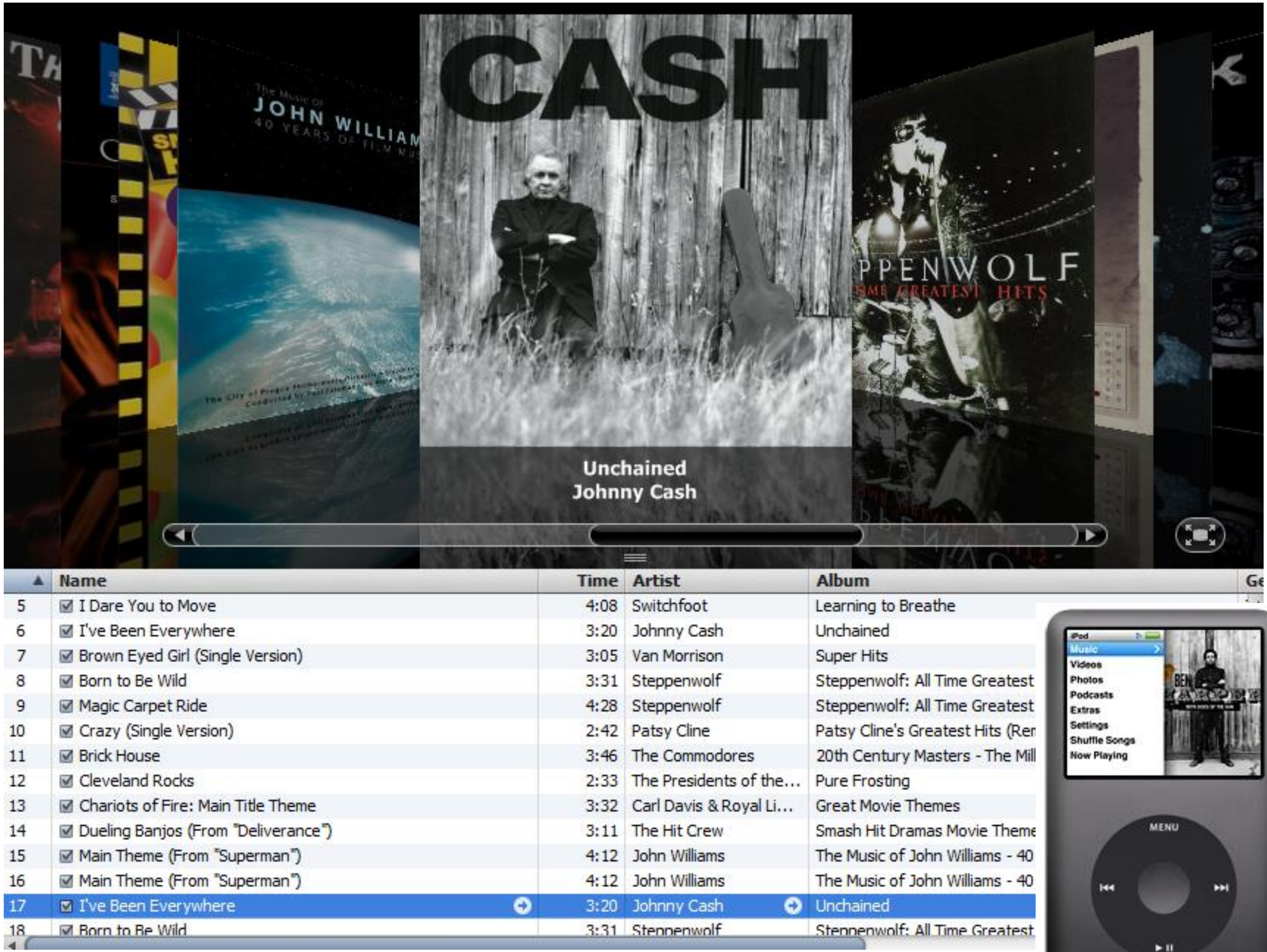
ArrayList and Lists

What is a list?



▲	Name	Time	Artist	Album	Ge
5	☑ I Dare You to Move	4:08	Switchfoot	Learning to Breathe	
6	☑ I've Been Everywhere	3:20	Johnny Cash	Unchained	
7	☑ Brown Eyed Girl (Single Version)	3:05	Van Morrison	Super Hits	
8	☑ Born to Be Wild	3:31	Steppenwolf	Steppenwolf: All Time Greatest	
9	☑ Magic Carpet Ride	4:28	Steppenwolf	Steppenwolf: All Time Greatest	
10	☑ Crazy (Single Version)	2:42	Patsy Cline	Patsy Cline's Greatest Hits (Rer	
11	☑ Brick House	3:46	The Commodores	20th Century Masters - The Mill	
12	☑ Cleveland Rocks	2:33	The Presidents of the…	Pure Frosting	
13	☑ Chariots of Fire: Main Title Theme	3:32	Carl Davis & Royal Li…	Great Movie Themes	
14	☑ Dueling Banjos (From "Deliverance")	3:11	The Hit Crew	Smash Hit Dramas Movie Theme	
15	☑ Main Theme (From "Superman")	4:12	John Williams	The Music of John Williams - 40	
16	☑ Main Theme (From "Superman")	4:12	John Williams	The Music of John Williams - 40	
17	☑ I've Been Everywhere	3:20	Johnny Cash	Unchained	
18	☑ Born to Be Wild	3:31	Steppenwolf	Steppenwolf: All Time Greatest	

What is an ArrayList?

ArrayList

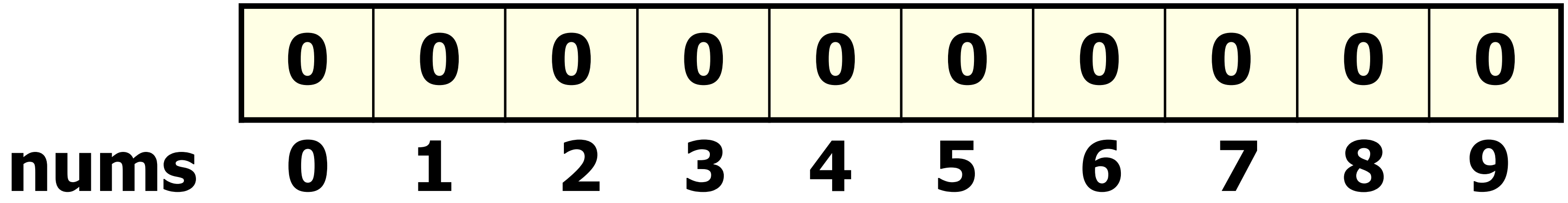
Arraylist is a class that houses an array.

An ArrayList can store any type.

All ArrayLists store the first reference at spot / index / position 0.

What is an array?

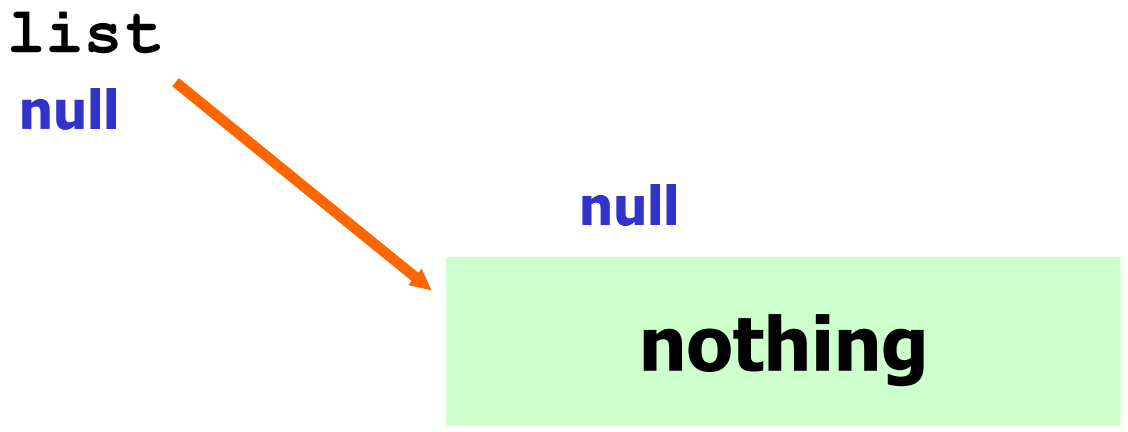
```
int[] nums = new int[10];        //Java int array
```



An array is a group of items all of the same type which are accessed through a single identifier.

ArrayList References (Declarations)

ArrayList list;



list is a reference to an ArrayList.

ArrayList Instantiation

```
new ArrayList();
```

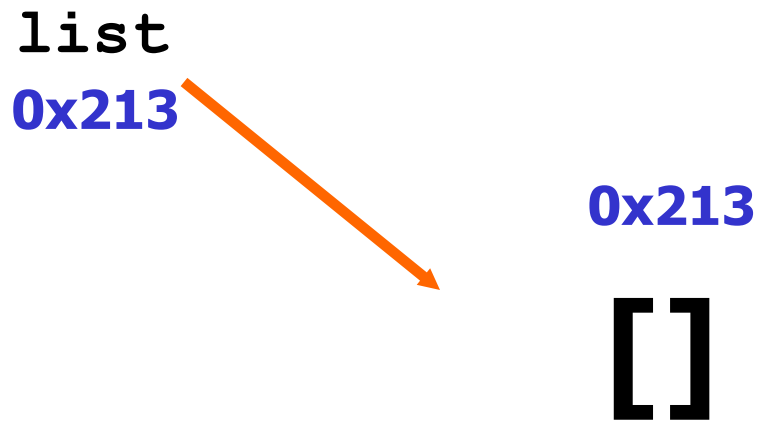
0x213

[]

ArrayLists are Objects.

ArrayList

ArrayList list = new ArrayList();



list is a reference to an ArrayList.

ArrayList

```
List ray = new ArrayList();

ray.add("hello");
ray.add("APCS");
ray.add("Patriots!");

System.out.println(((String)ray.get(0)).charAt(0));
System.out.println(((String)ray.get(2)).charAt(0));
```

OUTPUT

```
h
P
```

ray stores Object references.

Note: If no type at instantiation, then you MUST first cast the Object to call specific methods.

Generic ArrayLists

ArrayList

Since Java 5, you can specify which type of reference you want to store in the ArrayList. You should always do this… that way you don't need to cast them later.

```
ArrayList<String> words;
words = new ArrayList<String>();

List<Double> decNums;
decnums = new ArrayList<Double>();

List<It> itList;
itList = new ArrayList<It>();
```

ArrayList

```
List<String> ray;
ray = new ArrayList<String>();

ray.add("hello");
ray.add("APCS");
ray.add("Patriots!");

System.out.println(ray.get(0).charAt(0));
System.out.println(ray.get(2).charAt(0));
```

OUTPUT

```
h
P
```

ray stores String references, so no need to cast ☺

ArrayList Methods

ArrayList

frequently used methods

Name	Use
add(item)	adds item to the end of the list (appends)
add(index,item)	adds item at index – shifts items up-> (inserts)
set(index,item)	put item at index (overwrites/assigns) z[index]=item
get(index)	returns the item at index return z[index]
size()	returns the # of items in the list
remove()	removes an item from the list
clear()	removes all items from the list

```
import  java.util.ArrayList;
```

add() Ex1

```
ArrayList<String> words;
words = new ArrayList<String>();

words.add("it");
words.add("is");
words.add(0,"a");
words.add(1,"lie");

System.out.println(words);
```

OUTPUT

```
[a, lie, it, is]
```

add() Ex2

```
List<Integer> nums;
nums = new ArrayList<Integer>();

nums.add(34);
nums.add(0,99);
nums.add(21);
nums.add(0,11);

System.out.println(nums);
```

OUTPUT

```
[11, 99, 34, 21]
```

set()

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.set(0,66);
ray.add(53);
ray.set(1,93);
ray.add(22);

System.out.println(ray);
```

OUTPUT

```
[66, 93, 53, 22]
```

set()

```
List<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(0, 11);
ray.set(5,66);

System.out.println(ray);
```

OUTPUT

```
Runtime exception
```

Note: You cannot set a location to a value if the location does not already exist. If you try… you will get an index out of bounds exception.

get()

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(12);
ray.add(65);

System.out.println(ray.get(0));
System.out.println(ray.get(3));
```

OUTPUT

```
23
65
```

.get(index)
returns the reference stored at the index!

get()

```
List<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(12);
ray.add(65);

for(int i=0; i<ray.size(); i++)
   System.out.println(ray.get(i));
```

OUTPUT

```
23
11
12
65
```

.get(index)
returns the reference stored at the index!

Processing a list using loops

Traditional for loop

```
for (int i=0; i<ray.size(); i++)
{
    System.out.println(ray.get(i));
}
```

.size( ) returns the number of elements/items/spots/boxes or whatever you want to call them.

for each loop

```
List<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(53);

for(int num : ray)
{
  System.out.println(num);
}
```

OUTPUT

```
23
11
53
```

remove
methods

remove() Ex 1

```
ArrayList<String> ray;
ray = new ArrayList<String>();

ray.add("a");
ray.add("b");
ray.remove(0);
ray.add("c");
ray.add("d");
ray.remove(0);

System.out.println(ray);
```

OUTPUT

```
[c, d]
```

remove() Ex 2

```
List<String> ray;
ray = new ArrayList<String>();

ray.add("a");
ray.add("b");
ray.remove("a");
ray.add("c");
ray.add("d");
ray.remove("d");

System.out.println(ray);
```

OUTPUT

```
[b, c]
```

Removing Multiple Items

Removing multiple items

```
// Psuedocode / Algorithm:
spot = list size – 1
while( spot is >=0 )
{
   if ( this item is a match )
     remove this item from the list
   subtract 1 from spot
}
```

***Keep in mind that the ArrayList shrinks when items are removed.
The items in the ArrayList shift down towards spot 0.
The loop must start at size()-1 and go down in order to account for the shift.

Removing multiple items

```
spot = list.size() – 1;

while( spot >= 0 )
{
   if ( list.get(spot).equals( value ) )
     list.remove( spot );
   spot = spot – 1;
}
```

Open removeall.java

Complete the code

clear()

The `clear()` method removes all items from the ArrayList.

```
ArrayList<String> ray;
ray = new ArrayList<String>();

ray.add("a");
ray.add("x");
ray.clear();
ray.add("t");
ray.add("w");

System.out.println(ray);
```

OUTPUT

```
[t, w]
```

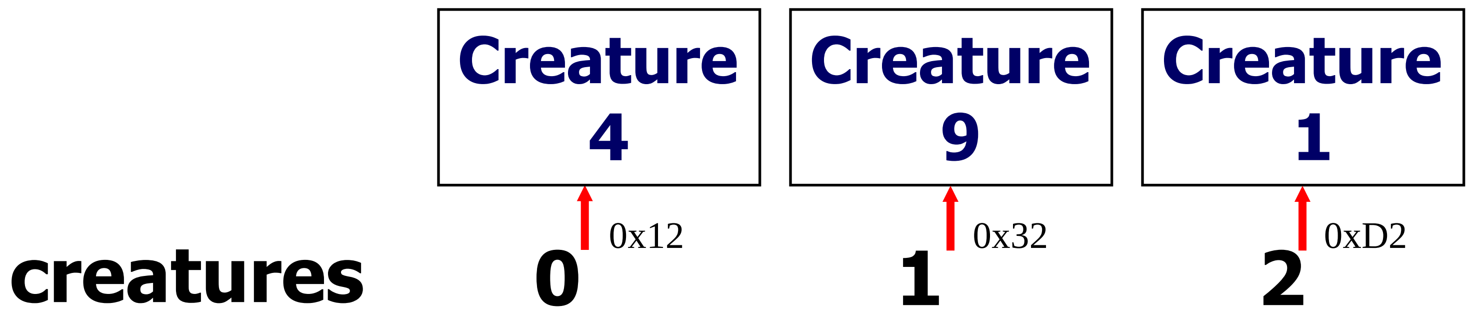
ArrayList with User-defined classes

ArrayList w/User Classes

```
public class Creature implements Comparable
{
   private int size;


    //checks to see if this Creature is big – size > x
   public boolean isBig()
            //implementation details not shown

   public boolean equals(Object obj)
            //implementation details not shown


    public int compareTo(Object obj)
            //implementation details not shown

    //other methods and constructors not shown
}
```

ArrayList w/User Classes

```
ArrayList<Creature> creatures;
creatures  = new ArrayList<Creature>();

creatures.add(new Creature(4));
creatures.add(new Creature(9));
creatures.add(new Creature(1));
```



ArrayList w/User Classes

```
ArrayList<Creature> creatures;
creatures  = new ArrayList<Creature>();
creatures.add( new Creature(41) );
creatures.add( new Creature(91) );
creatures.add( new Creature(11) );

System.out.println( creatures.get(0) );

creatures.get(0).setSize(79);
System.out.println( creatures.get(0) );

System.out.println( creatures.get(2) );
System.out.println( creatures.get(1).isBig() );
```

OUTPUT

```
41
79
11
true
```

ArrayList w/User Classes

```
creatures.get(0).setSize(7);
```

0x242

What does this return?

What does the . dot do?

0x242

Creature

The . dot grants access to the Object at the stored address.

ArrayList w/User Classes

```
/* method countBigOnes should return the count of
   big creatures - use the isBig() Creature method
*/
public int countBigOnes()
{
  int cnt = 0;

  //for each loop
     //if statement
         //increase cnt by 1

  return cnt;
}
```

CREATURE

Open
creature.java
herd.java
herdrunner.java
Complete the code

AutoBoxing
AutoUnboxing

Box/Unbox

primitive	object
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean
==	.equals()

Box/Unbox

Before Java 5 added in autoboxing and autounboxing, you had to manually wrap primitives.

```
Integer x = new Integer(98);
int y = 56;
x= new Integer(y);
```

Box/Unbox

Java now wraps automatically.

```
Integer numOne = 99;
Integer numTwo = new Integer(99);
```

```
=99;
=new Integer(99);
```

These two lines are equivalent.

Box/Unbox

Java now wraps automatically.

```
Double numOne = 99.1;
Double numTwo = new Double(99.1);

=99.1;
=new Double(99.1);
```

These two lines are equivalent.

Box/Unbox

Before Java 5 added in autoboxing and autounboxing, you had to manually unwrap references.

```
Integer ref = new Integer(98);
int y = ref.intValue();
```

Box/Unbox

Java now unwraps automatically.

```
Integer num = new Integer(3);
int prim = num.intValue();
System.out.println(prim);
prim = num;
System.out.println(prim);
```

OUTPUT

```
3
3
```

```
prim=num.intValue();
prim=num;
```

These two lines are equivalent.

Box/Unbox

```
Double dub = 9.3;
double prim = dub;
System.out.println(prim);

int num = 12;
Integer big = num;
System.out.println(big.compareTo(12));
System.out.println(big.compareTo(17));
System.out.println(big.compareTo(10));
```

OUTPUT

```
9.3
0
-1
1
```

new for loop

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

//add some values to ray

int total = 0;
for(Integer num : ray)
{
   //this line shows the AP preferred way
   //it shows the manual retrieval of the primitive
   total = total + num.intValue();

   //the line below accomplishes the same as the line above
   //but, it uses autounboxing to get the primtive value
   //total = total + num;
}
System.out.println(total);
```

OUTPUT

```
153
```

Collections class

Collections

frequently used methods

Name	Use
sort(x)	puts all items in x in ascending order
binarySearch(x,y)	checks x for the location of y
fill(x,y)	fills all spots in x with value y
rotate(x,y)	shifts items in x left or right y locations
reverse(x)	reverses the order of the items in x

```
import  java.util.Collections;
```

Collections

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(66);
ray.add(53);
Collections.sort(ray);

System.out.println(ray);
System.out.println(Collections.binarySearch(ray,677));
System.out.println(Collections.binarySearch(ray,66));
```

OUTPUT

```
[11, 23, 53, 66]
-5
3
```

Collections

```
ArrayList<Integer> ray;
ray = ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(53);
System.out.println(ray);
rotate(ray,2);
System.out.println(ray);
rotate(ray,2);
reverse(ray);
System.out.println(ray);
```

OUTPUT

```
[23, 11, 53]
[11, 53, 23]
[11, 23, 53]
```

Collections

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(0);
ray.add(0);
ray.add(0);
System.out.println(ray);

Collections.fill(ray,33);
System.out.println(ray);
```

OUTPUT

```
[0, 0, 0]
[33, 33, 33]
```

search methods

ArrayList frequently used methods	
Name	**Use**
contains(x)	checks if the list contains x
indexOf(x)	checks the list for the location of x

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();

ray.add(23);
ray.add(11);
ray.add(66);
ray.add(53);

System.out.println(ray);
System.out.println(ray.indexOf(21));
System.out.println(ray.indexOf(66));

System.out.println(ray);
System.out.println(ray.contains(21));
System.out.println(ray.contains(66));
```

OUTPUT

```
[23, 11, 66, 53]
-1
2
[23, 11, 66, 53]
false
true
```

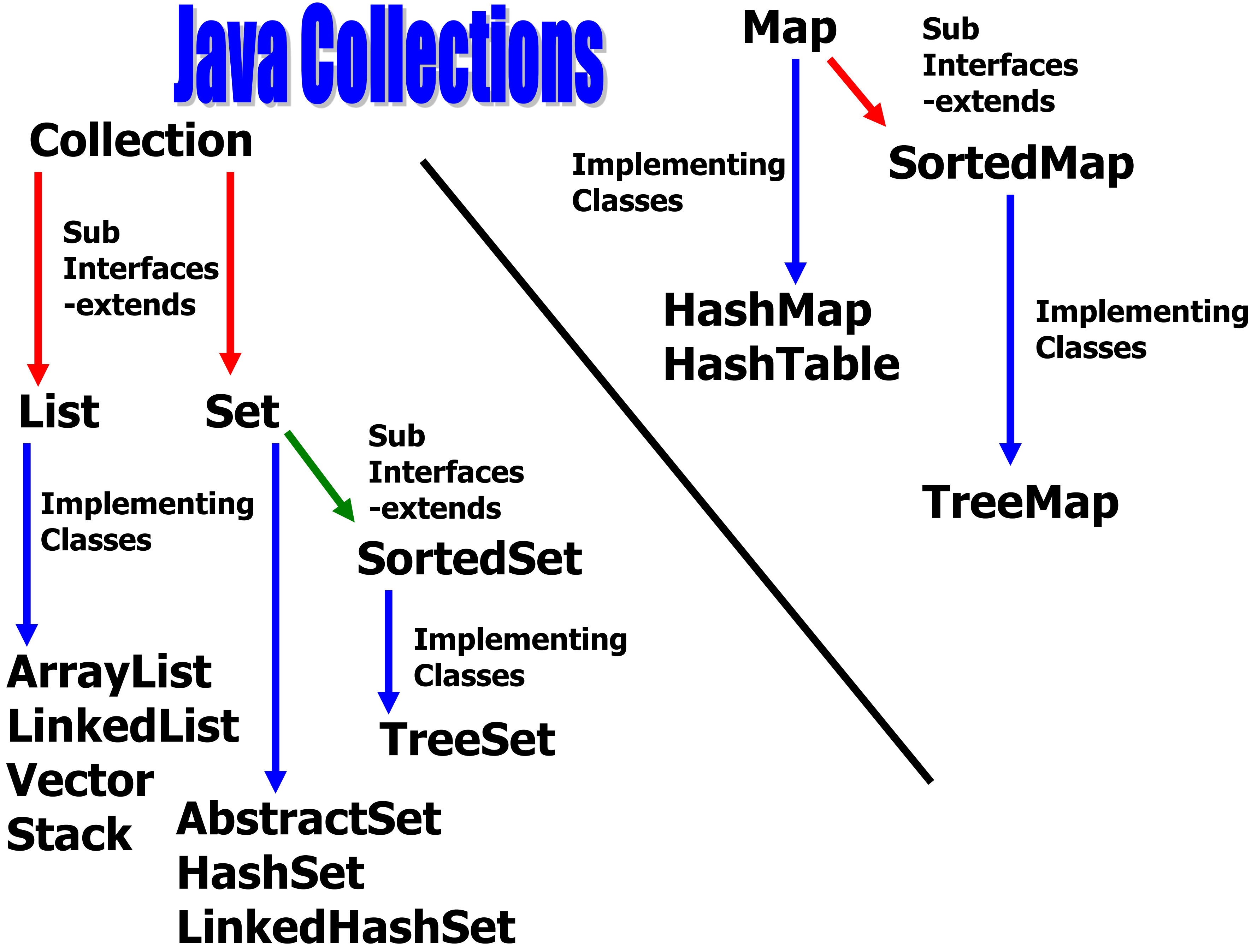
Java Collections

Java Interfaces

The following are important interfaces included in the Java language:

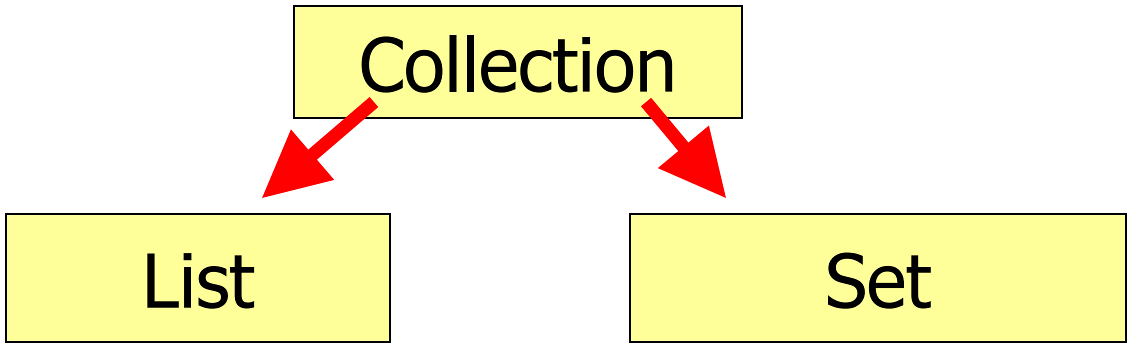
Collection
List

Java Collections



The Collection Interface

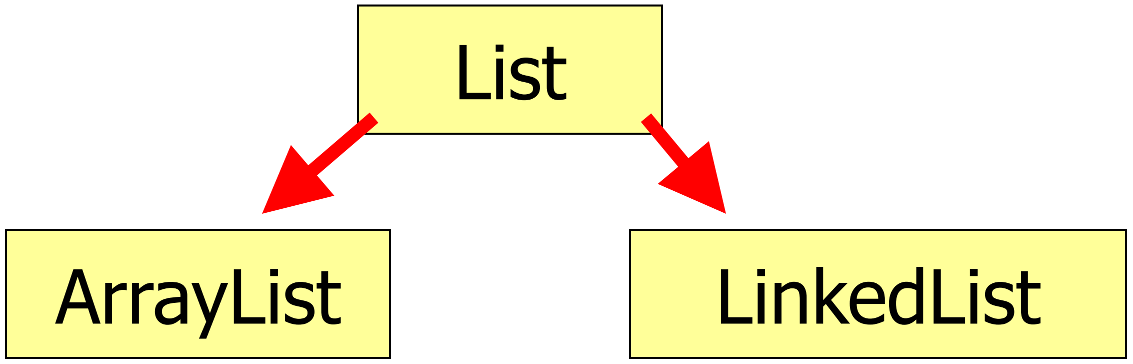
The Collection interface is the parent of List and Set. The Collection interface has many methods listed including add(), clear(), remove(), and size().



others not shown

The List Interface

The List interface extends the Collection interface. The List interface adds in the get() method as well as several others.



others not shown

ArrayList

ArrayList is a descendant of List and Collection, but because List and Collection are interfaces, you cannot instantiate them.

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

List ray = new ArrayList();          //legal
ArrayList list = new ArrayList();    //legal
```

ray and list store Object references.