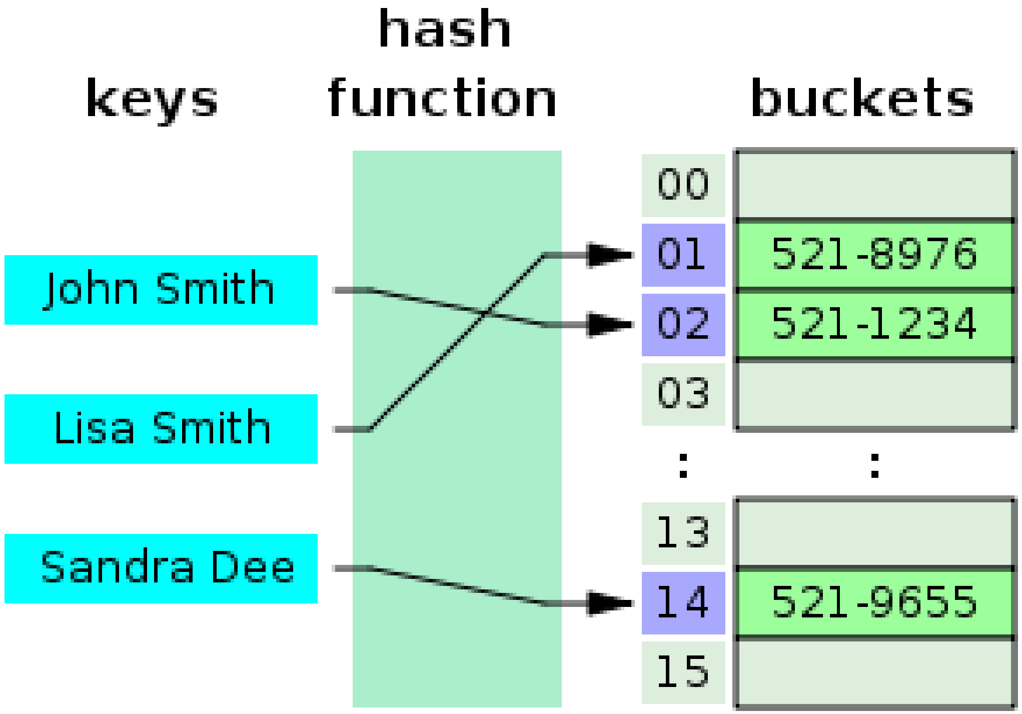
HashTables

Lab 16

HashTable

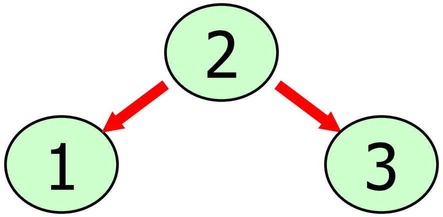
The hash function calculates an index from the data item's key and use this index to place the data into the array.



Binary Tree

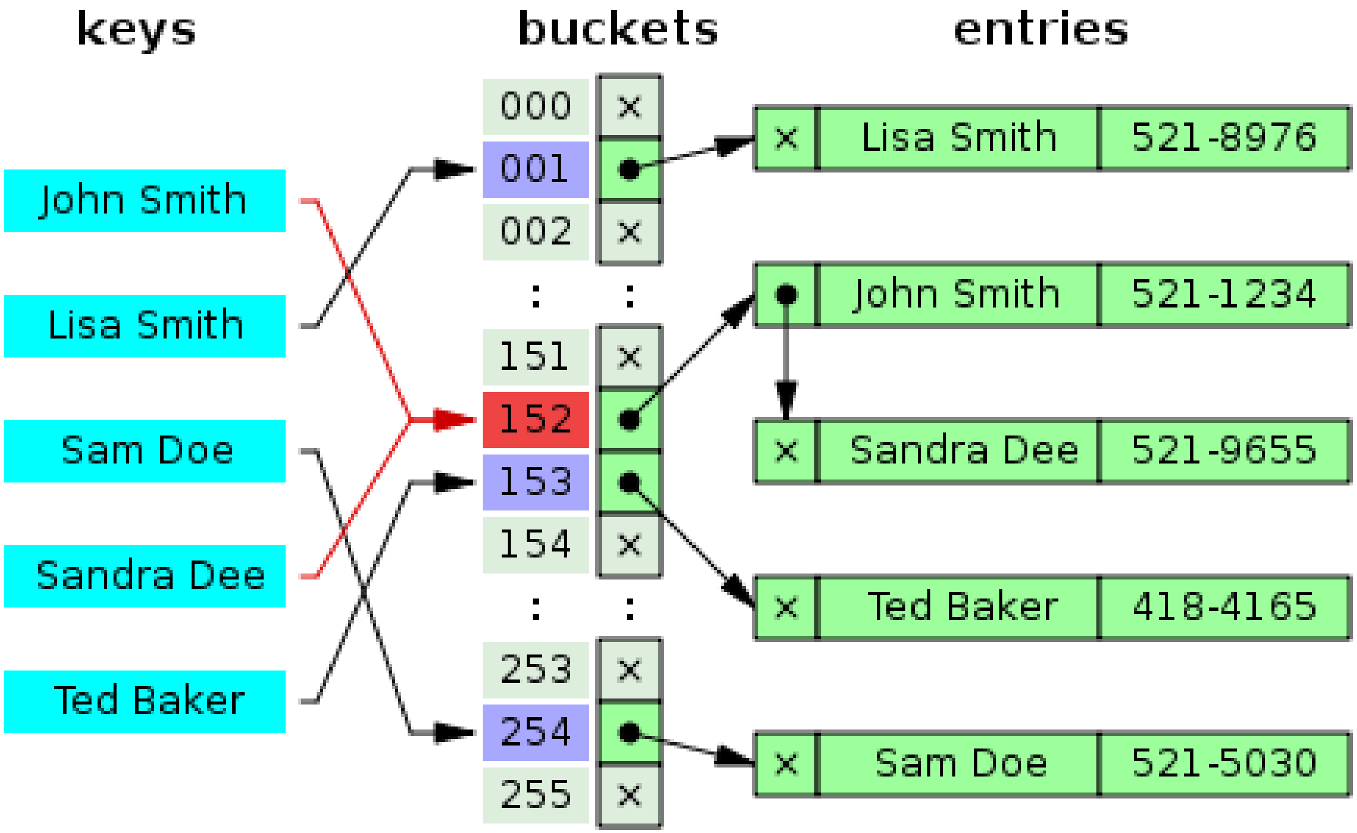
TreeSet and TreeMap were built around binary trees.

A Binary Tree is a group of nodes that contain left and right references. Each item is inserted into the tree according to its relationship to the other nodes.

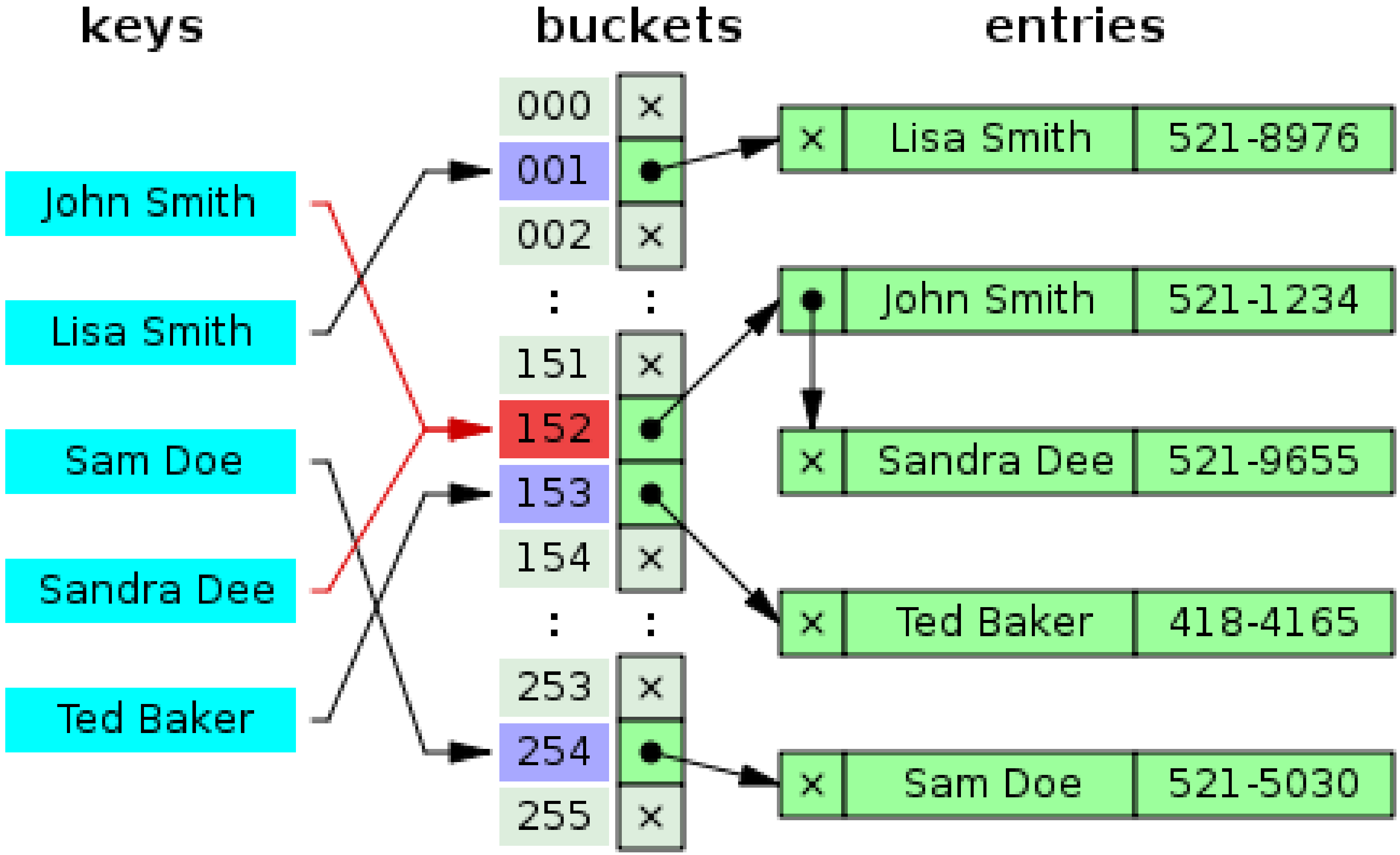


HashTable

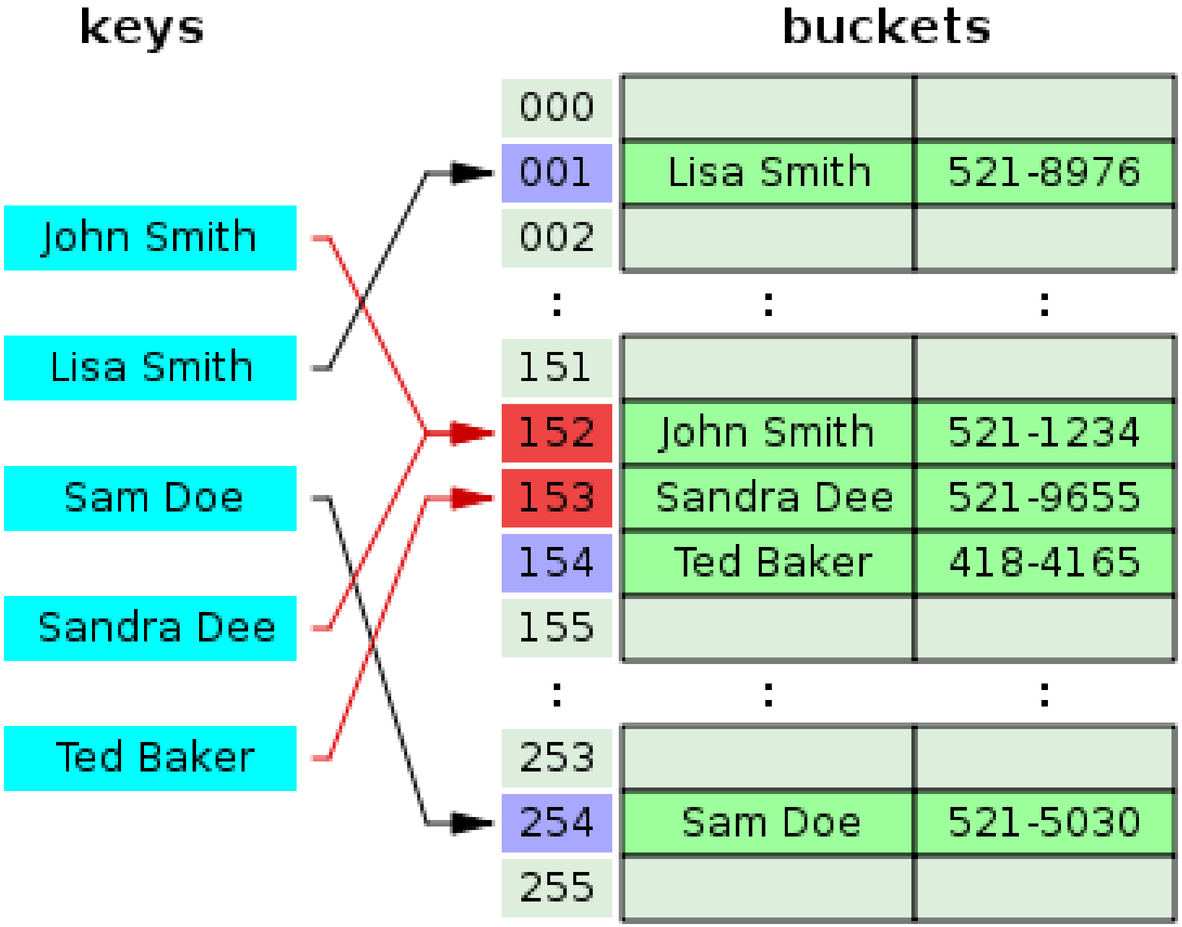
The hash function might produce the same hashcode for different keys, which is called a collision.



In the strategy known as *separate chaining*, each slot of the bucket array is a pointer to a linked list that contains the key-value pairs that hashed to the same location.



With open addressing, all entry records are stored in the bucket array itself. When a new entry has to be inserted, the buckets are examined, starting with the hashed-to slot and proceeding until an unoccupied slot is found.



	buckets	
000		
001	Lisa Smith	521-8976
002		
:	:	:
151		
152	John Smith	521-1234
153	Sandra Dee	521-9655
154	Ted Baker	418-4165
155		
:	:	:
253		
254	Sam Doe	521-5030
255		

What is a hashcode?

A hashcode is used to create a key
for an item.

The hashcode may be the same as the value of the item or may involve the use of some elaborate formula in hopes of creating a unique key.

What is a hashcode?

```
Character c = new Character('a');
out.println(c.hashCode());

c = new Character('0');
out.println(c.hashCode());

c = new Character('A');
out.println(c.hashCode());
```

OUTPUT

```
97
48
65
```

What is a hashcode?

```
Integer num = 45;
out.println(num.hashCode());

num = new Integer(101);
out.println(num.hashCode());

int n = new Integer(50);
out.println(n.hashCode());
```

OUTPUT

```
45
101
error
```

Example HashCode Methods

```
public int hashCode( )
{
   return  x ~~% someSize~~;
}

public int hashCode( )
{
   return (numVowels(s) + s.length) ~~% aSize~~;
}
```

Example HashCode Methods

```
public int hashCode( )
{
   return  x;
}

public int hashCode( )
{
   return (numVowels(s) + s.length);
}
```

Hashcodes for Strings

```
String s1 = "A";
out.println(s1.hashCode());

String s2 = "a";
out.println(s2.hashCode());

String s3 = "act";
out.println(s3.hashCode());
```

OUTPUT

```
65
97
96402
```

Hashcodes for Strings

The String class’s hashcode method returns:

s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]

```
String[] words;
words = {"Computer", "Science",
         "computer", "science", . . .

for(String word : words)
{
  int h = word.hashCode();
  out.print(word + " " + h);
```

word	**hashcode(h)**	**h%19**	**(h&0x7fffffff)%19**
Computer	-534518981	-18	4
Science	-712232380	-14	8
computer	-599163109	-9	13
science	1918081636	1	1
BFND	2035962	17	17
bfnd	3021050	12	12
Cy-Fair	-1453895141	-17	5
Bobcats	1717087858	3	3
at	3123	7	7
art	96867	5	5
an	3117	1	1
in	3365	2	2
am	3116	0	0

```
   7FFFFFFF
&         7
-----------
          7
```

```
 2147483647
&         7
-----------
          7
```

```
  01111111111111111111111111111111
& 00000000000000000000000000000111
----------------------------------
  00000000000000000000000000000111
```

```
   7FFFFFFF
&        -7
-----------
   7FFFFFF9
```

```
 2147483647
&        -7
-----------
 2147483641
```

```
  01111111111111111111111111111111
& 11111111111111111111111111111001
  --------------------------------
  01111111111111111111111111111001
```

Open hashcode.java

What is a hash table?

A hash table stores values based on the hash code of each value.

```
Object[] hashTable = new Object[10];

Character c = new Character('1');
hashTable[c.hashCode()%10] = c;
```

What is a hash table?

```
Object[] hashTable = new Object[10];

Character c = new Character('1');
hashTable[c.hashCode()%10] = c;

Integer num = new Integer(113);
hashTable[num.hashCode()%10] = num;

String s = "e";
hashTable[s.hashCode()%10] = s;

for( Object thing : hashTable )
{
  System.out.println(thing);
}
```

OUTPUT

```
null
e
null
113
null
null
null
null
null
1
```

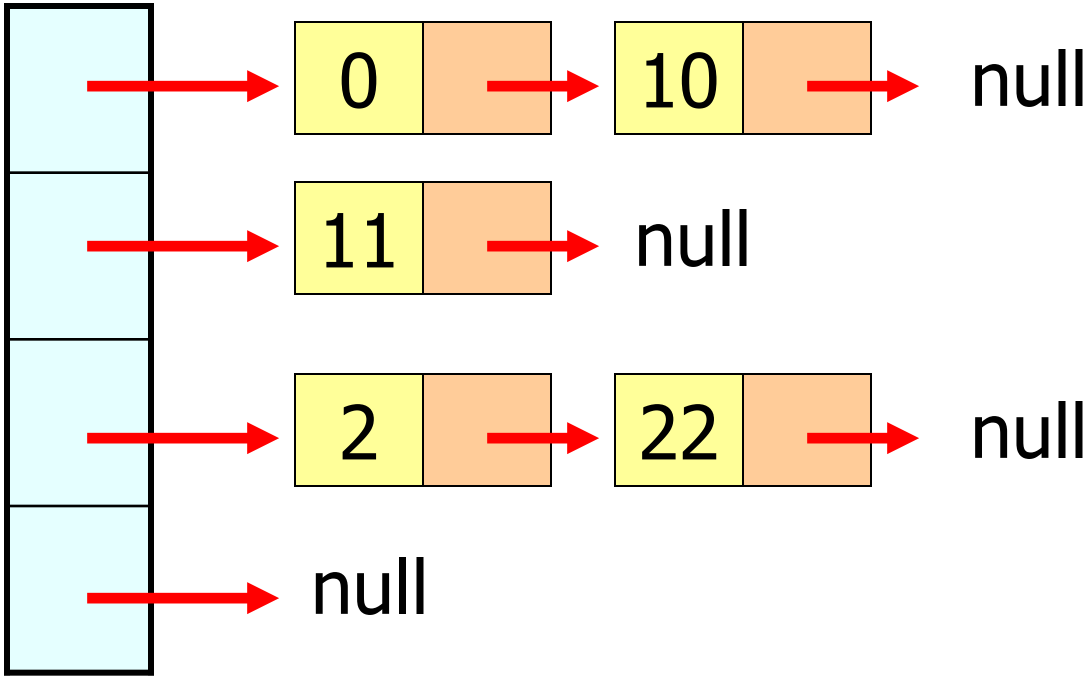
Open hashtableone.java

What is a hash table?

Most hash tables are built using an array of linked lists.

```
LinkedList[]   table;

table  = new LinkedList[10];
```

What is a hash table?



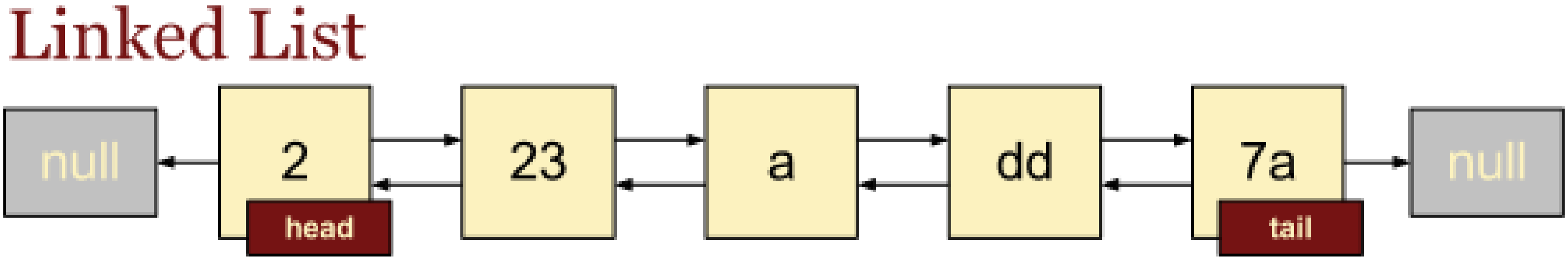
What is a hash table?

```
LinkedList[] hashTable = new LinkedList[5];

for( LinkedList list : hashTable )
{
  System.out.println(list);
}

for(int i = 0; i< hashTable.length; i++)
{
  hashTable[i] = new LinkedList();
}

for( LinkedList list : hashTable )
{
  System.out.println(list);
}
```

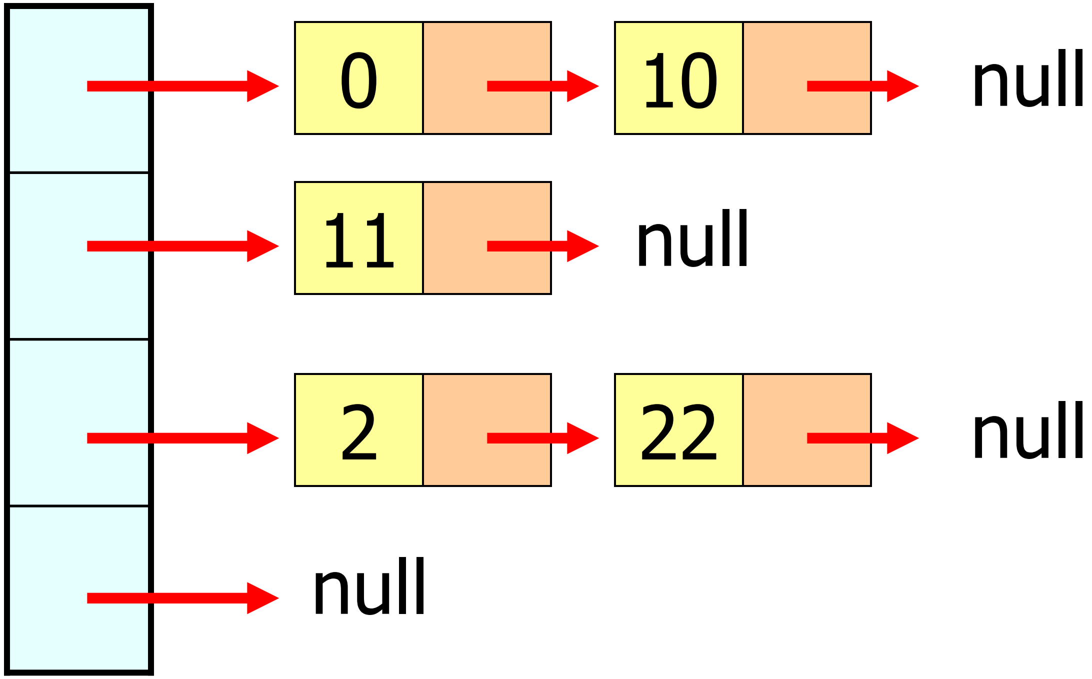
OUTPUT

```
null
null
null
null
null
[]
[]
[]
[]
[]
```

Java’s LinkedList Class



What is a hash table?



Java’s LinkedList Class

add(E element)

add(int index, E element)

addFirst(E element)

addLast(E element)

getFirst()

getLast()

removeFirst()

removeLast()

set(int index, E element)

Open hashtabletwo.java

Open hashtablethree.java

Open hashtablefour.java

HashTable

HashSet and HashMap were both created around hash tables.

A hash table is a giant array. Each item is inserted into the array according to a hash formula.

0	1	2	3	4

MAPS

Key	Value
restroom	bano
cat	gato
boy	muchacho
house	casa
toad	sapo
water	agua

Hash tables are similar to maps.

Both structures store each key and the value associated with that key.

HashTable

Key	Value
0	100
1	31
2	22
3	93
4	64
5	5

In most hash tables, the key involves some manipulation of the hash code and the value is just the thing being stored.

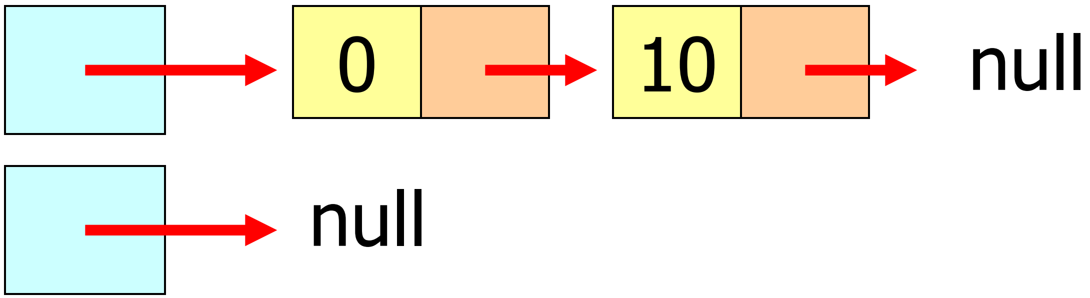
Collisions

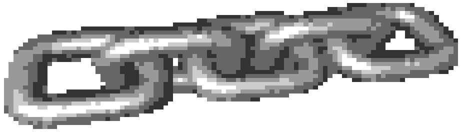
Anytime you use a hash formula to generate index positions, you could end up with the same index position for several values.

Collisions are handled by putting all the values with the same hash code in a list and storing the list at the index position that matches the hash code.

Bucket Hashing/Chaining

Using a bucket or a chain is a common way to describe putting all of the values with the same hash code in one list.

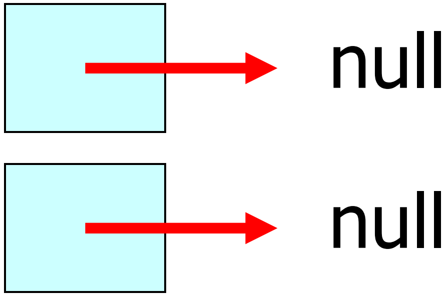




Bucket Hashing/Chaining

With only two buckets, this table would have lots of collisions and very long chains and be quite pointless.

```
LinkedList[]   table  = new LinkedList[2];
```



null

null

Bucket Hashing/Chaining

The larger the array, the less likely you are to have collisions. The table will be much more efficient as well.

```
LinkedList[]   table  = new LinkedList[100];

LinkedList[]   bigTable  = new LinkedList[1000];
```

Using ListNode

The LinkedList class works well to chain items together, but ListNode could also be used.

```
ListNode[]   table  = new ListNode[100];

ListNode[]   bigTable  = new ListNode[1000];
```

Open hashtablefive.java

HashTable
Big O Notation

Best Case / Average Case BigO of O(1)

Worst Case - BigO of O(N)

The Java HashSet and HashMap classes have a BigO of O(1). This is considered a generous BigO.

Start work on Lab 16