AP Computer Science A Home Exam Reference Sheet Link to Quick Reference Guide

- The `Math` Class
    - Methods and Examples
        - `Math.abs`, `Math.pow`, and `Math.sqrt`
        - `Math.random`
    - Helpful Algorithms
        - Generating random integers
        - Divisibility
        - Place value for integers
        - Rounding doubles
- The `String` Class
    - Methods and Examples
        - Creating a `String`
        - `length`, `substring`, `equals`, `indexOf`, `equals`, and `compareTo` methods
    - Helpful Algorithms
        - Traversing a `String`
        - Reversing a `String`
        - Search a `String`
        - Count occurrences in a `String`
        - Replace letters in a `String`
- Arrays
    - Examples
        - Creating an array, size of an array
    - Helpful Algorithms
        - Traversing an array
        - Find max/min of an array
        - Average of an array
        - Search an array
        - Find the mode of an array
        - Sort an array (selection and insertion)
- `ArrayLists`
    - Methods and Examples
        - Creating an `ArrayList`
        - `add`, `size`, `get`, `set`, `remove` methods
        - Traversing an `ArrayList`
    - Helpful Algorithms
        - Find max/min of an `ArrayList`
        - Average of an `ArrayList`
        - Search an `ArrayList`
        - Find the mode of an `ArrayList`
        - Sort an `ArrayList` (selection)

The Math Class

Methods and Examples

Absolute Value

`static int abs(int x)` returns the absolute value of an `int` value

`static double abs(double x)` returns the value of a `double` value

```
//Calculates |3 - 2 + 8|

int y = Math.abs(3 - 2 + 8);
System.out.println(y);
```

Output:

```
9
```

```
//Calculates |-5.7|

double a = -5.7;
double b = Math.abs(a);
System.out.println(b);
```

Output:

```
5.7
```

Exponents

`static double pow(double base, double exponent)` returns the value of the first parameter raised to the power of the second parameter.

```
//Calculates (3.0)^2

double y;
y = Math.pow(3.0,2);
System.out.println(y);
```

Output:

```
9.0
```

```
//Calculates (x + 5)^9

double x, y;
x = 2;
y = Math.pow(x + 5, 9);
```

Output:

```
40353607.0
```

```
//Produces an error

int z = Math.pow(3,2);
System.out.println(z);
```

Output:

```
Error: Incompatible data
types double to int
```

Square Roots

`static double sqrt(double x)` returns the positive square root of a `double` value.

```
//Calculates √0.49

double a = 0.49;
double b = Math.sqrt(a);
```

Output:

```
0.7
```

```
//Calculates √(c + 10)

double c, d;
c = 90;
d = Math.sqrt(c + 10);
```

Output:

```
10.0
```

Random Number Generation

`static double random()` returns a double value greater than or equal to 0.0 and less than 1.0

```
//Produces a double in the range [0,9)
double a = Math.random() * 9;
//Produces an int in the range [0,9]
int b = (int)(Math.random() * 10);
```

```
//Produces a double in the range [4,9)
double c = Math.random() * 5 + 4;
//Produces an int in the range [4,9]
int d = (int) (Math.random() * 6) + 4;
```

```
//Produce a random int between [min and max]
int randInt = (int)(Math.random() * (max - min + 1)) + min;
```

The Math Class

Helpful Algorithms

Generate a random number from Minimun value to Maximum value	
Generate a random number from min to max	`int r = (int)(Math.random() * (max - min + 1)) + min;`

Divisibility	

Determine if a number is divisible by another number	`int num = 100`
	`int divisor = 5;`
	
	`boolean isDivisible;`
	
	`isDivisible = num % divisor == 0;`
	
	`-OR-`
	
	`if(num % divisor == 0)`
	`  isDivisible = true;`
	`else`
	`  isDivisible = false;`

Digits	

Identify the **n**th digit of an integer.	`int num = 5278;`
	
	`//ones place`
	`System.out.println(num / 1 % 10);`
	
	`//tens place`
	`System.out.println(num / 10 % 10);`
	
	`//100s place`
	`System.out.println(num / 100 % 10);`
	
	`//nth place`
	`System.out.println(num / (int)(Math.pow(10, n-1)) % 10);`

Stripping off the last digit one at a time.	`int num = 5278;`
	
	`if (num == 0)`
	`{`
	`  System.out.println(num);`
	`}`
	`else`
	`{`
	`  while (num > 0)`
	`  {`
	`  System.out.print(num % 10); //print last digit`
	`  num = num / 10; //strip off last digit`
	`  }`
	`}`

Rounding	

Round `double` to **n** decimal places	`double num = 12345.6531;`
	
	`//round to one decimal place`
	`num = (int)(num * 10 + .5);`
	`num = num / 10;`
	
	`//round to two decimal places`
	`num = (int)(num * 100 + .5);`
	`num = num / 100;`
	
	`//round to n decimal places`
	`num = (int)(num * Math.pow(10, n) + .5);`
	`num = num / Math.pow(10, n);`

The String Class

Methods and Examples

Creating a `String`

You can create a `String` object by using a `String` literal or the `String` constructor

```
//Using String literal

String x = "Gucci";
System.out.println("String is: " + x);
```

Output:

```
String is Gucci
```

```
//Using constructor

String y = new String("Team");
System.out.println("String is: " + y);
```

Output:

```
String is Team
```

`length` method

`int length()` returns the number of characters in a `String` object

```
String x = "Gucci";
int xLen = x.length();
System.out.print("Length is: " + xLen);
```

Output:

```
Length is 5
```

```
String y = new String("Team");
int yLen = y.length();
System.out.print("Length is: " + yLen);
```

Output:

```
Length is 4
```

`equals` method

`boolean equals(String other)` returns `true` if `this` equals `other`; returns `false` otherwise

```
String x = "bat";
String y = "man";
String z = "batman";

System.out.println(x.equals(y));
System.out.println(y.equals("man"));
System.out.println(z.equals(x+y));
//equals compares values of Strings
```

Output:

```
false
true
true
```

```
//Logic error can occur when using ==

String a = "Turtle";
String b = new String("Turtle");

System.out.println(a == b);

//Compares memory location, not value
```

Output:

```
false
```

substring Methods

int substring(int from, int to)
returns the substring beginning at index `from` and ending at index `to - 1`
0 <= from <= to <= length()

int substring(int from) returns `substring(from, length())`
returns the substring beginning at index `from` to the end of the `String`

```
/*

0 1 2 3 4 5
|G|u|c|c|i|

*/

String x = "Gucci";
System.out.println(x.substring(2, 4));
System.out.println(x.substring(0, 5));
System.out.println(x.substring(3, 5));
```

Output:

```
cc
Gucci
ci
```

```
/*

0 1 2 3 4
|T|e|a|m|

*/

String y = "Team";
System.out.print(y.substring(2));
System.out.print(y.substring(1));
System.out.print(y.substring(0));
```

Output:

```
am
eam
Team
```

indexOf Method

int indexOf(String str) returns the index of the first occurrence of a `String`, returns -1 if string not found.

```
/*

 0 1 2 3 4
|G|u|c|c|i|

*/

String x = "Gucci";
System.out.println(x.indexOf("c");
System.out.println(x.substring("ci");
System.out.println(x.substring("xx"));
```

Output:

```
2
3
-1
```

```
/*

 0 1 2 3
|T|e|a|m|

*/

String y = "Team";
System.out.print(y.indexOf("m"));
System.out.print(y.indexOf("T"));
System.out.print(y.indexOf("Team"));
```

Output:

```
3
0
0
```

`compareTo` method

`int compareTo(String other)` returns the alphabetic difference between two strings.

- If the result is `== 0`, then the two strings are **equal**;
- If the result is `< 0`, then `this` occurs **before** `other` alphabetically.
- If the result is `> 0`, then `this` occurs **after** `other` alphabetically

```
int x = "Tom".compareTo("Tim");

/* 'o' - 'i' = 111 - 105 = 6.
   Thus "Tim" is before "Tom" */
```

**Note – You do not need to know the value of each character or what the actual number will be when the `compareTo` method is called. You only need to know the relative order of the calling `String` and its parameter.

```
int y = "Tim".compareTo("Tom");

/* 'i' - 'o' = 105 - 111 = -6
   Thus "Tim" is before "Tom"  */
```

The String Class

Helpful Algorithms

Basic `String` Traversal

```
String myString = "Gucci";

for (int i = 0; i < myString.length(); i++){

     System.out.println(myString.substring(i, i + 1);
}
```

Output:

```
G
u
c
c
i
```

`String` Reversal - Using a reverse loop

```
String start = "Word";
String reverse = "";

for (int i = start.length(); i > 0); i--){

     reverse += start.substring(i - 1, i);
}

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

Output:

```
droW
```

`String` Reversal - Using a forward loop

```
String start = "Word";
String reverse = "";

for (int i = 0; i < start.length(); i++){

     reverse += start.substring(i, i + 1) + reverse;
}

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

Output:

```
droW
```

Search a String

```
String twist = "PeterPiperPicked";

boolean findE = false;

//Searches twist for the letter e

for(int i = 0; i < twist.length(); i++){

      if(twist.substring(i, i + 1).equals("e")){

            foundE = true;
      }
}

System.out.println("Contains e: " + foundE);
```

Output:

```
Contains e: true
```

Count Occurrences in String

```
String lyric = "PalmsAreSweatyKneesWeakArmsAreHeavy";

int countARE = 0;

//Counts the number of times "Are" appears in the String lyric
//Note:  The loop below shows loop balancing

for (int i = 0; i < lyric.length() - 2; i++){

      if (lyric.substring(i, i + 3).equals("Are")){

            countARE++;
      }
}

System.out.println("The word 'Are' appears " + countARE + " times.");
```

Output:

```
The word 'Are' appears 2 times.
```

Replace Letters in `String`

```
//This code replaces the "e" with the "3"

String start = "I_Am_Leet_Gamer";

String makeLeet = "";


for (int i = 0; i < start.length(); i++){

      String letter = start.substring(i, i + 1);

      if(letter.equals("e")){

            makeLeet += "3";
      }
      else{
            makeLeet += letter;
      }
}

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

Output:

```
I_Am_L33t_Gam3r
```

Remove all occurrences of a substring in a `String`

```
//This code removes all occurrences of "an" in "Panama Canal"

String original = "Panama Canal";
String toFind = "an";

String temp = original;


int index = temp.indexOf(toFind);


while (index != -1){

      temp = temp.substring(0, index) + temp.substring(index + toFind.length());

      index = temp.indexOf(toFind);

}

System.out.println(original + " with all occurrences of " + toFind +
                                " removed is " + temp);
```

Output:

```
Pama Cal
```

Arrays

(The size of an array cannot be changed)

Methods and Examples

Creating an Array

Arrays must have a specified size and data type.

- **When an array is created by calling a constructor, each element is initialized to its "Zero" value.**
  - **Objects initialize to `null`; `int` and `double` initialize to 0; `boolean` initializes to `false`**
- **An array can be created using an initializer list. Must be done when declaring the array.**
  `int[] numbers = {1, 2, 3, 4, 5};`

```
/*
  Creating an Empty Array of ints with
  size 5
*/

int[] myArray = new int[5];


myArray[0] = 5;
myArray[3] = 2;
```

Array Created:

0	1	2	3	4
5	0	0	2	0

```
/*
  Creating an initialized array of
  Strings
*/

String[] myArray = {"Bill", "Paxton"};
```

Array Created:

0	1
"Bill"	"Paxton"

Length of an Array

To access the size of the array, use its `length` property.
**Note: `length` is NOT a method – it is a constant of the class

```
/*
  Creating an Empty Array of ints
  with size 5
*/

int[] myArray = new int[5];

System.out.println(myArray.length);
```

Output:

```
5
```

```
/*
  Creating an initialized array
  of Strings
*/

String[] myArray = {"Bill", "Paxton"};

System.out.println(myArray.length);
```

Output

```
2
```

Arrays

(The size of an array cannot be changed)

Helpful Algorithms

Traversing an Array

You can access each element of an array by using a for-loop or an enhanced for loop.

Basic For Loop

```
String[] band =
{"John","Paul","George","Ringo"};

for(int i = 0; i < band.length; i++){

      System.out.println(band[i]);
}

for(int j = band.length - 1; j >= 0; j --)
{
      System.out.print(band[j]);
}
```

Output:

```
John
Paul
George
Ringo
RingoGeorgePaulJohn
```

Enhanced For Loop

```
String[] ratPack =
{"Sinatra", "Martin, "Davis"};
for(String artist: ratPack){
      System.out.println(artist);
}
```

Output:

```
Sinatra
Martin
Davis
```

Find Max and Min in Array

Basic For Loop

```
int[] nums = {3, 2, 6, 8, 100, 4, 1};

int min = nums[0];
int max = nums[0];

for(int i = 0; i < nums.length; i++){

      if(nums[i] > max)
            max = nums[i];

      if(nums[i] < min)
            min = nums[i];
}

System.out.println("Min is: " + min);
System.out.println("Max is: " + max);
```

Output:

```
Min is: 1
Max is: 100
```

Enhanced For Loop

```
int[] nums = {3, 2, 6, 8, 100, 4, 1};

int min = nums[0];
int max = nums[0];

for(int value : nums){

      if(value > max)
            max = value;

      if(value < min)
            min = value;
}

System.out.println("Min is: " + min);
System.out.println("Max is: " + max);
```

Output:

```
Min is: 1
Max is: 100
```

Find Average in Array

Basic For Loop

```
int[] nums = {1, 2, 2, 3, 3, 4, 4, 5};

int sum = 0;

for(int i = 0; i < nums.length; i++){

      sum += nums[i];
}

double average;
average = (double) sum / nums.length;

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

Output:

```
3
```

Enhanced For Loop

```
int[] nums = {1, 2, 2, 3, 3, 4, 4, 5};

int sum = 0;

for(int value : nums){

      sum += value;
}

double average;
average = 1.0 * sum / nums.length;

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

Output:

```
3
```

Searching an Array

```
int[] nums = {1, 2, 2, 3, 3, 3, 4, 4, 5}; //initializer list

int target = 4;
boolean found = false;
```

Basic For Loop

```
for (int i = 0; i < nums.length; i++){
    if (nums[i] == target)
        found = true;
}
```

Enhanced For Loop

```
for (int value : nums){
    if (value == target)
        found = true;
}
```

```
System.out.println("It is " + found + " the target: " + target +
                                    " is in the array");
```

Output:

```
It is true the target: 4 is in the array
```

Find Mode in Array

Basic For Loop

```
int[] nums = {1, 2, 2, 3, 3, 3, 4, 4, 5};

int currMaxCount = 0;
int mode = nums[0];

for(int i = 0; i < nums.length; i++){

      int count = 1;
      int currNum = nums[i];

      for(int j = i + 1; j < nums.length; j++){

            if( currNum == nums[j])
                  count++;
      }

      if(count > currMaxCount){
            currMaxCount = count;
            mode = currNum;
      }
}

System.out.println("The Mode is: " + mode);
```

Output:

```
The Mode is: 3
```

Selection Sort for an Array

Sorts from lowest to highest

```
/** Sorts a given array of ints into ascending (non-decreasing) order */
public static void selection(int[] arr)
{
   for (int i = 0; i < arr.length - 1; i++) // Outer loop (passes)
   {
      int min = i; // index of initial minimum of "remaining"
      for (int k = i + 1; k < arr.length; k++) // Loop through "remaining".
      {
         if (arr[k] < arr[min])
         {
            min = k;
         }
      }

      // Swap minimum of "remaining" into the correct position.
      int temp = arr[i];
      arr[i] = arr[min];
      arr[min] = temp;
   }
}
```

Insertion Sort for an Array

Sorts from lowest to highest

```
// note: we start with 1 instead of 0
/** Sorts a given array of ints into ascending (non-decreasing) order */
public static void insertion(int[] arr)
{
   for (int i = 1; i < arr.length; i++) // Outer loop (passes)
   {
      int temp = arr[i];   // Move data element to temp.
      int k = i;           // Set k to current position of "hole".

      while (k > 0 && temp < arr[k - 1]) // Loop until "hole" is in correct place.
      {
         arr[k] = arr[k - 1];   // Move next data element to "hole".
         k--;                   // Set k to new position of "hole".
      }

      arr[k] = temp;            // Move data element to "hole".
   }
}
```

Array Lists:

(The length of an array list can change)

Methods and Examples

Creating an `ArrayList`

You can create an `ArrayList<E> = new ArrayList<E>` , where `E` represents a reference data type.

```
//Creating an ArrayList of String


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

```
//Creating an ArrayList of Integer
//Note Integer must be used instead of
//int

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

add methods

`boolean add(E obj)` appends `obj` to the end of the list; returns `true`.

`void add(int index, E obj)` inserts `obj` at position `index (0 <= index <= size)` moving elements at position `index` and higher to the right (adds `1` to their indices) and adds `1` to `size`.

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


//Fills the ArrayList with the numbers 110 - 118
for (int i = 110; i < 119; i++){

      integerList.add(i);
}
integerList.add(3, 200);
```

integerList after the for loop has completed

```
  0   1   2   3   4   5   6   7   8
|110|111|112|113|114|115|116|117|118|
```

integerList after 200 has been inserted at index 3

```
  0   1   2   3   4   5   6   7   8   9
|110|111|112|200|113|114|115|116|117|118|
```

size method

int size() returns the number of elements in the list.

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

System.out.print(integerList.size() + "\t");

//Fills the ArrayList with the numbers 0 - 9
for (int i = 0; i < 10; i++){

      integerList.add(i);
}

System.out.print(integerList.size());
```

Output:

```
0     10
```

get method

E get(int index) returns the element at position index in the list.
0 <= index < size()

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

beatles.add("George");
beatles.add("Paul");
beatles.add("Ringo");
beatles.add("John");

System.out.println(beatles.get(1));
System.out.println(beatles.get(3));
```

List:

```
    0       1       2      3
|"George"|"Paul"|"Ringo"|"John"|
```

Output:

```
Paul
John
```

set method

E set(int index, E obj) replaces the element at position `index` with `obj`; returns the element formerly at position `index`

0 <= index < size()

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

beatles.add("George");
beatles.add("Paul");
beatles.add("Ringo");
beatles.add("John");

String newMember = "Yoko";

beatles.set(0, "George Harrison");

System.out.println("The member at index 0 is: " + beatles.get(0));

System.out.println("The member at index 3 was: " + beatles.set(3, newMember));

System.out.println("But now it is: " + beatles.get(3));
```

List:

0	1	2	3
"George Harrison"	"Paul"	"Ringo"	"Yoko"

Output:

```
The member at index 0 is:  George Harrison
The member at index 3 was:  John
But now it is:  Yoko
```

remove method

E remove(int index) removes element from position index, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from `size`; returns the element formerly at position `index`

0 <= index < size()

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

beatles.add("George");
beatles.add("Paul");
beatles.add("Pete");
beatles.add("John");

beatles.remove(2);
beatles.add("Ringo");

for (int i = 0; i < beatles.size(); i++){

      System.out.print(beatles.get(i) + " ");
}
```

Output:

```
George Paul John Ringo
```

ArrayList:

(The length of an array list can change)

- Methods and Examples
- Helpful Algorithms
- Home

Helpful Algorithms

Traversing an ArrayList

```
ArrayList<String> languages = new ArrayList<String>();
languages.add("Spanish");
languages.add("French");
languages.add("English");
```

Basic for Loop

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

Enhanced for Loop

```
for (String word : languages){
   System.out.println(word);
}
```

Output:

```
Spanish
French
English
```

Find Max and Min in ArrayList

Basic for Loop

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

/*Assume nums is initialized with

  a max of 100 and min of 1*/

int min = nums.get(0);
int max = nums.get(0);

for (int i = 0; i < nums.size(); i++){

      if(nums.get(i) > max)
            max = nums.get(i);

      if(nums.get(i) < min)
            min = nums.get(i);
}

System.out.println("Min is: " + min);
System.out.println("Max is: " + max);
```

Output:

```
Min is: 1
Max is: 100
```

Enhanced for Loop

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

/*Assume nums is initialized with

  a max of 100 and min of 1*/

int min = nums.get(0);
int max = nums.get(0);

for (Integer value : nums){

      if(value > max)
            max = num;

      if(value < min)
            min = num;
}

System.out.println("Min is: " + min);
System.out.println("Max is: " + max);
```

Output:

```
Min is: 1
Max is: 100
```

Find Average of the ArrayList

Basic for Loop

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

/*Assume nums is initialized*/

int sum = 0;

for (int i = 0; i < nums.size(); i++){

      sum += nums.get(i);
      //nums.get(i) is unboxed to an int
}

double average;
average = (double) sum / nums.size();

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

Enhanced for Loop

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

/*Assume nums is initialized*/

int sum = 0;

for (Integer value : nums)

      sum += value;
      //value is unboxed to an int
}

double average;
average = 1.0 * sum / nums.size();

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

Find Mode in ArrayList

Basic for Loop

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

/*Assume nums is initialized*/

int currMaxCount = 0;
int mode = nums.get(0);

for (int i = 0; i < nums.size(); i++){

      int count = 1;
      int currNum = nums.get(i);

      for(int j = i + 1; j < nums.size(); j++){

         if( currNum == nums.get(j)) //nums.get(i) is unboxed to an int
                count++;
      }

      if(count > currMaxCount){
            currMaxCount = count;
            mode = currNum;
      }
}

System.out.println("The Mode is: " + mode);
```

Searching an ArrayList

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

/*Assume nums is initialized*/

int target = 4;
boolean found = false;
```

Basic for Loop

```
for (int i = 0; i < nums.size(); i++){
   if (nums.get(i) == target)
      found = true;
}
```

Enhanced for Loop

```
for (Integer value : nums){
   if (value == target)
      found = true;
}
```

Count occurrences in an ArrayList

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

/*Assume nums is initialized*/

int target = 4;
int count = 0;
```

Basic for Loop

```
for (int i = 0; i < nums.size(); i++){
     if (nums.get(i) == target)
         count++;
}
```

Enhanced for Loop

```
for (int value : nums){
     if (value == target)
         count++;
}
```

Removing all occurrences of a value from an ArrayList

```
public static void removeTarget(ArrayList<String> words, String target)
{
```

for Loop

```
  for (int i = words.size() - 1; i >= 0; i--)
  {
    String item = words.get(i);
    if (item.equals(target))
       words.remove(i);
  }
}
/* Remember to go backwards to avoid
   skipper code!
*/
```

while Loop

```
int i = 0;
while (i < words.size())
{
   String item = words.get(i);
   if (item.equals(target))
      words.remove(i);
   else
      i++; //only advance if no remove
}
```

Remember, if you need to remove all or multiple occurrences of a value from an ArrayList, going forward could cause you to skip some items. When the `remove(int index)` ArrayList method is called, it removes the item, shifts all the values at a higher index to down to a lower index (one less than their current index).

Selection Sort for an ArrayList

Sorts from lowest to highest

```
/** Sorts a given ArrayList of String into ascending (non-decreasing) order */

public static void selection(ArrayList<String> arr)
{
   for (int i = 0; i < arr.size() - 1; i++) // Outer loop (passes)
   {
      int min = i; // index of initial minimum of "remaining"
      for (int k = i + 1; k < arr.size(); k++) // Loop through "remaining".
      {
         if (arr.get(k).compareTo(arr.get(min)) < 0)
         {
            min = k;
         }
      }

      // Swap minimum of "remaining" into the correct position.
      String temp = arr.get(i);
      arr.set(i, arr.get(min));
      arr.set(min, temp);
   }
}
```

Inserting an new item into an already sorted ArrayList

```
//for loop version

/** Inserts one Integer value into an ArrayList that is sorted into ascending

(non-decreasing) order */

public static void insertOne(ArrayList<Integer> arr, Integer val)
{
   for (int i = 0; i < arr.size(); i++)
   {
       if (val.compareTo(arr.get(i)) < 0){
          arr.add(i, val);
          return; //you are done - leave the method.
                  //this type of return DOES NOT return a value - void method
   }
   arr.add(val); //if the value was never inserted, it is the largest-add to end
}
```

```
//while loop version


/** Inserts one Integer value into an ArrayList that is sorted into ascending

(non-decreasing) order */


public static void insertOne(ArrayList<Integer> arr, Integer val)
{
   int i = 0;

   while (i < arr.size() && val.compareTo(arr.get(i)) > 0)
   {
      i++;  //as long val is greater than arr.get(i), keep looking.
            //When val is less than or equal to arr.get(i), insert at i
   }
   arr.add(i, val);
}
```