Sorting and Searching

Lab 22

Java
Sorts and Searches

Arrays
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
equals(x,y)	checks if x and y have the same values
fill(x, y)	fills all spots in x with value y

```
import  java.util.Arrays;
```


The `Arrays` class methods above are very useful methods for manipulating Java arrays.

`sort()` will naturally order the items in an array.

`binarySearch()` will find an item in the array and return the spot at which the item was found.

`equals()` will see if two arrays contain the exact same items in the exact same order.

`fill()` will fill in all spots in the array with a provided value.

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)	shifts items in x left or right
reverse(x)	reverses the order of the items in x

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


The `Collections` class methods above are very useful methods for manipulating Java Collections.

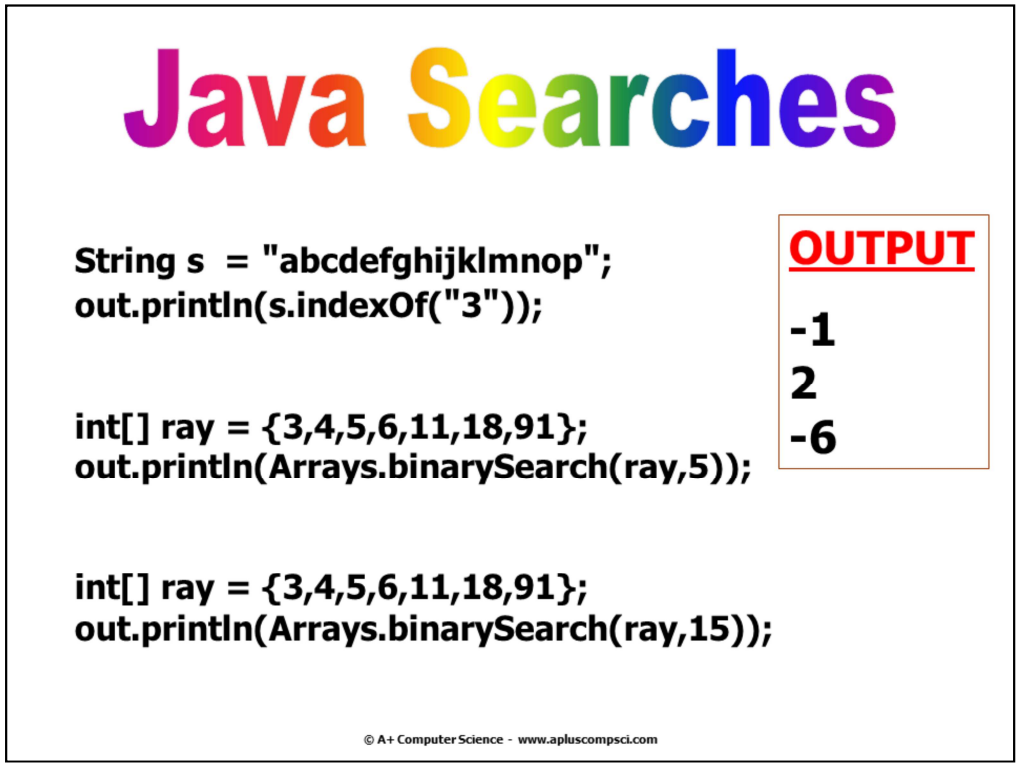
`sort()` will naturally order the items in the collection.

`binarySearch()` will find an item in the array and return the spot at which the item was found.

`fill()` will fill in all spots in the array with a provided value.

`rotate()` will shift items to the left(- negative x) a specified amount or shift items to the right(+ positive x) a specified amount.

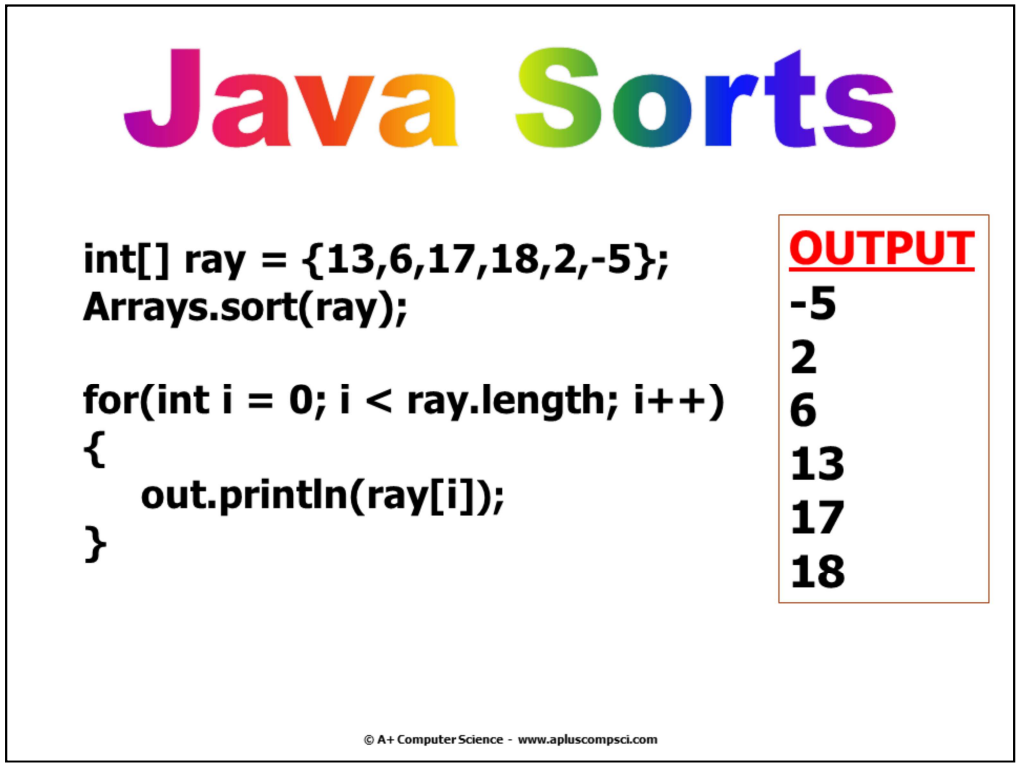
`reverse()` will reverse the order of all items.



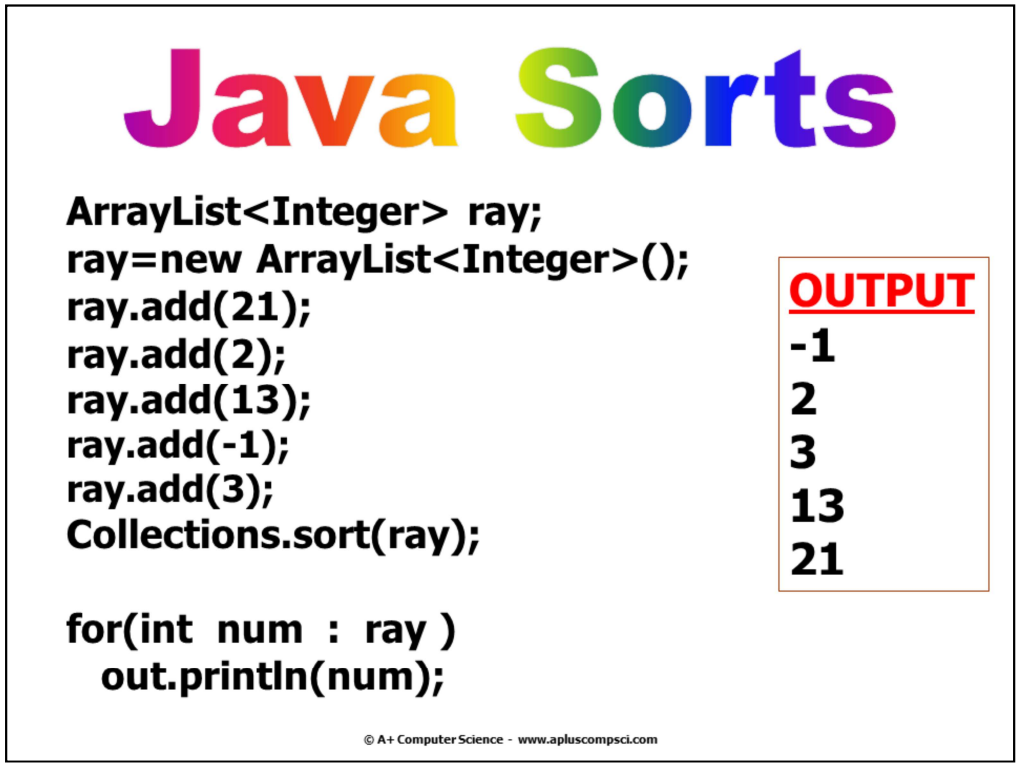
`indexOf()` and `binarySearch()` seach and array for specified value.

`indexOf()` will return the spot at which the item was found. It will return -1 if the item was not present.

`binarySearch()` will return the spot at which the item was found. It will return -1 + -location(where the item should be) if the item was not present.



`sort()` will naturally order the items in an array.



`sort()` will naturally order the items in the Collection.

Searching Algos

Linear Search

```
int linearSearch(int[]  list, int toFind)
{
  for(int i=0; i<list.length; i++)
  {
    if (list[i] == toFind)  //found it
          return i;
  }
  return -1;     //returns -1 if not found
}
```

indexOf( )

Linear / Sequential search will search an array or Collection for a specified value. The spot at which the item was found will be returned. If the item was not found, -1 is returned.

```
int binarySearch(int[] list, int toFind)
{
  int bot = 0, top = list.length-1;
  while(bot<=top)
  {
    int middle = (bot + top) / 2;
    if (list[middle] == toFind)
      return middle;
    else
      if (list[middle] > toFind)
        top = middle-1;
      else
        bot = middle+1;
  }
  return -1;
}
```

binarySearch

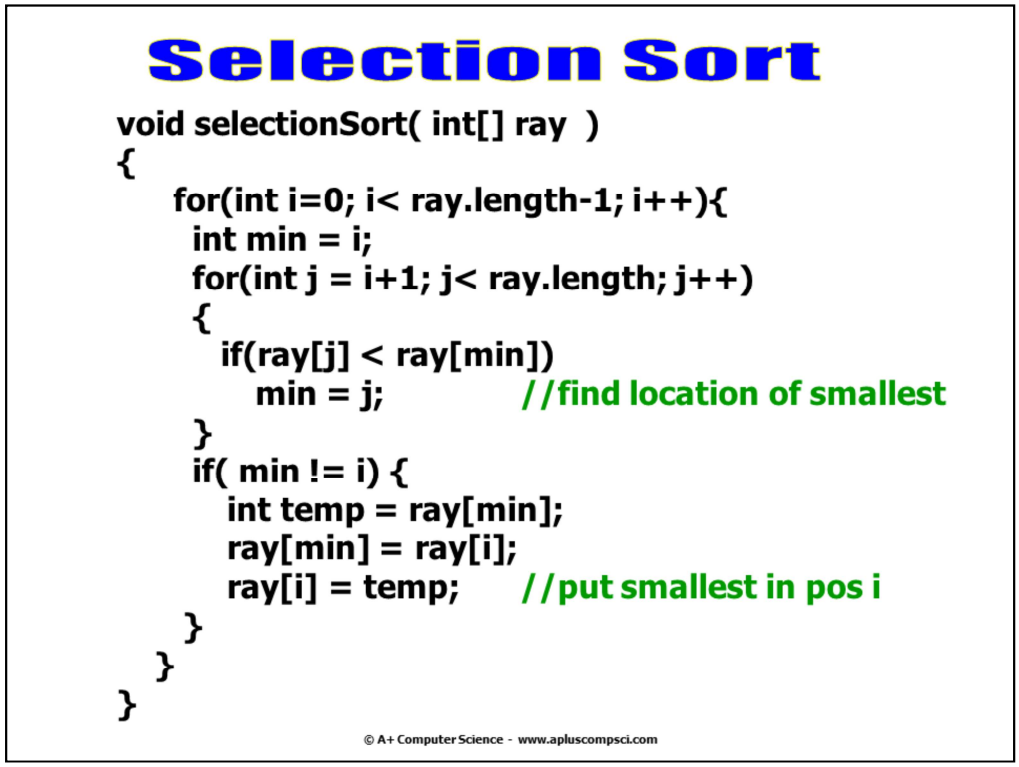
Binary search will search a sorted array or Collection for a specified value. The spot at which the item was found will be returned. If the item was not found, -1 is returned.

Binary search is much more efficient that a Linear / Sequential search, but the data must be sorted. Binary Search will spit the list in half each time it looks for a match. If the search value is not found at the current spot, the current spot is compared to the search value. If the search value is bigger than the item at the current spot, the top half of the list is split and checked. If the search value is less than the item at the current spot, the bottom half of the list is split and checked. This process continues until the search value is found or the bottom and top indexes cross.

open
linearsearch.java
linearsearchtester.java


People
Search
Demo
Line up some students and search for someone.
© A+ Computer Science - www.apluscompsci.com


Sorting
Algos
© A+ Computer Science - www.apluscompsci.com

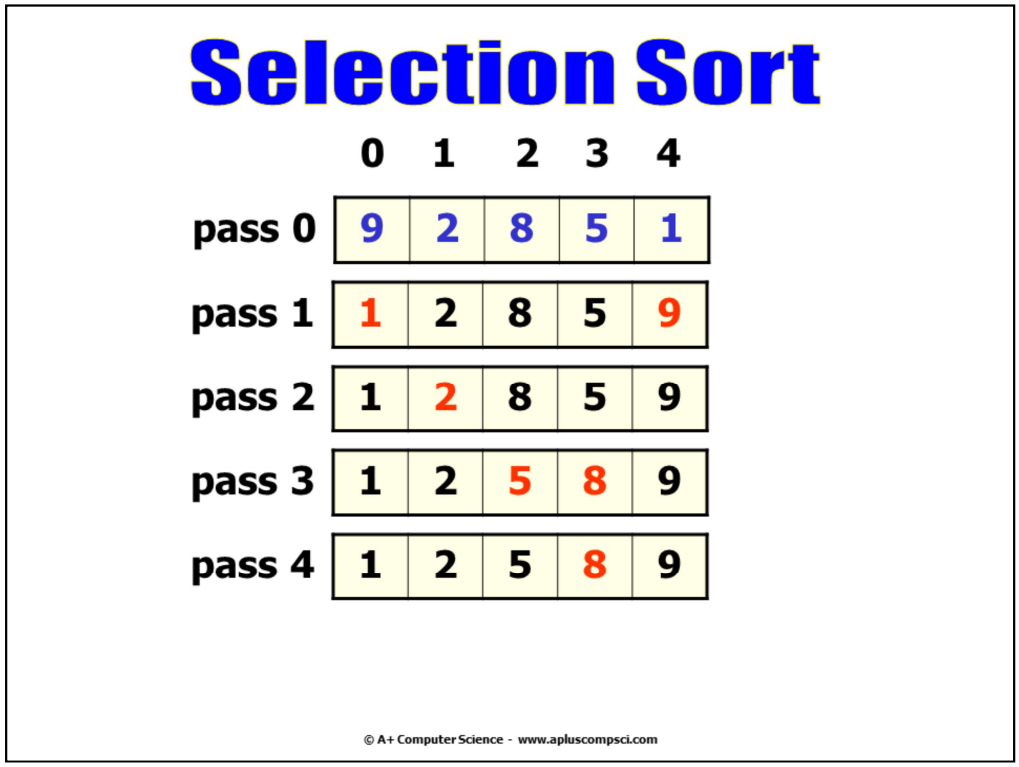


Selection sort is pretty effective for small lists, but pretty horrible is used on large lists.

Selection sort consists of two loops.

The outer loops run based on the number of items in the list.

The inner loop runs to find the items that need to be moved. The inner loop either locates the spot with the smallest value or the spot with the largest value. After the inner loop completes, a swap may occur if needed. At most, selection sort will make one swap per pass. A pass is one complete execution of the inner loop.



Selection sort is pretty effective for small lists, but pretty horrible is used on large lists.

Selection sort consists of two loops.

The outer loops run based on the number of items in the list.

The inner loop runs to find the items that need to be moved. The inner loop either locates the spot with the smallest value or the spot with the largest value. After the inner loop completes, a swap may occur if needed. At most, selection sort will make one swap per pass. A pass is one complete execution of the inner loop.

open
selectionsort.java
selectionsorttester.java


People
Sort
Demo
Line up some students and selection sort them.
© A+ Computer Science - www.apluscompsci.com

Lab 22
Help

Insertion Sort Logic

word = puppy

Original list
[one, two, xylaphone]

Original list after adding puppy
[one, puppy, two, xylaphone]

Removing an Item

```
void remove(String word )
{
  int loc = Collections.binarySearch(list, word);
  if(loc >=0)
  {
     list.remove(loc);
  }
}
```


`binarySeach()` works quick to locate items and to find the spot at which an item should be placed in the list..

`binarySeach()` will return the spot at which an item is located or -1 + -where the item should be if it were present.

Inserting an item into an ArrayList

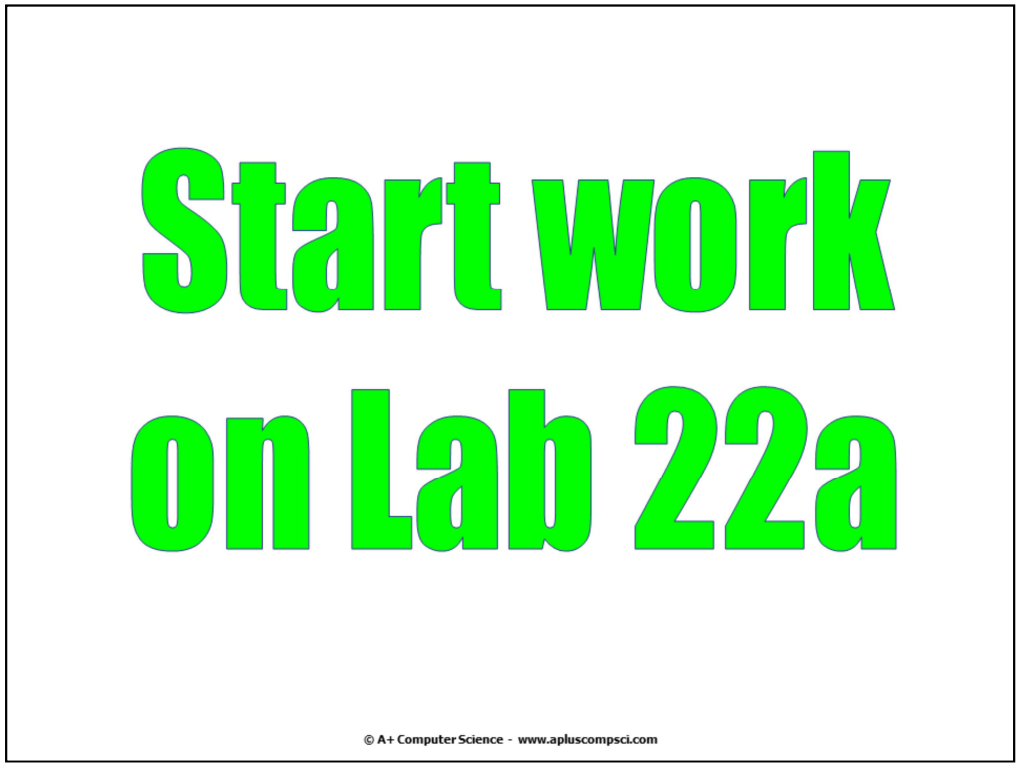
```
void add(String word)
{
  int loc = Collections.binarySearch(list, word);
  if(loc < 0)
  {
      loc = Math.abs(loc+1);  //??
      list.add(loc, word);
  }
}
```


`binarySeach()` works quite to locate items and to find the spot at which an item should be placed in the list..

`binarySeach()` will return the spot at which an item is located or -1 + -where the item should be if it were present.

open
insertionsort.java
insertionsorttester.java


People
Sort
Demo
Line up some students and insertion sort them.
© A+ Computer Science - www.apluscompsci.com


Start work
on Lab 22a
© A+ Computer Science - www.apluscompsci.com