A+ Computer Science

LISTS

What is a List?

ArrayList

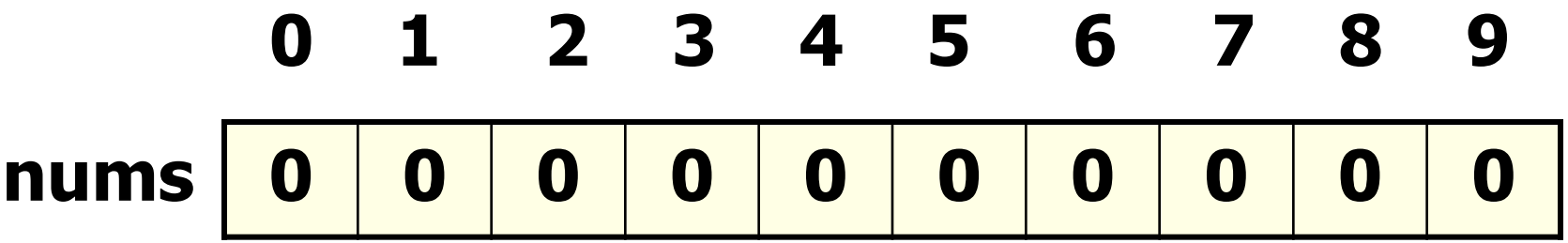
Arraylist is a class that houses an array.

An ArrayList can store any type.

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

ArrayList

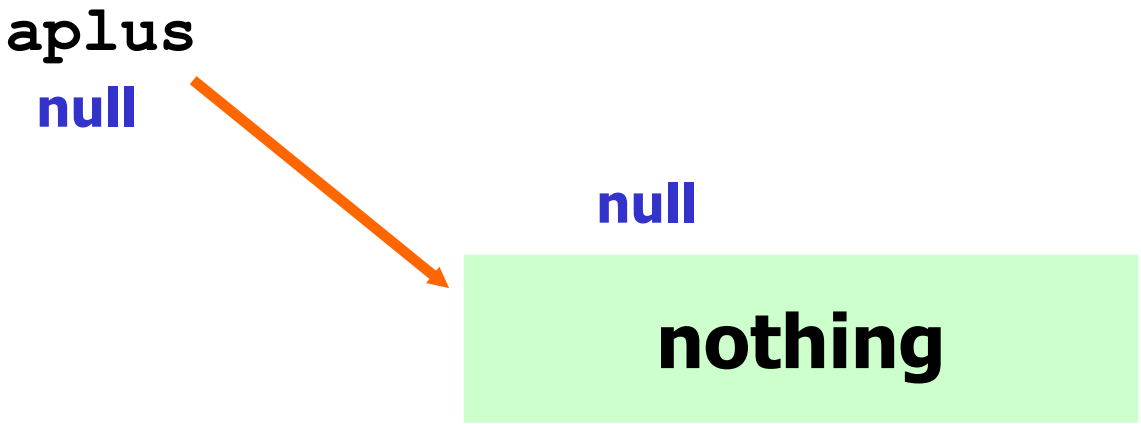
```
int[] nums = new int[10];        //Java int array
```



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

ArrayList

ArrayList aplus;



aplus is a reference to an ArrayList.

ArrayList

new ArrayList();

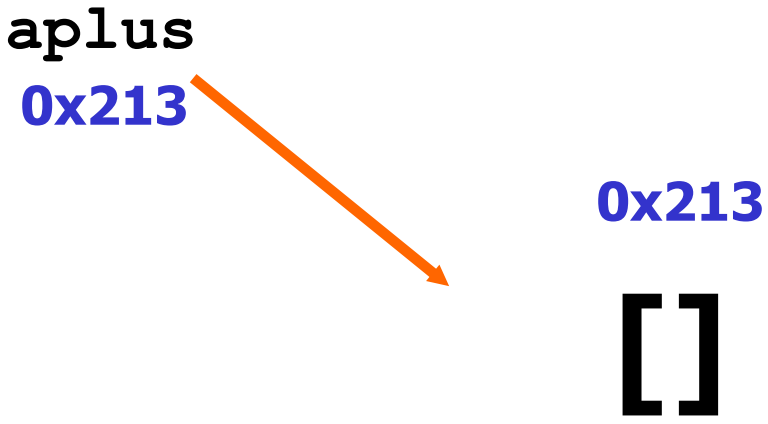
0x213

[]

ArrayLists are Objects.

ArrayList

```
ArrayList aplus = new ArrayList();
```



aplus is a reference to an ArrayList.

Generics

ArrayList

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

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

ArrayList

```
ArrayList<Long> bigStuff;
bigStuff = new ArrayList<Long>();

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

ArrayList

```
List<String> vals;
vals = new ArrayList<String>();
vals.add("aplus");
vals.add("compsci");
vals.add("contests");
out.println(vals.get(0).charAt(0));
out.println(vals.get(2).charAt(0));
```

OUTPUT

a

c

vals stores String references.

generics.java

ArrayList Methods

ArrayList

frequently used methods

Name	Use
add(item)	adds item to the end of the list
add(spot,item)	adds item at spot – shifts items up->
set(spot,item)	put item at spot z[spot]=item
get(spot)	returns the item at spot return z[spot]
size()	returns the # of items in the list
remove(int spot)	removes item at spot from the list

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

add( )

```
ArrayList<String> vals;
vals = new ArrayList<String>();
vals.add("aplus");
vals.add("rocks");
vals.add(0, "comp");
vals.add(1, "sci");
out.println(vals);
```

OUTPUT

[comp, sci, aplus, rocks]

add( )

```
ArrayList<Integer> nums;
nums = new ArrayList<Integer>();
nums.add(34);
nums.add(0,99);
nums.add(21);
nums.add(1,7);
nums.add(3);
nums.add(11);
out.println(nums);
```

OUTPUT

[99, 7, 34, 21, 3, 11]

addone.java
addtwo.java

set( )

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();
ray.add(23);
ray.add(11);
ray.set(1,5);
ray.add(4);
ray.set(0,7);
ray.add(53);
out.println(ray);
```

OUTPUT

[7, 5, 4, 53]

set( )

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();
ray.add(23);
ray.add(11);
ray.set(1,5);
ray.add(4);
ray.set(0,7);
ray.add(53);
out.println(ray.set(1,93) );
```

OUTPUT

5

set( )

```
List<Integer> ray;
ray = new ArrayList<Integer>();
ray.add(23);
ray.add(0, 11);
ray.set(5,66);
out.println(ray);
```

OUTPUT

Runtime exception

get( )

```
ArrayList<Integer> ray;
ray = new ArrayList<Integer>();
ray.add(23);
ray.add(11);
ray.add(12);
ray.add(65);

out.println(ray.get(0));
out.println(ray.get(1));
out.println(ray.get(2));
```

OUTPUT

```
23
11
12
```

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

get( )

```
List<Integer> ray;
ray = new ArrayList<Integer>();
ray.add(23);
ray.add(11);
ray.add(12);
ray.add(65);
```

OUTPUT

65

Runtime exception

```
out.println(ray.get(3));
out.println(ray.get(4));
```

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

set.java
get.java

Traversing Lists with Loops

Standard For Loop

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

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

Standard For Loop

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

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

OUTPUT

```
23
11
```

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

For Each Loop

```
for ( int item : bunchOfNums )
{
    out.println(  item );
}
```

The for each loop will access each reference in the list, starting with the first and ending with the last.

For Each Loop

```
for ( Integer  num  :  primeList  )
{
    out.println(  num );
}
```

The for each loop will access each reference in the list, starting with the first and ending with the last.

For Each Loop

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

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

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

OUTPUT

```
23
11
53
```

For Each Loop

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

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

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

OUTPUT

```
23
11
53
```

forloopone.java
foreachloopone.java

ArrayList Algorithms Counting

Counting List Values

In order to count the number of occurrences of a particular value, you must use a loop to access all items in the list.

You must also include an if statement to check for the specified value and a variable with which to count each of the variable's occurrences.

Counting List Values

```
loop through all list items

   if current item == search value

      increase the count by 1
```

Counting List Values

```
//assume nums is an list with values

int count = 0;
for( int  i = 0; i < nums.size(); i++ )
{
    if ( num.get( i )  matches value )
       count = count + 1;
}

//return or print count
```

Counting List Values

//assume nums is an list with values

```
int count = 0;
for( int  item  :  nums )
{
    if ( item matches value )
       count = count + 1;
}
```

//return or print count

listcount.java

ArrayList Algorithms Removal

remove( )

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

OUTPUT

[c, d]

remove( )

```
ArrayList<String> ray;
ray = new ArrayList<String>();
ray.add("a");
ray.add("b");
System.out.println( ray.remove(0) );
ray.add("c");
ray.add("d");
ray.remove(0);
System.out.println(ray);
```

OUTPUT

```
a
[c, d]
```

removeone.java

Removing Multiple Items

removing multiple values

```
spot = size – 1
while( spot is greater than
                    or equal to 0 )
{
  if ( this item is a match )
    remove this item from the list
  subtract 1 from spot
}
```

removing multiple values

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

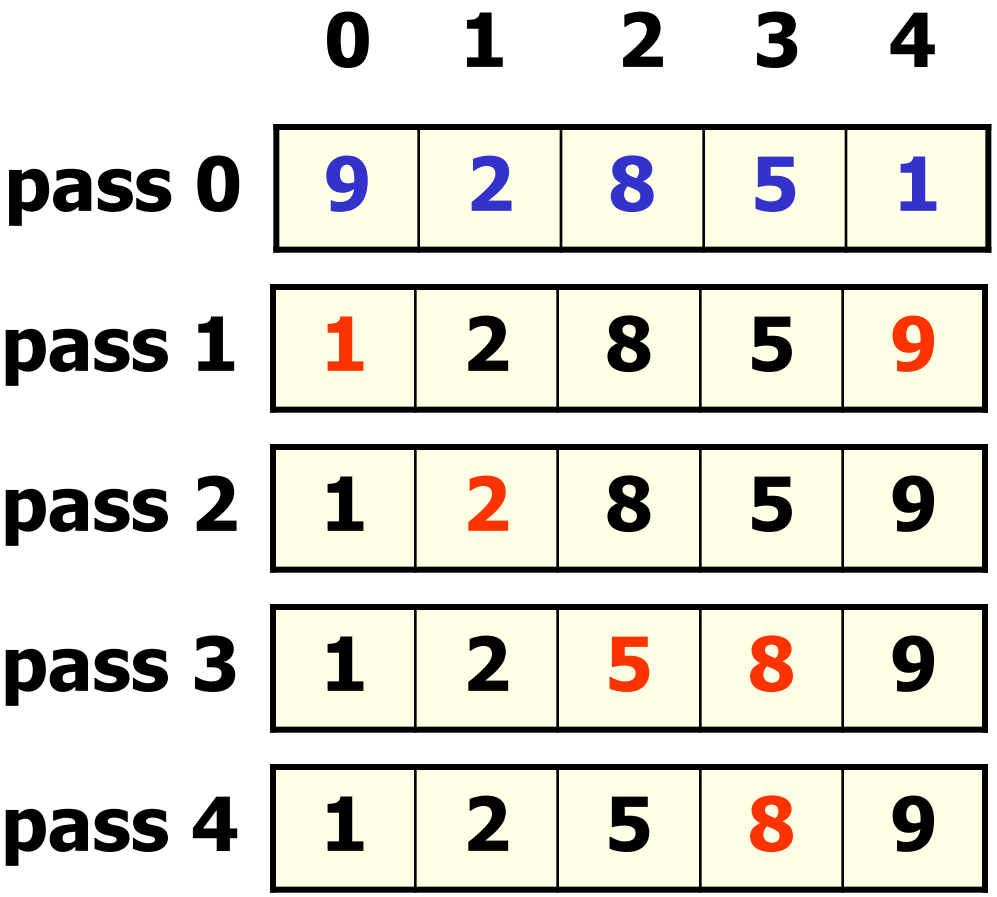
removeall.java

Sorting Algorithms

Sort Algorithms

```
void selectionSort( ArrayList<Integer> stuff )
{
    for(int i=0; i< stuff.size()-1; i++){
      int min = i;
      for(int j = i+1; j< stuff.size(); j++)
      {
        if( stuff.get(j) <  stuff.get(min) )
            min = j;      //find location of smallest
      }
      if( min != i) {
          int temp = stuff.get(min);
          stuff.set( min, stuff.get(i) );
          stuff.set( i, temp);    //put smallest in pos i
    }
  }
}
```

Sort Algorithms



Sort Algorithms

```
void insertionSort( ArrayList<Integer> stuff)
{
  for (int i=1; i< stuff.size(); ++i)
  {
    int val = stuff.get(i);
    int j=i;
    while( j>0&&val<stuff.get(j-1) ){
      stuff.set( j, stuff.get(j-1) );
      j--;
    }
    stuff.set(j,val);
  }
}
```

selectionsort.java
insertionsort.java

Just For Fun

clear( )

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

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

OUTPUT

[t, w]

clear.java

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);
out.println(ray);
out.println(Collections.binarySearch(ray,677));
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);
out.println(ray);
rotate(ray,2);
out.println(ray);
rotate(ray,2);
reverse(ray);
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);
out.println(ray);

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

OUTPUT

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

binarysearch.java
rotate.java
fill.java

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);

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

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

OUTPUT

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

search.java

Work on Programs!

Crank Some Code!

A+ Computer Science

LISTS