Chapter 7 Book work

Logan Yeubanks

7.1: Hour 10 and hour 18 are the busiest times of the day.

7.2: Person[] people;

7.3: boolean[] vacant;

7.5: []int counts; is wrong because the square brackets were put before the int. The data type of the variable must come before the square brackets.

boolean[5000] occupied; is wrong because the size of the array does not occur while initializing but later when a new declaration is made.

7.6:

double readings = new double[60];

String[] urls = new String[90];

TicketMachine[] machines = new TicketMachines[5];

7.7: A array capable of holding 20 String objects is made but nothing is yet stored into it.

7.8: (50) should be [50]. Square brackets are used in array creation instead of parenthesis.

7.9: A out of bounds error occurs because 24 is not less than 24.

7.10:

while(hour < hourCounts.length)

{

System.out.println(hour + ":" + hourCounts[hour]);

hour++;

}

7.11:

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

{

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

{

If(marks[index] > mean)

{

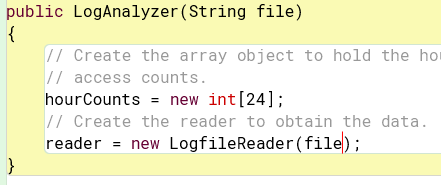
System.out.println(marks[index]);

{

}

}

7.12:



7.13:

public int numberOfAccesses()

{

Int total = 0;

For(int hour = 0; hour < hourCounts.length; hour++)

{

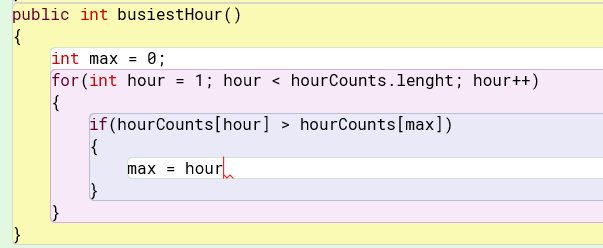
Total += hourCounts[hour];

}

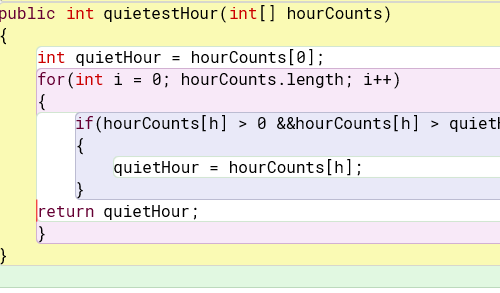
return total;

}

7.15:

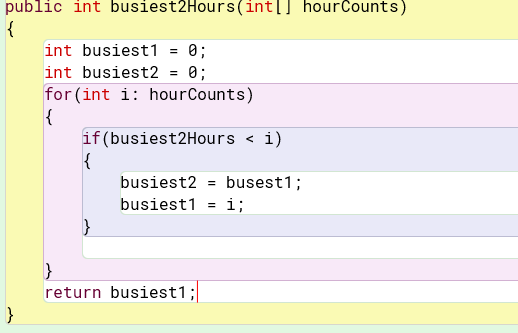


7.16:



7.17: It returns the largest amount that was last.

7.18:



7.22: I think it makes more sense to have flexible sized arrays because it will grow as needed and takes less time. I fixed sized array would be the opposite so it just comes down to need.

7.23:

public void listAllFiles()

{

iterator I = files.iterator();

for(I = 0; i.hasNext(); i++)

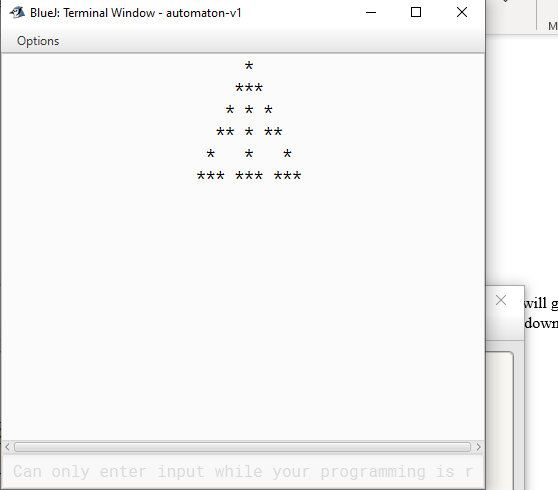
{

System.out.println(i.next());

}

}

7.24:



7.25: They are the same shape but appear in different positions.

7.26: The fill method is used to initialize the contents of the array.

7.27: It shows that the shape depends on the contents of the fill() method and whatever is initialized in it.

7.29: The shape changed due to the array being updated after iterating through the array. Now it updates after every iteration through the loop.

7.30: It would seem like no values need to be retained because they are being updated throughout the loops execution.

7.31:

For(int I = 0; I < state.lenght; i++)

{

Int right = I + 1;

State[i+1] : 0;

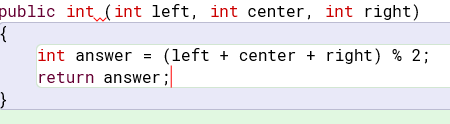
nextState[i] = (left + center + right) % 2;

left = center;

center = right;

}

7.32:



7.33: There is likely infinite possibilities because there are so many different combinations.

7.42:

As list returns a list of the inserted array.

binarySearch searches for a key in a array using binary

fill assigns something to each element of the array

sort puts the elements of a array into order from least to greatest.

7.44:

int[] copy = new int[500];

for( it = 0; it < 500; it++)

{

for(it = 0; it < 500; it++)

{

Copy[row][column] = original[row][column];

}

}