Assignment 3.2

Eric Wallace

Eric Wallace

Exercise7.1- 10am, 2pm, 6pm, and 5pm are the busiest.

Exercise7.2- private Person [ ] people;

Exercise7.3- private boolean [ ] vacant;

Exercise7.4- It's used in four places.

Exercise7.5- The brackets were placed before instead of after the type declaration.

int[ ] counts;

boolean[5000] occupied;

Exercise7.6-

double[ ] readings = new double[60];

String[ ] urls = new String[90];

TicketMachine[ ] machines = new TicketMachine[5];

Exercise7.7- None are created just the allowed amount of strings in the array is set.

Exercise7.8- It is using parentheses instead of brackets.

double[ ] prices = new double[50];

Exercise7.9- It throws an out of bounds exception.

Exercise7.10- Did it, works.

Exercise7.11-

/\*\*

\* Print all the values in th marks array that are

\* greater than mean.

\* @param marks An array of mark values

\* @param mean The mean (average) mark.

\* /

public void printGreater(double[ ] marks, double mean)

{

for (int index = 0; index < marks.length; index++) {

if(marks[index] > mean) {

System.out.println(marks[index]):

}

}

}

Exercise7.13-

/\*\*

\* Return the number of accesses recorded in the log file.

\*

\*/

public int numberOfAccesses()

{

int total = 0;

// Add the value in each element of hourCounts to // total.

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

total += hourCounts[i];

}

return total;

}

Exercise7.14- Did it.

Exercise7.15- It needs to check every element. The for loop was best to keep track of the index.

Exercise7.16- Did it.

Exercise7.17- Only the first one.

Exercise7.18- Did it.

Exercise7.19- Defined the accessors for year month and day, add the arrays to hold daily count and weekly count, analyzed the data for weekly patterns, added methods for the busiest and quietest days in a 7-day cycle.

Exercise7.20- Did it.

Exercise7.21- Did it.

Exercise7.22- Using the flexible-size collection is better so we can add students and resize the collection as needed.

Exercise7.23-

public void listAllFiles()

{

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

System.out.println(files.get(i));

}

}

Exercise7.25- Yes same pattern.

Exercise7.26- Two versions. One sets all elements of the array the other only sets the defined range of elements. The reset takes all elements to 0.

Exercise7.27- Different pattern this time. They are deterministic.

Exercise7.28- Did it.

Exercise7.29- The process relies on not changing the previous state while we fill the next state.

Exercise7.31- Did it.

Exercise7.32- Done, using version2

Exercise7.33- There is an infinite number of possibilities.

Exercise7.34- Did it.

Exercise7.35- Did it.

Exercise7.37- Did it.

Exercise7.38- Did it.

Exercise7.39- Dit it.

Exercise7.41-

int[ ] copy = new int[original.length];

System.arraycopy(original, 0, copy, 0, original.length);

Exercise7.42- The asList method converts an array to a list. The binarySearch method performs a binary search of the int array elements. The fill method fills the int array with a given integer. Sort method sorts the elements of an int array in ascending order.

Exercise7.44-

int[ ] [ ] original = {{2573, 3, 95}, {2, 20, 7}}; int[ ] [ ] copy = new int[2][];

System.arraycopy(original, 0, copy, 0, original.length);

System.out.prinf("The copy array is now: ");

for (int i = 0; i < 2; i++) {

for(int j = 0 ; j < 3; j++) {

System.out.print(copy[i][j] + " ");

}

System.out.println();

}