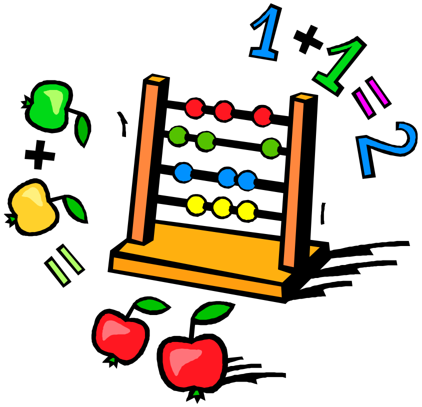
Sets



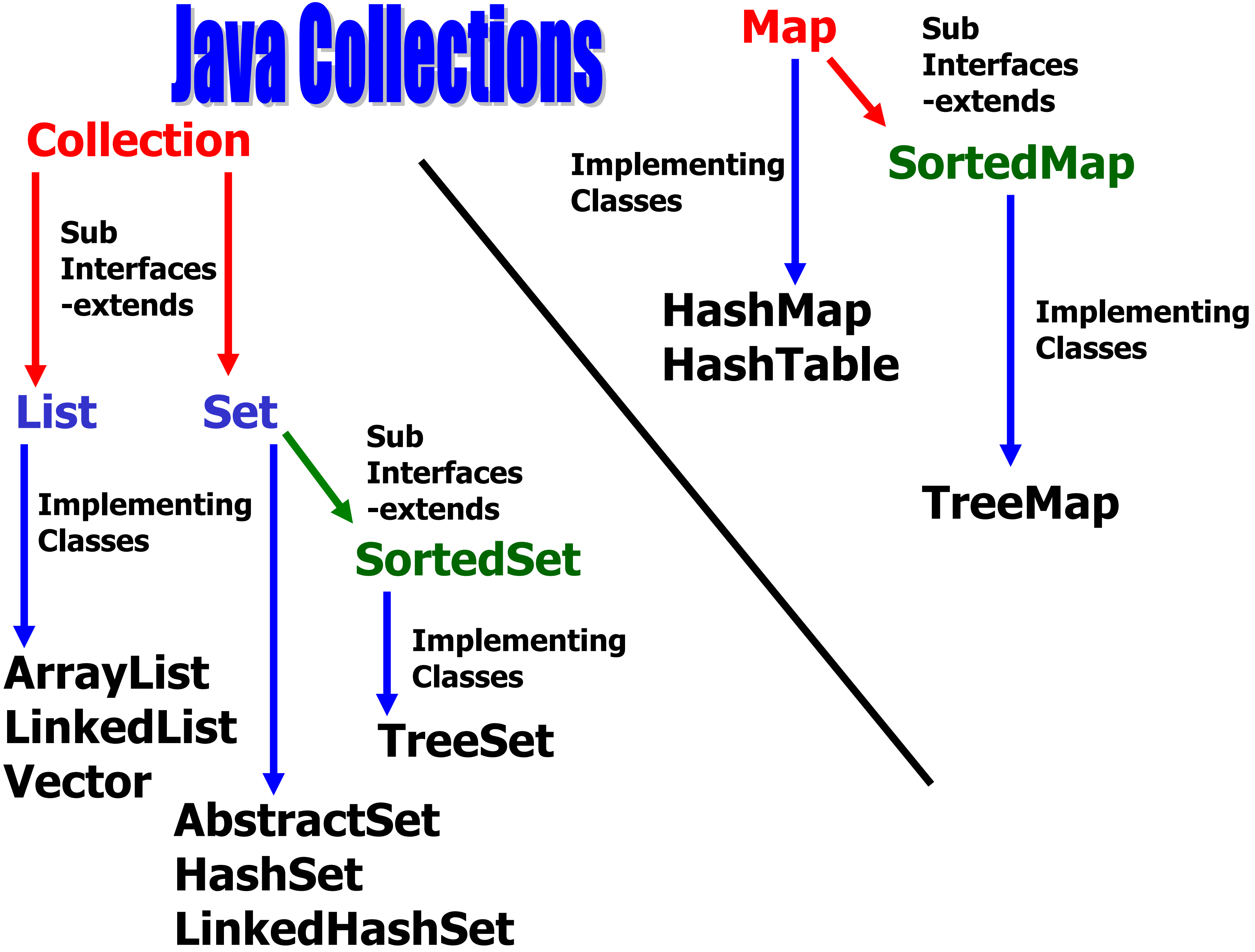
Lab 07

Java Interfaces

The following are important interfaces included in the Java language ::

Collection
Set
Map

Java Collections



Collection

Sub Interfaces -extends

List

Set

Implementing Classes

ArrayList
LinkedList
Vector

AbstractSet
HashSet
LinkedHashSet

Sub Interfaces -extends

SortedSet

Implementing Classes

TreeSet

Map

Implementing Classes

HashMap
HashTable

Sub Interfaces -extends

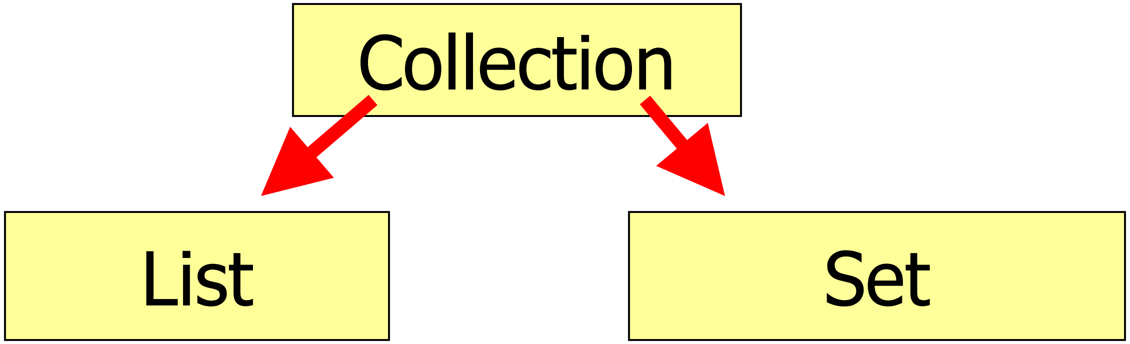
SortedMap

Implementing Classes

TreeMap

The Collection Interface

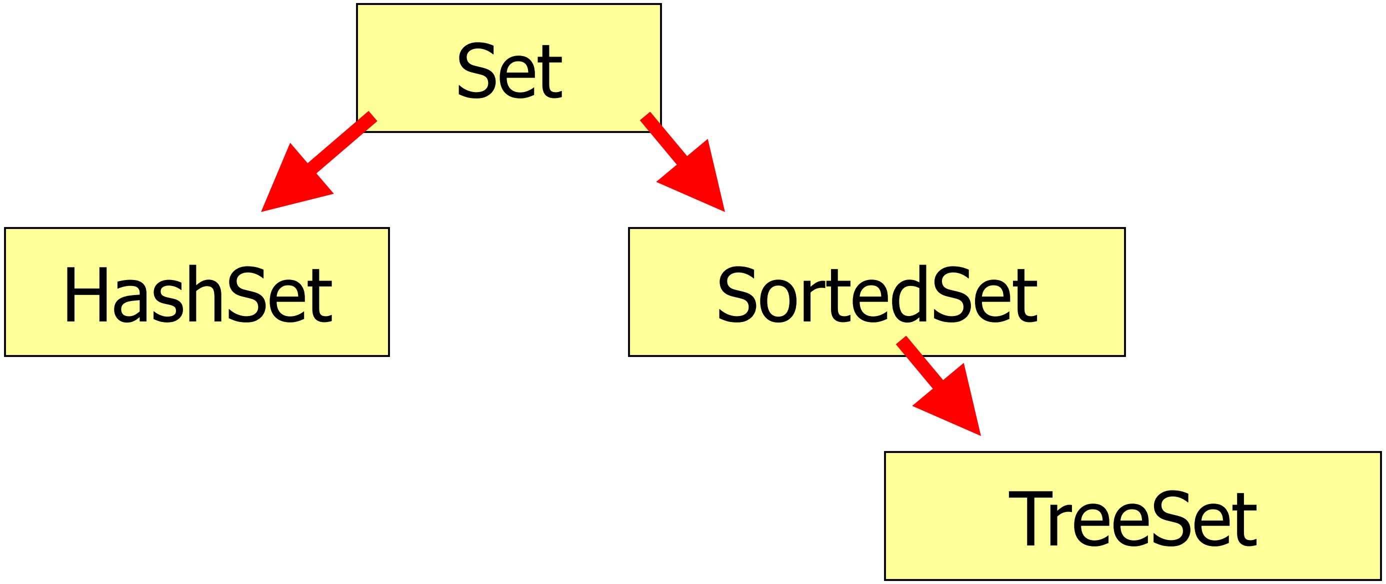
The Collection interface is the parent of List and Set. The Collection interface has many methods listed including add(), clear(), remove(), and size().



others not shown

The Set Interface

The Set interface extends the Collection interface.



Set

A set is a group of items all of the same type of which none are duplicates.

Set

Because Set is an interface, you cannot instantiate it.

```
Set bad = new Set();                //illegal

Set hash = new HashSet();           //legal
Set tree = new TreeSet();           //legal
```

hash and tree store Object references.

Set

With Java 5, you can now specify which type of reference you want to store in the TreeSet or HashSet.

```
Set<Byte> bytes = new TreeSet<Byte>();
Set<It> its = new HashSet<It>();
```

Set

HashSet – a set ordered by each item's hashCode that is extremely time efficient.

TreeSet – a naturally ordered set that is very efficient, but not as efficient as HashSet.

HashTable

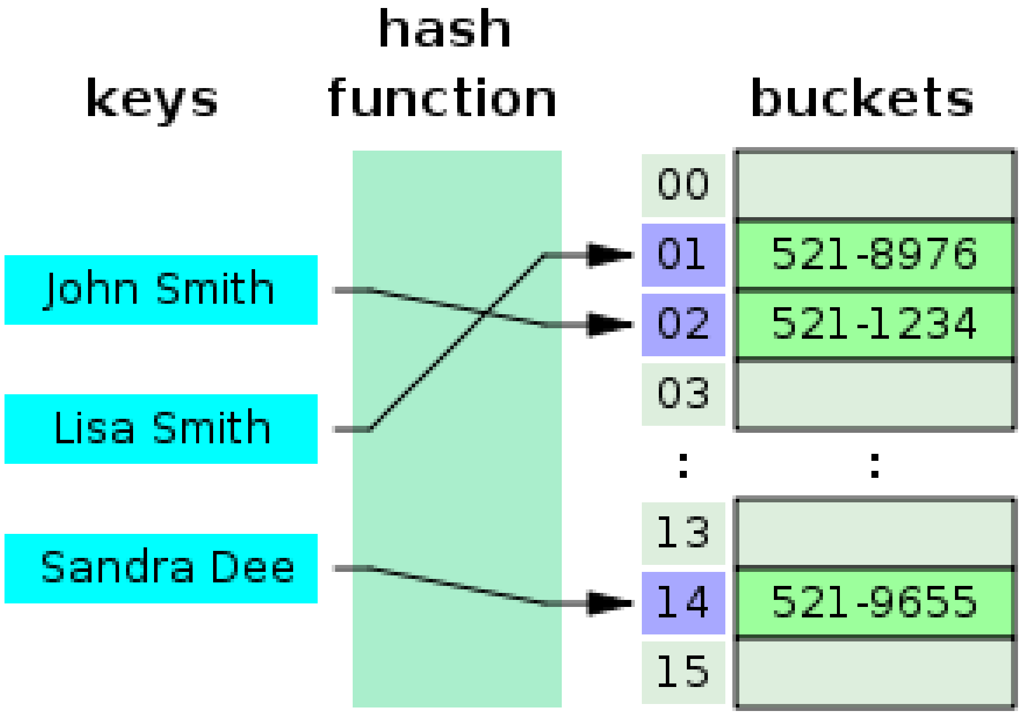
HashSet and HashMap were both created around hash tables.

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

0	1	2	3	4

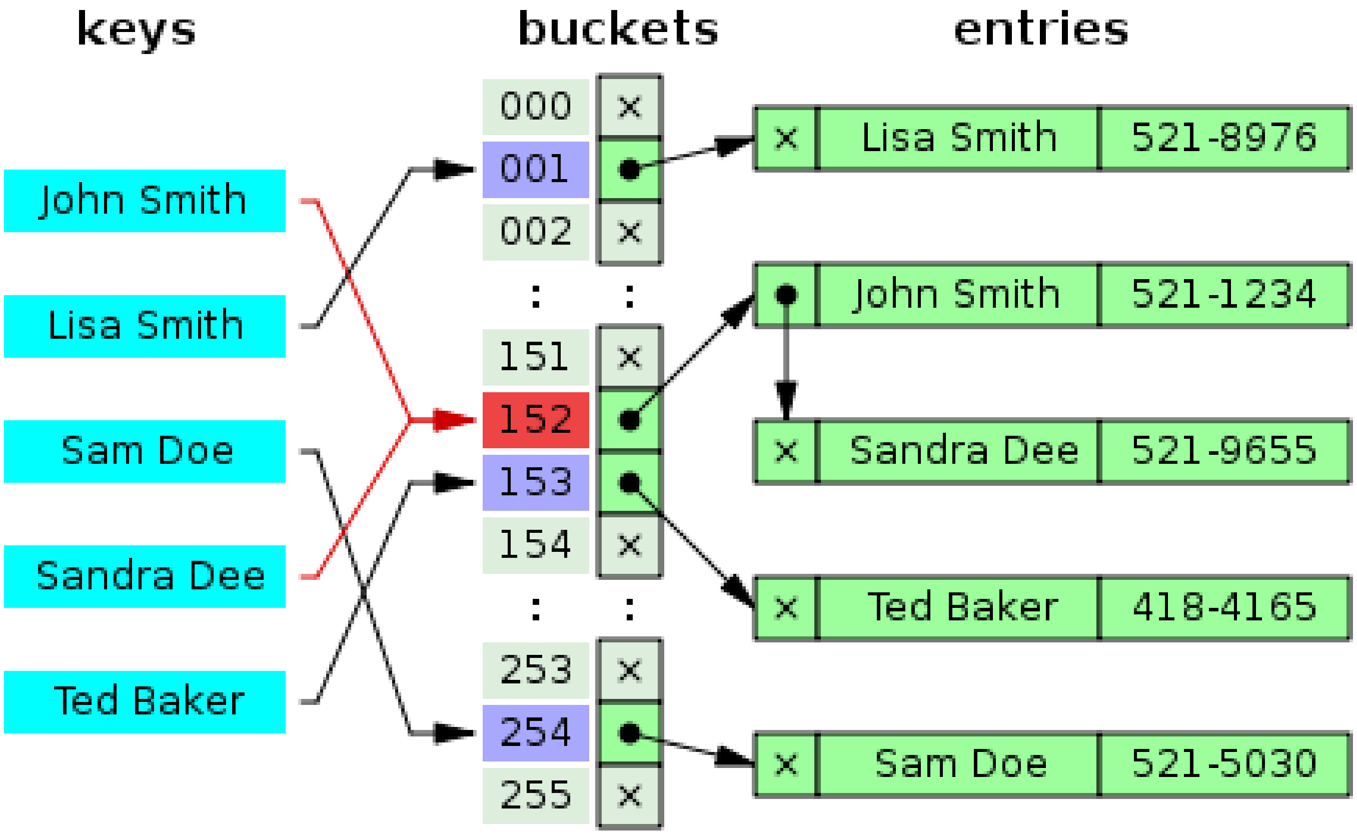
HashTable

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

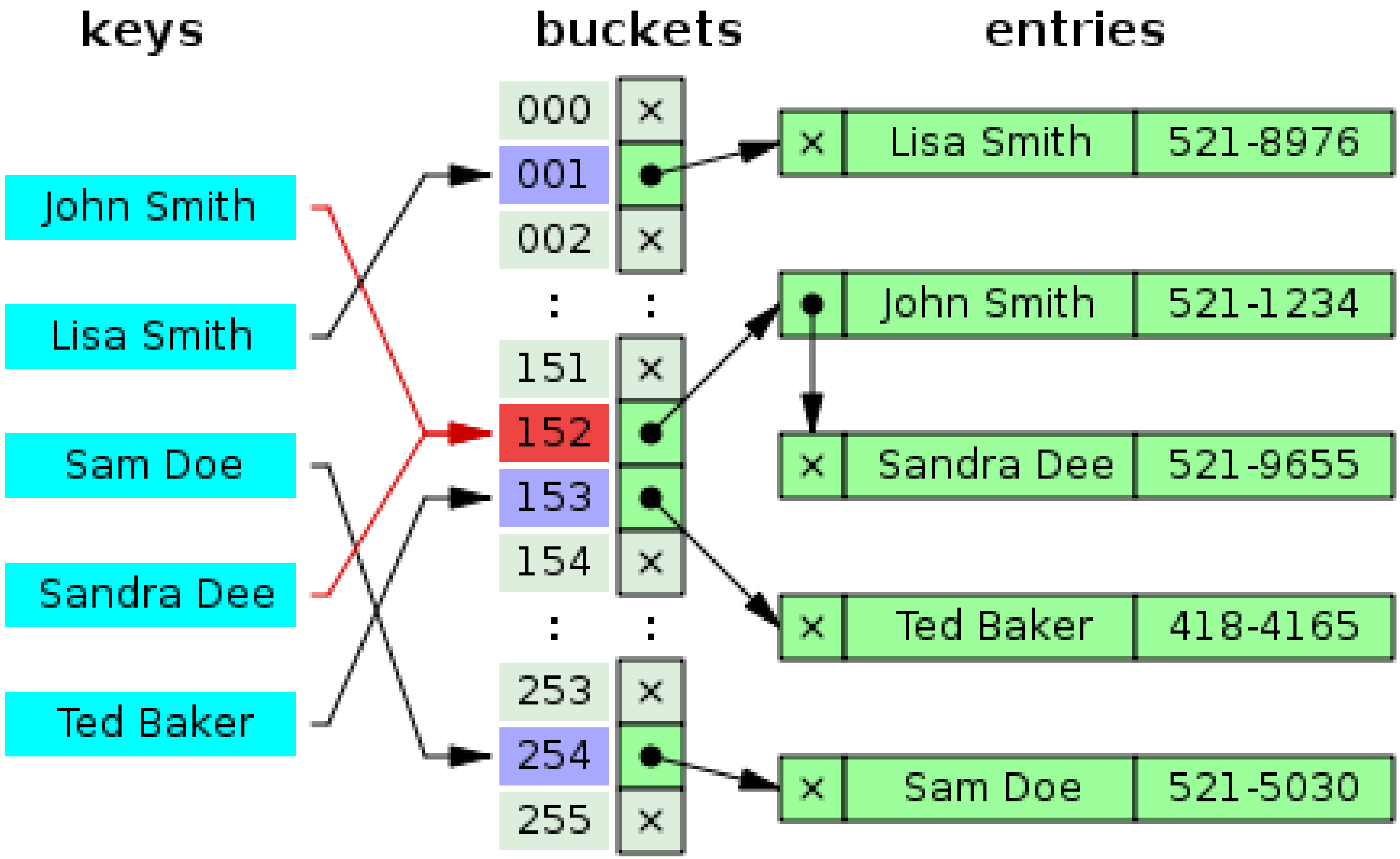


HashTable

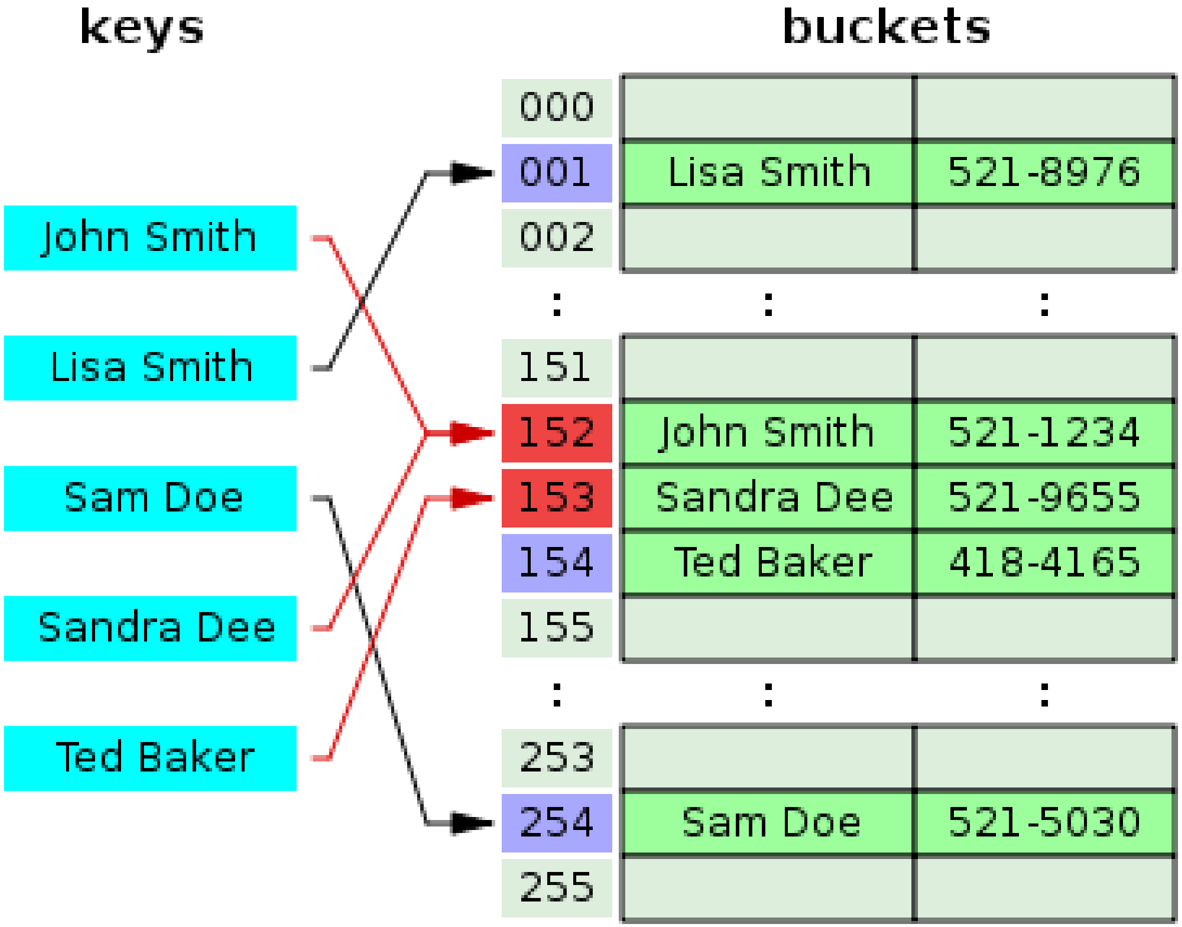
The hash function might produce the same hashcode for different keys, which is called a collision.



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.

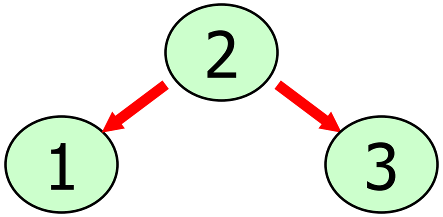


000		
001	Lisa Smith	521-8976
002		
:	:	:
151		
152	John Smith	521-1234
153	Sandra Dee	521-9655
154	Ted Baker	418-4165
155		
:	:	:
253		
254	Sam Doe	521-5030
255		

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.



Set Methods

Set

frequently used methods

	Name	Use
boolean	add(x)	adds item x to the set
boolean	remove(Object)	removes an item from the set
void	clear()	removes all items from the set
Int	size()	returns the # of items in the set

Why does the Set interface not include corollaries to these List methods?

get(int index)

remove(int index)

add(int index, E element)

set(int index, E element)

HashSet add

```
Set<Integer> intSet;
intSet = new HashSet<Integer>();
System.out.println(intSet.add(45));
intSet.add(12);
System.out.println(intSet.add(12));
intSet.add(23);
System.out.println(intSet);
```

OUTPUT

```
true
false
[45, 23, 12]
```

HashSet add

```
Set<String> stringSet;
stringSet = new HashSet<String>();
stringSet.add("AB");
stringSet.add("23");
stringSet.add("ab");
System.out.println(stringSet);
```

OUTPUT

```
[ab, 23, AB]
```

Open hashsetint.java hashsetstring.java

Quick Creation of a Set

```
String[] words = “I am Crazy I am".split(" ");
List<String> wordList = Arrays.asList(words);
Set<String> set = new TreeSet<String>(wordList);
System.out.println(set);
```

OUTPUT

```
[Crazy, I, am]
```

Open SetSplit.java

HashSet remove()

```
Set<Double> doubleSet;
doubleSet = new HashSet<Double>();
doubleSet.add(2.5);
doubleSet.add(5.8);
doubleSet.add(7.3);
System.out.println(doubleSet);
doubleSet.remove(5.8);
System.out.println(doubleSet.remove(0));
System.out.println(doubleSet);
```

OUTPUT

```
[7.3, 2.5, 5.8]
false
[7.3, 2.5]
```

Open hashsetremove.java

TreeSet add

```
Set<Integer> intSet;
intSet = new TreeSet<Integer>();
System.out.println(intSet.add(45));
intSet.add(12);
System.out.println(intSet.add(12));
intSet.add(23);
System.out.println(intSet);
```

OUTPUT

```
true
false
[12, 23, 45]
```

TreeSet add

```
Set<String> stringSet;
stringSet = new TreeSet<String>();
stringSet.add("AB");
stringSet.add("23");
stringSet.add("ab");
System.out.println(stringSet);
```

OUTPUT

```
[23, AB, ab]
```

Open treesetint.java treesetstring.java

TreeSet remove()

```
Set<Double> doubleSet;
doubleSet = new TreeSet<Double>();
doubleSet.add(2.5);
doubleSet.add(5.8);
doubleSet.add(7.3);
System.out.println(doubleSet);
doubleSet.remove(5.8);
doubleSet.remove(0.0);
System.out.println(doubleSet);
```

OUTPUT

```
[2.5, 5.8, 7.3]
[2.5, 7.3]
```

Open treesetremove.java

set output

```
Set<Double> doubleSet;
doubleSet = new TreeSet<Double>();
doubleSet.add(7.3);
doubleSet.add(2.5);
doubleSet.add(5.8);

Iterator<Double> it;
it = doubleSet.iterator();
while(it.hasNext()){
   System.out.println(it.next());
}
```

OUTPUT

```
2.5
5.8
7.3
```

Open

setoutput.java

set output new

```
Set<Double> doubleSet;
doubleSet = new TreeSet<Double>();
doubleSet.add(5.8);
doubleSet.add(7.3);
doubleSet.add(2.5);

for(double dec : doubleSet)
{
  System.out.println(dec);
}
```

OUTPUT

```
2.5
5.8
7.3
```

Open setoutputnew.java setsplit.java

BigO

Big-O Notation

Big-O notation is an assessment of an algorithm’s efficiency. Big-O notation helps gauge the amount of work that is taking place.

Actual runtime function	Big-Oh notation
31, 500, etc.	**$O(1)$**
$3n + 1$, $5n – 2$, etc.	**$O(N)$**
$n^2 + 5n – 9$, $4n^2 + 12$, etc.	**$O(N^2)$**
$5n^3 – 4n^2 + 5n – 9$, etc.	**$O(N^3)$**

Big-O

frequently used notations

Name	Notation
constant	$O(1)$
logarithmic	$O(\log_2 N)$
linear	$O(N)$
linearithmic	$O(N \log_2 N)$
quadratic	$O(N^2)$
exponential	$O(2^n)$

Big-O

frequently used notations

n	O(1)	$O(\log_2 N)$	O(N)	$O(N^2)$	$O(2^n)$
16	1	4	16	256	65536
128	1	7	128	16384	3.4E38
1024	1	10	1024	1048576	> 1E99
1048576	1	20	1048576	1.1E12	> 1E99
2^30	1	30	2^30	1.2E18	> 1E99

Examples

1. Search an unordered list of *n* elements.

Runtime: O(*n*)

2. Print the last five elements of an large array.

Runtime: O(1)

Java Collections

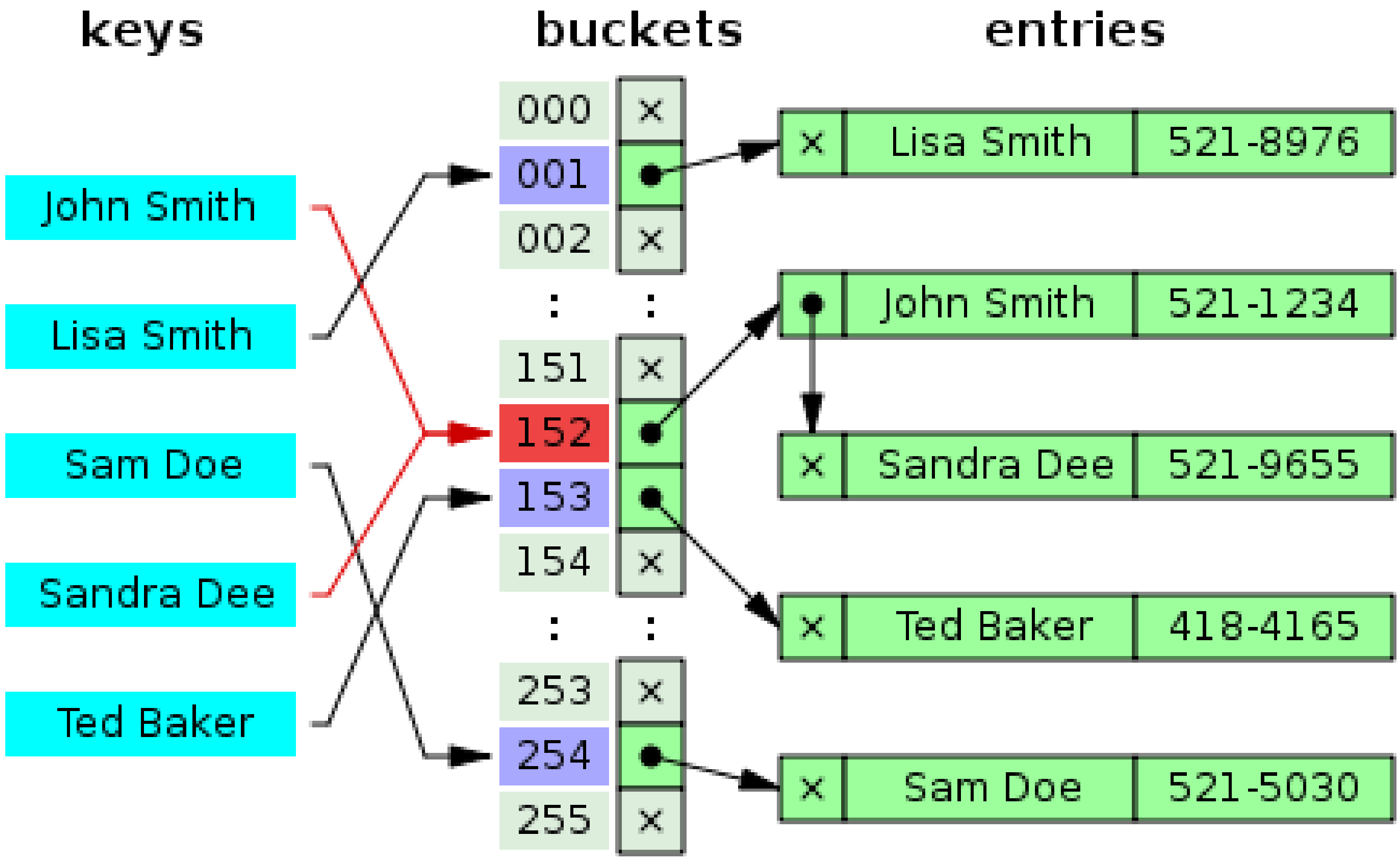
Set

	Tree Set	Hash Set
add	$O(Log_2N)$	O(1)
remove	$O(Log_2N)$	O(1)
contains	$O(Log_2N)$	O(1)

TreeSets are implemented with balanced binary trees (red/black trees).

HashSets are implemented with hash tables.

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.



Start work on Lab 7