CS4A Lab #5 Arrays Professor Tom DeDonno   
Your Name: Edgar Ontiveros  
  
1) Array Sizes

double[][] m = { {1, 4, -7}, {2, -1, 4}, {0, -9,18 } };  
double[][] n = { {2,3}, {3, 4, 5}, {0,1, 2} , {1} };

How many rows does m have? 3  
How many rows does n have? 4

What does n[ (int)m[0][0] ].length equal? 3 – Expression ends up being n[1].length which counts the number of columns in row 1.

Which matrix occupies more space on the heap and why?

N taks more space in the heap because it has an extra row which means an extra reference has to be stored.

2) Where is the method arraycopy located?

Java.lang.system

3) Does the following code resize the array (explain)?

int[] myList; myList = new int[10];

// Sometime later you want to assign a new array to myList

myList = new int[20];

No, it will not resize the array. A new array will be made of size 20 and will scrap the old array of size 10.

4) True or false? When an array is passed to a method, a new array is created and passed to the method.

False, a reference to the original array will be passed.

5) What sorting algorithms does Arrays.sort uses and why?

Quicksort is faster and more efficient with memory but is it not stable (if you sort an already sorted array, it may not stay unchanged), which is where merge sort comes into play.

5) Use Figure 7.9 (or binary search method discussed in class) as an example to show how to apply the binary search approach to a search for key 10 and key 12 in list {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79}.

Key 10:

Start -- 2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79 ((0+12)/2 = 6)

(mid=50,50>10)— 2,4,7,10,11,45 ((0+5)/2=2)

(mid=7,7<10) – 10,11,45 ((3+5)/2=4)

(mid=11,11>10) – 10

Key 12:

Start -- 2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79 ((0+12)/2 = 6)

(mid=50, 50<12) – 2,4,7,10,11,45 ((0+5)/2=2)

(mid = 7,7<12) – 10,11,45 ((3+5)/2=4)

(mid=11, 11<12) – 45 low and high are both the same number, 45 is not = 12 so the key was not found.

6) Use Figure 7.11 as an example to show how to apply the selection-sort approach to sort {3.4, 5, 3, 3.5, 2.2, 1.9, 2}.

public class Test {

public static void selectionSort1(double[] x) {

for (int i=0; i<x.length-1; i++) {

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

if (x[i] > x[j]) {

//... Exchange elements

double temp = x[i];

x[i] = x[j];

x[j] = temp;

}

}

}

for(int k = 0; k<x.length;k++){

System.out.print(x[k] + “ “);

}

}

public static void main(String[] args) {

double arr[] = {3.4, 5, 3, 3.5, 2.2, 1.9, 2};

selectionSort1(arr);

}

}

Results are: 1.9 2.0 2.2 3.0 3.4 3.5 5.0