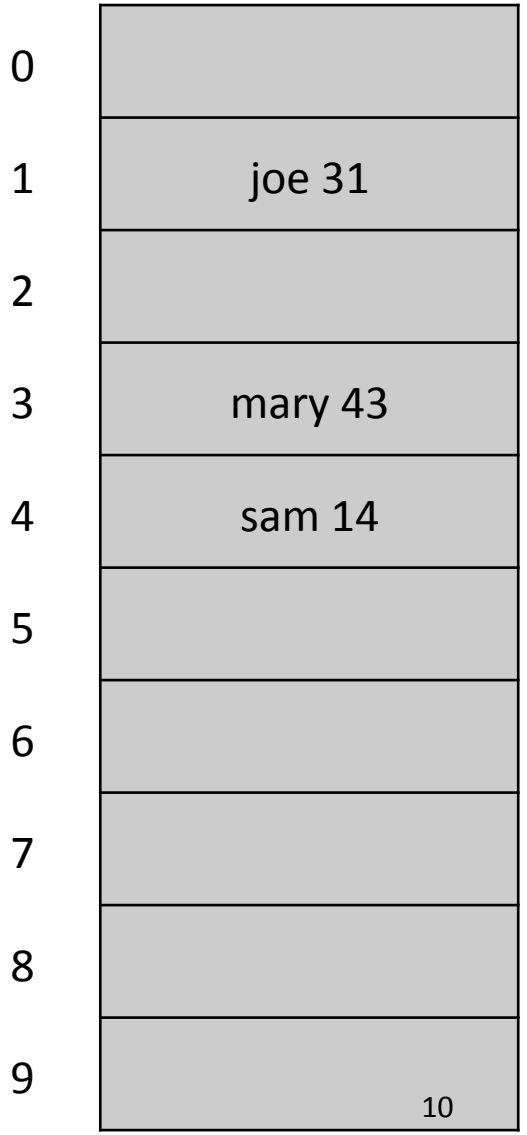
- Today
  - Hash
- After the exam
  - Tree

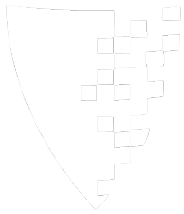
	HashSet/HashMap	TreeSet/TreeMap
Get element with key	O(1)*	O(logN)
Set element with key	O(1)*	O(logN)
Check if key exists	O(1)	O(logN)

Hashing

- HashTable
  - array of fixed size

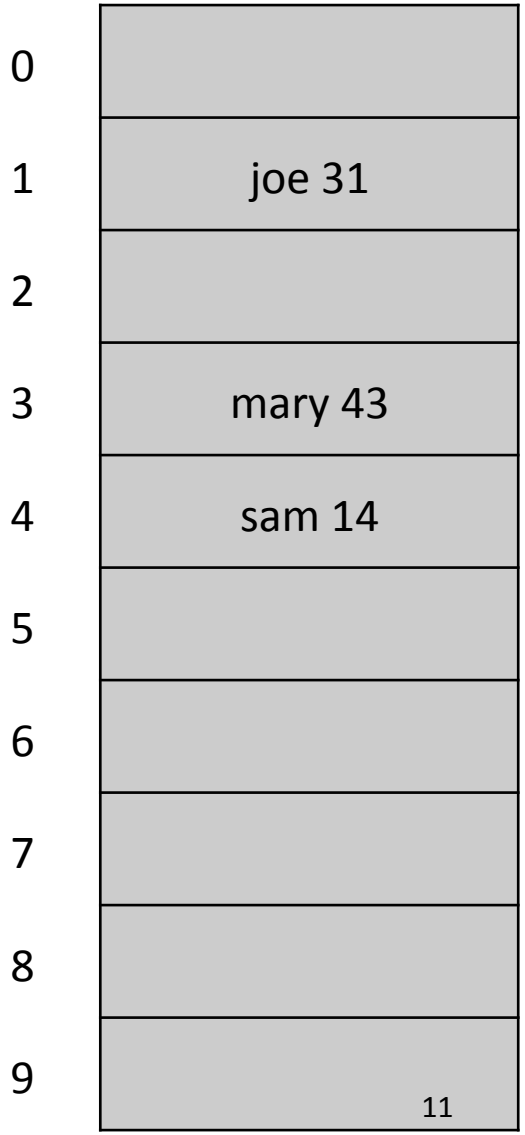
    - with a key to each location
    - each key is mapped to an index in the table





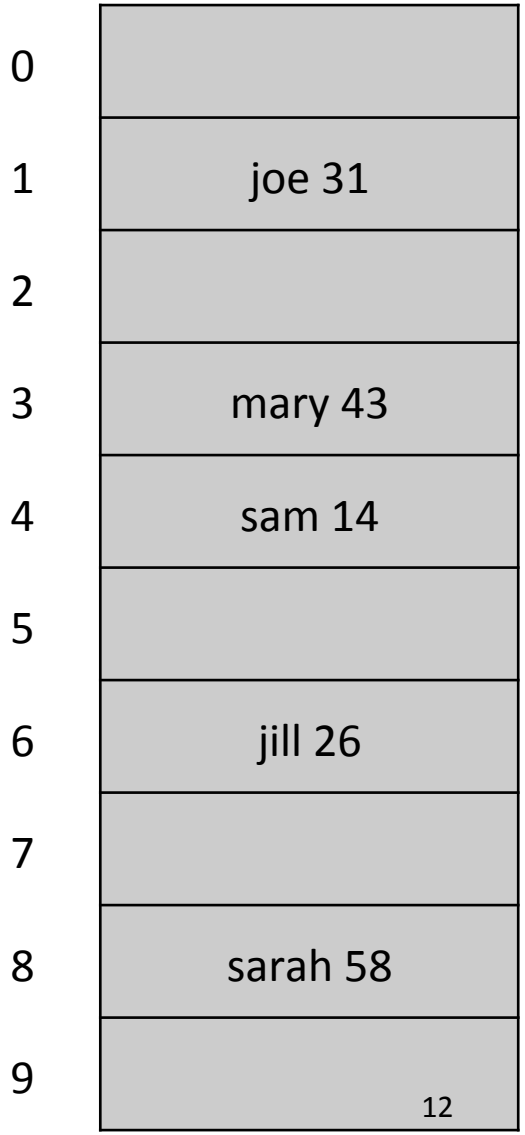
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



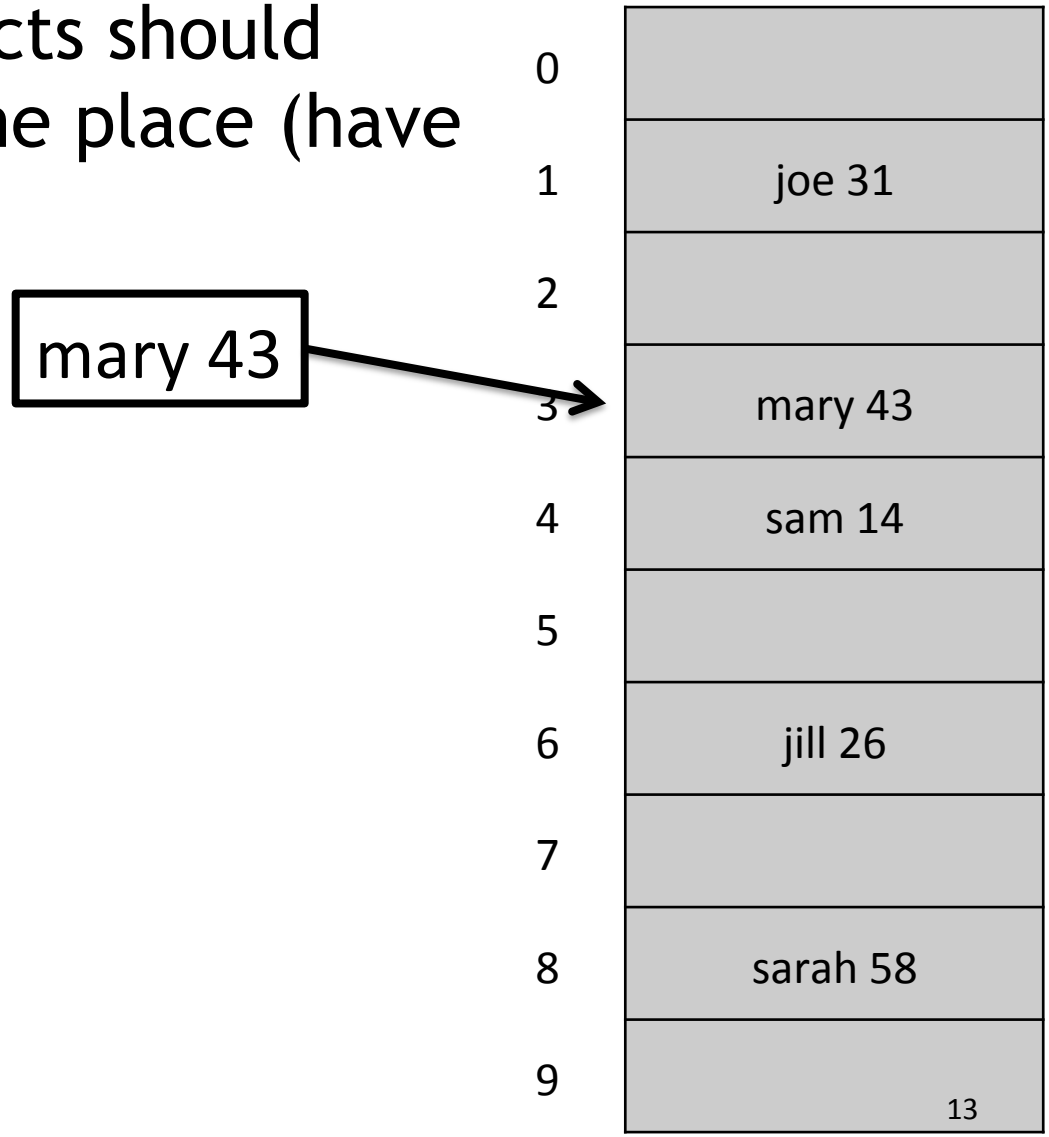
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



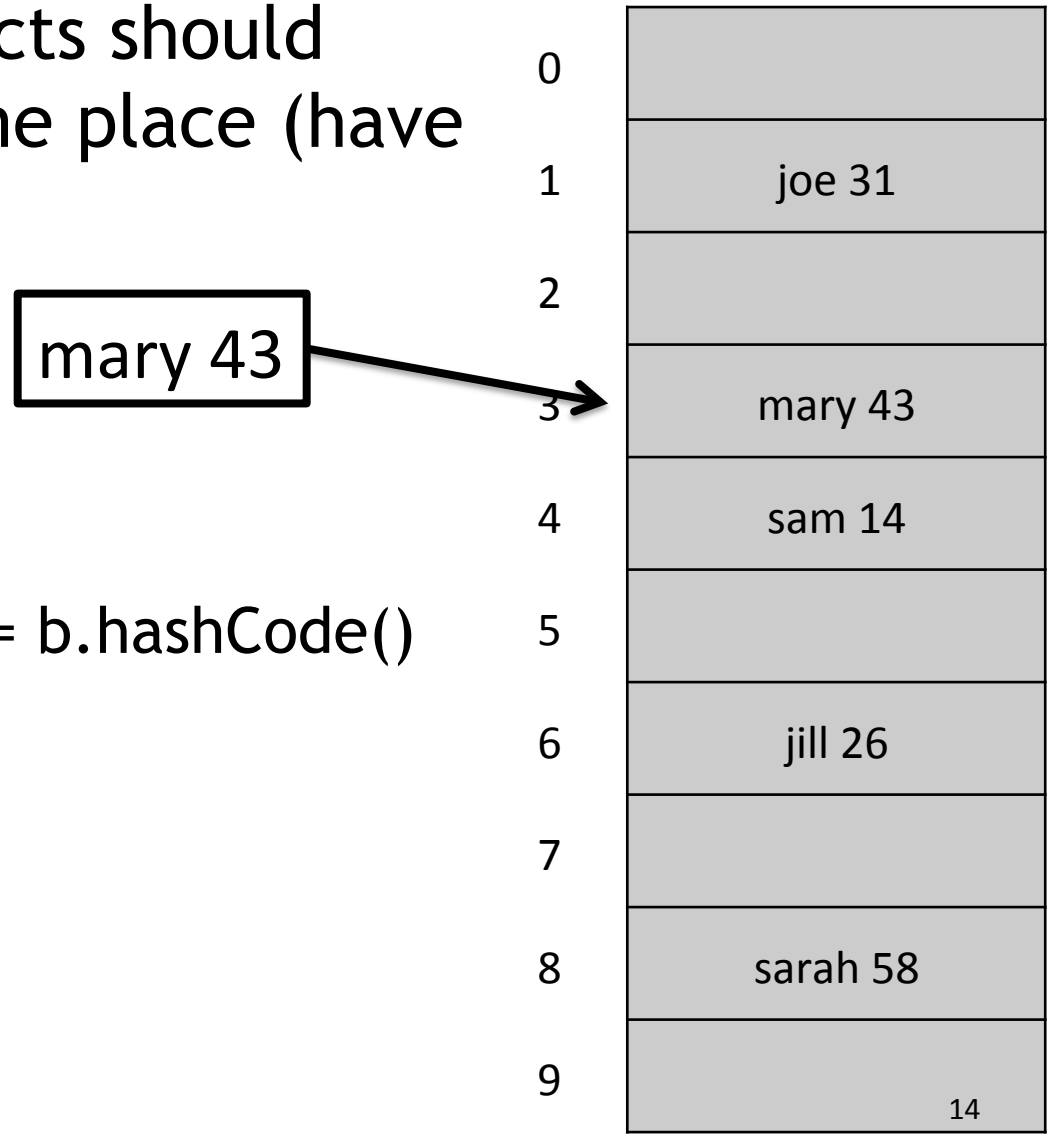
Hashing

- Two equal objects should hash to the same place (have the same key)



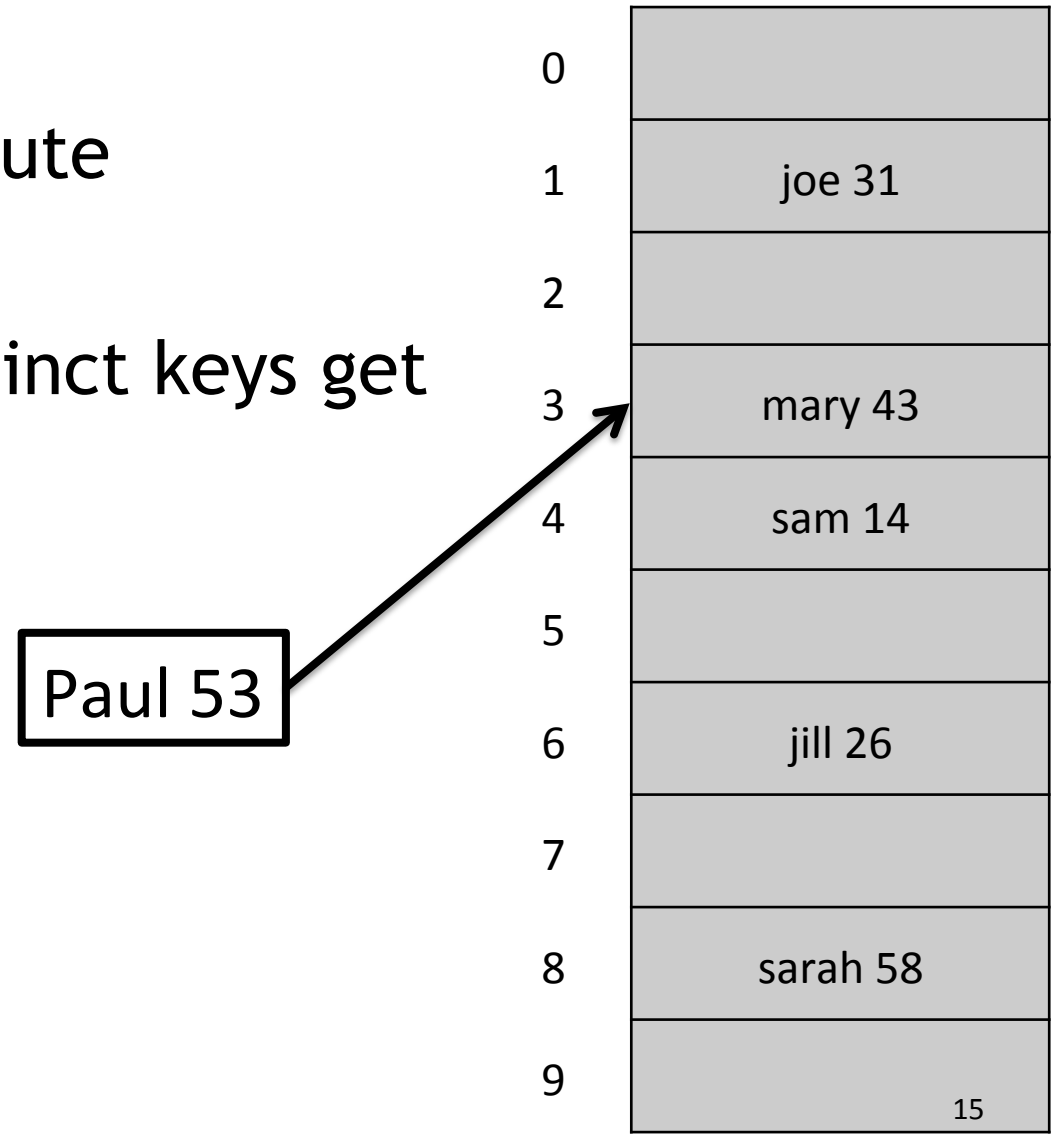
Hashing

- Two equal objects should hash to the same place (have the same key)
- if a.equals(b)
- then
  - a.hashCode() == b.hashCode()



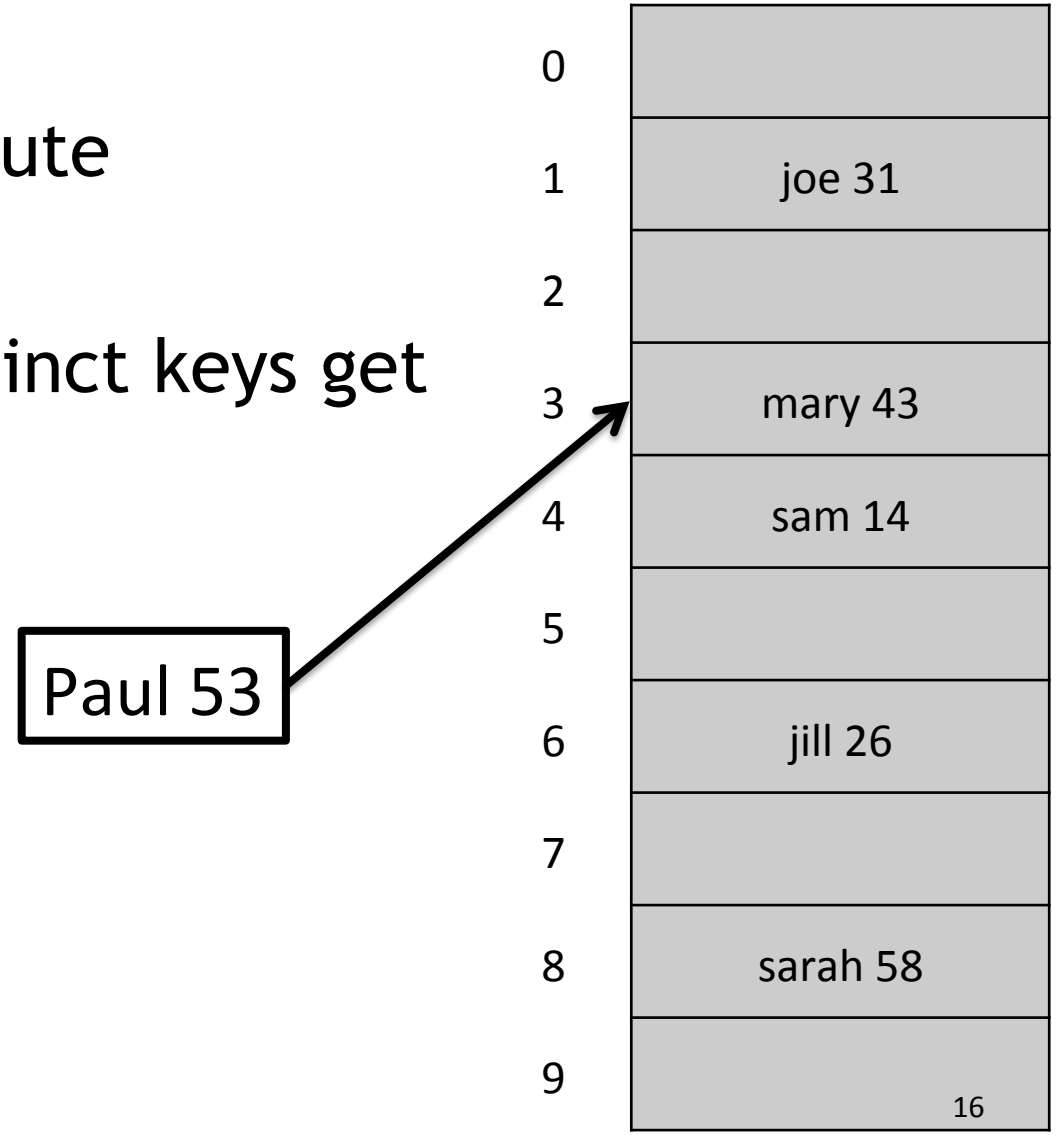
Hashing

- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



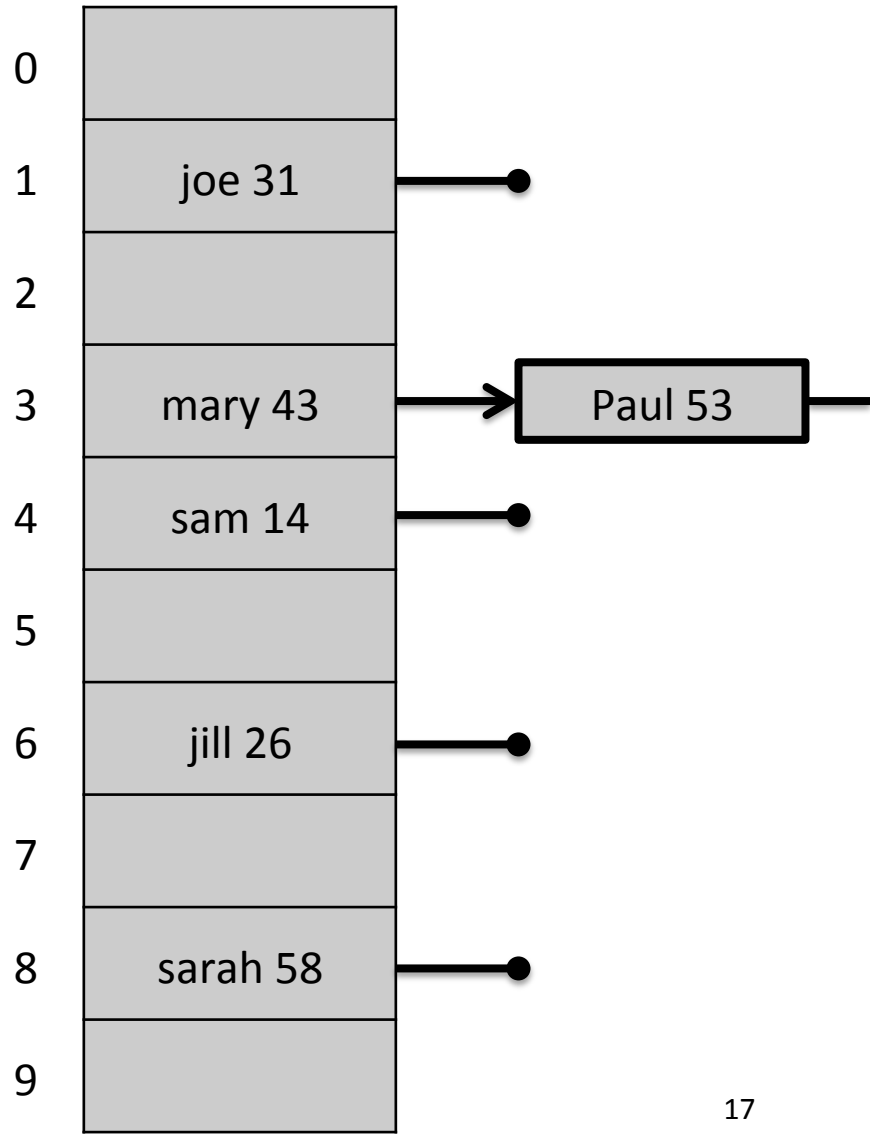
Hashing

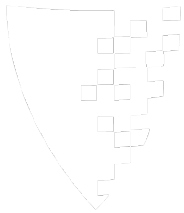
- Hash function
  - simple to compute
  - ensure two distinct keys get different cells



Hashing

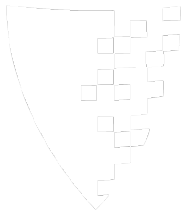
- Separate Chaining
  - make your table into linked lists!





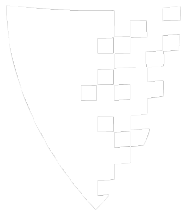
Hashing

- Code
  - Open your code from Jan 28
    - Practice with Hashing!
  - if a.equals(b)
  - then
    - a.hashCode() == b.hashCode()

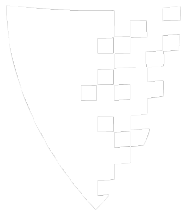


Hashing

- hashCode
  - every object should have a hash code
  - hashCode should not change unless value changes
  - if a.equals(b)
    - a.hashCode() == b.hashCode()



	HashSet/HashMap	TreeSet/TreeMap
Get element with key	O(1)*	O(logN)
Set element with key	O(1)*	O(logN)
Check if key exists	O(1)	O(logN)

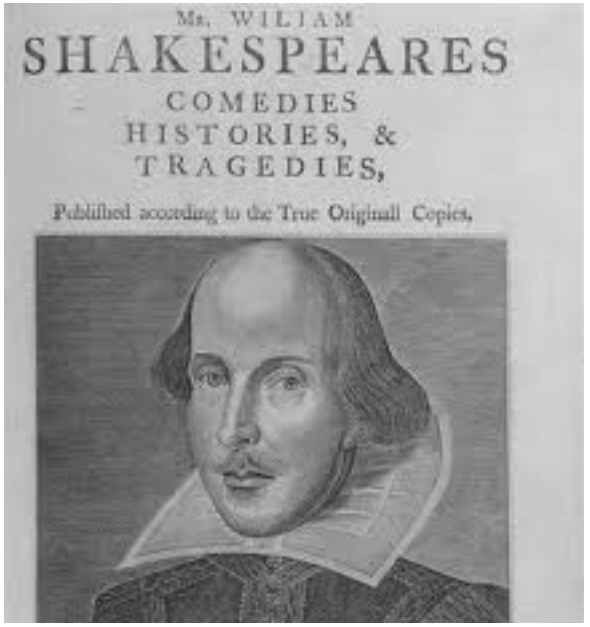


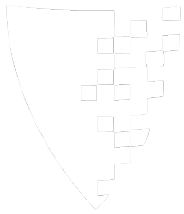
Today

- Set
- Map
- Hashing
- Markov Models (The next assignment)

Markov

- The infinite monkey theorem





Markov

- Training text
  - ex. Huckleberry Finn
- Build a map from text
  - ‘e’ is followed by ‘a’ 30% of the time
  - ‘a’ is followed by ‘t’ 20% of the time
- Generate random text

Markov

- “bbbabbabbbbbaba”

3-gram

bbb			

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		
bba	bab		

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		
bba	bab		
bab	abb		

Markov

- “bbbabbabbbbaba”

3-gram

bbb	bba		
bba	bab		
bab	abb		
abb	bba		

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		
bba	bab	bab	
bab	abb		
abb	bba		

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		
bba	bab	bab	
bab	abb	abb	
abb	bba		

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba		
bba	bab	bab	
bab	abb	abb	
abb	bba	bbb	

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba	bbb	
bba	bab	bab	
bab	abb	abb	
abb	bba	bbb	

Markov

- “bbbabbabbbbaba”

3-gram

bbb	bba	bbb	bba
bba	bab	bab	
bab	abb	abb	
abb	bba	bbb	

Markov

- “bbbabbabbbbaba”

3-gram

bbb	bba	bbb	bba
bba	bab	bab	bab
bab	abb	abb	
abb	bba	bbb	

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba	bbb	bba
bba	bab	bab	bab
bab	abb	abb	aba
abb	bba	bbb	

Markov

- “bbbabbabbbbbaba”

3-gram

bbb	bba	bbb	bba
bba	bab	bab	bab
bab	abb	abb	aba
abb	bba	bbb	
aba	bab		

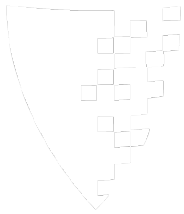
Markov

- “bbbabbabbbbbaba”

3-gram

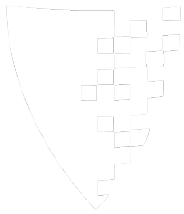
bbb	bba	bbb	bba
bba	bab	bab	bab
bab	abb	abb	aba
abb	bba	bbb	
aba	bab		

- “bbb” followed by “bba” 66% and “bbb” 33%
- “bba” always followed by “bab”



Markov

- Training text
  - ex. Huckleberry Finn
- Build a map from text
  - ‘e’ is followed by ‘a’ 30% of the time
  - ‘a’ is followed by ‘t’ 20% of the time
- Generate random text



Today

- Set
- Map
- Hashing
- Markov Models (The next assignment)
  - Due Feb 19
  - Start Early!!!!!