CS 1401, Exam #3, MW 9-10:20 version

48/50 = 96/100

Date: Wednesday, November 6, 2013
Name (please type legibly, ideally in block letters): Danielle Willis

---

On November 6, 1893, Pyotr Ilyich Tchaikovsky, a well-known Russian composer, died during the cholera epidemic. He died a few days after the premiere of his Sixth Symphony "Pathetique", a piece full of suffering and pain. In the US, Tchaikovsky is most known for his music to the Nutcracker ballet -- usually performed at Christmas time, and for his 1812 Overture, often performed on July 4.

1a. Many of his musical pieces -- e.g., all his symphonies -- consist of several parts (called "movements"). Write a method named *average* that, given the total duration *d* of a piece and the number of parts *p* in this piece, returns the average duration of a movement.

10/10

1b. The Sixth Symphony lasts 46 minutes and consists of 4 movements. Call (invoke) your method *average* in the *main* method to compute the average duration of each movement. You do not need to write the entire *main* method, just the part that assigns values to the corresponding variables *duration* and *parts* and calls your method.

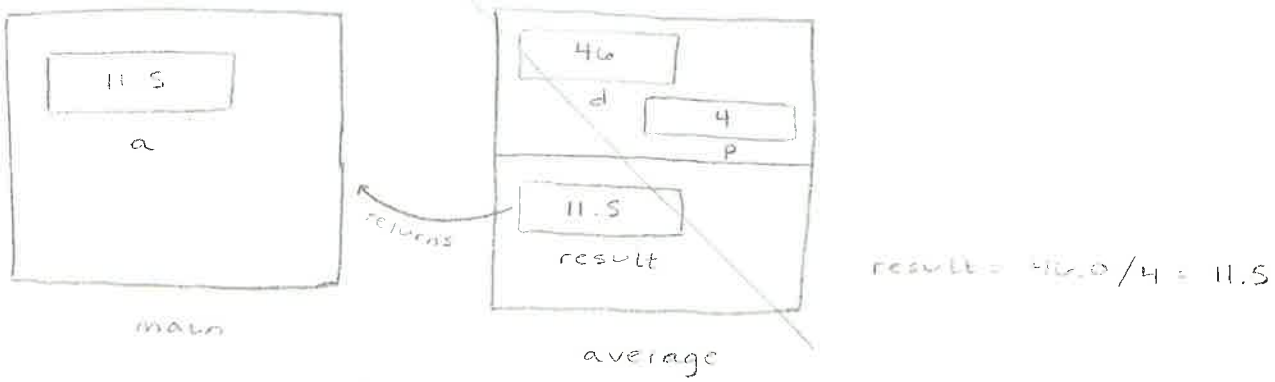
1c. Trace, step by step, how the computer will perform the needed computations, and check that the result is indeed correct.

a.)
```
public static double average(int d, int p) {
    double result = ((double)d / p);
    return result; }
```

b.) In the main method:

```
double a = average(46, 4);
```

c.) Trace:



result = 46.0 / 4 = 11.5

2a. Define a class *Symphony* whose objects are different symphonies written by Tchaikovsky. The description of each symphony should contain its title, its duration, and the number of parts (movements). Your class should contain a constructor method, get- and set-methods, and a method for computing the average duration of a movement.

2b. Use your class in the *main* method to define a new object *pathetique* of type *Symphony* describing Tchaikovsky's Sixth Symphony that lasts for 46 minutes and consists of 4 movements. Compute and print the average duration of a movement. El Paso Symphony performed it in 48 minutes. Replace 46 with the new value 48, and compute and print the new average duration of a movement.

10/10

2c. Trace your program step-by-step.

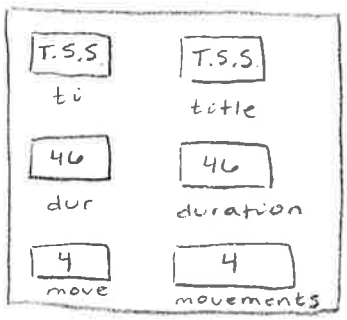
a.)
```
public class Symphony{
    private String title;
    private int duration;
    private int movements;
    public Symphony(String ti, int dur, int move){
        title = ti;
        duration= dur;
        movements= move;}
    public void setTitle(String ti){ title = ti;}
    public void setDuration(int dur){duration= dur;}
    public void setMovements(int move){movements= move;}
    public String getTitle(){return title;}
    public int getDuration(){return duration;}
    public int getMovements(){return movements;}
    public double findAverage(){
        double result = ((double)duration /movements);
        return result;}
}
```

b.) In the main method:

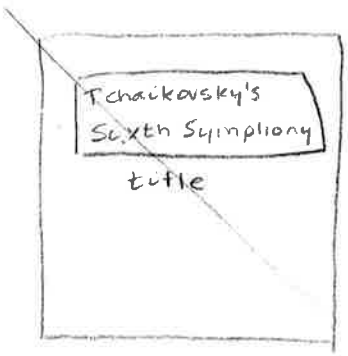
```
Symphony pathetique = new Symphony("Tchaikovsky's Sixth Symphony", 46, 4);
System.out.println(pathetique.findAverage());
pathetique.setDuration(48);
System.out.println(pathetique.findAverage());
```

Part (c.) written on back of page →

c Trace:

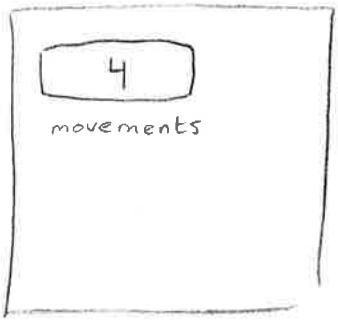


Symphony



setTitle

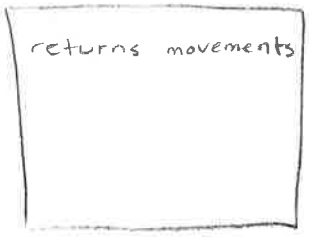
setDuration



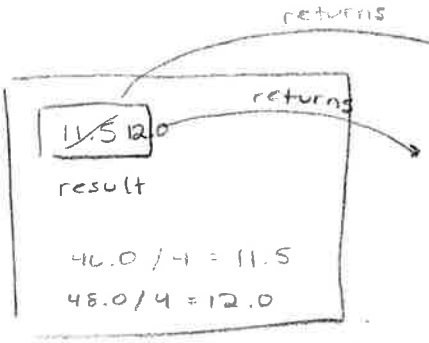
setMovements

getTitle

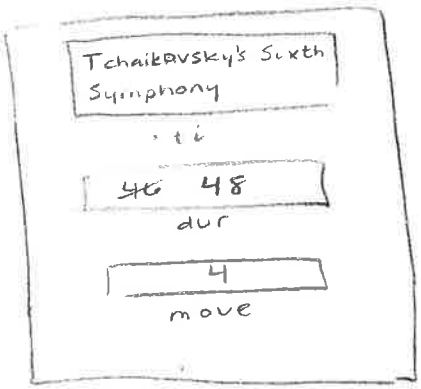
getDuration



getMovements



FindAverage



prints 11.5

prints 12.0

3a. Write a piece of code that, given an array *duration* of durations of different symphonies and an array *parts* of numbers of movements in different symphonies, creates a new array *av* consisting of average durations of a movement in each symphony. Assume the arrays *duration* and *parts* have been declared, initialized, and that they have the same length.

10/10

3b. To check the correctness of the code you wrote in Part 3a, write a piece of code that defines a new array *duration* with values 46 and 60 and a new array *parts* with elements 4 and 5.

3c. Trace step-by-step how the piece of code you wrote in Part 3a will compute the corresponding average durations.

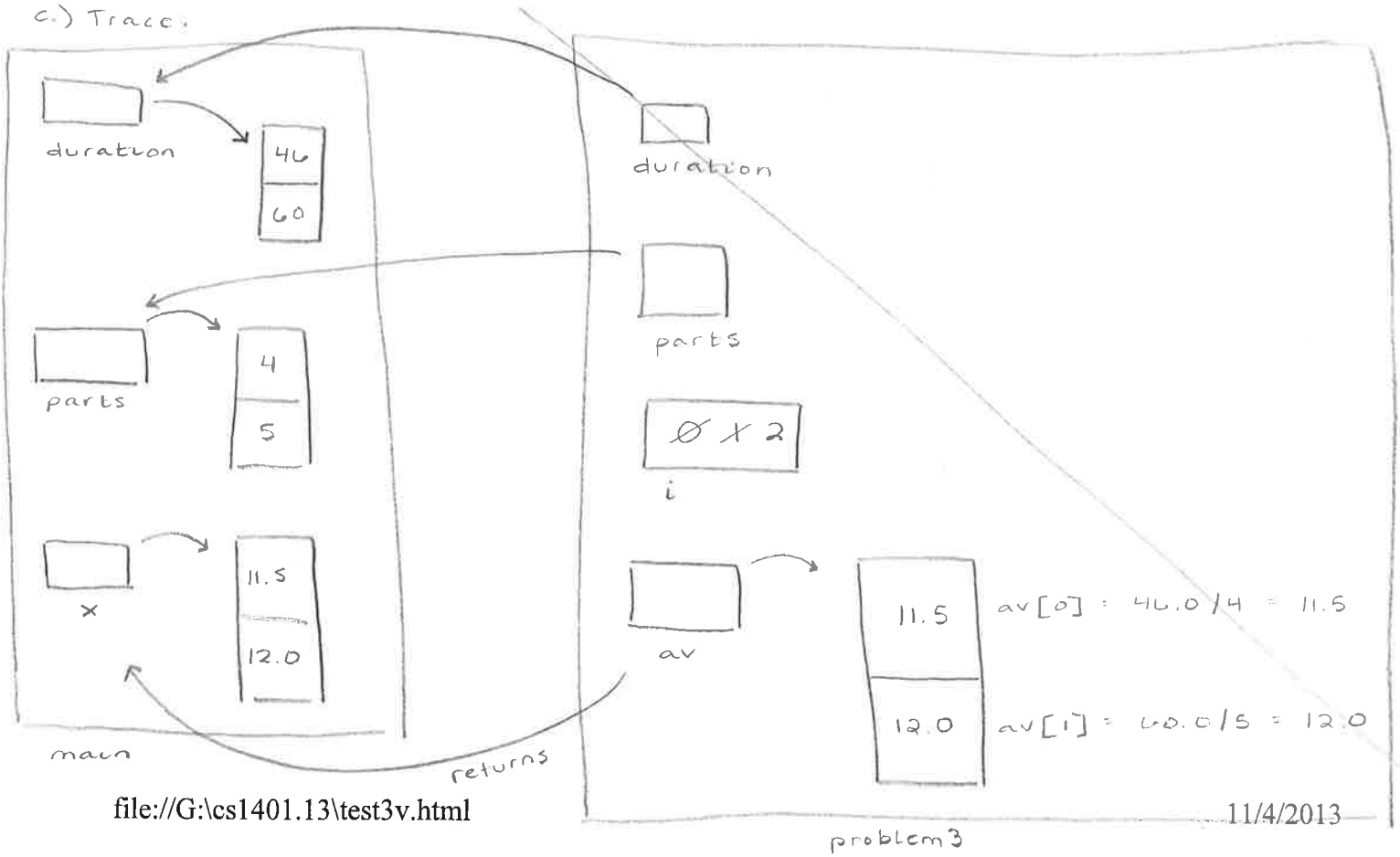
a.)

```
public static double[] problem3(int[] duration, int[] parts){
    double[] av;
    for(int i=0; i<duration.length; i++){
      av[i] = ((double)duration[i]/parts[i]);}
    return av;}
```

b.) In main method:

```
int[] duration = {46,60};
int[] parts = {4,5};
double[] x = problem3(duration, parts);
```

c.) Trace:



4a. Which of Tchaikovsky's symphonies is the longest? Write a method that, given an array *d* of durations of different symphonies and the array *t* of their titles, returns the title of the longest symphony. Assume the arrays have been declared, initialized, and have the same non-zero length.

4b. To check the correctness of your method, write a piece of code that defines new arrays *duration* consisting of 3 elements 46, 60, and 40, and *titles* with elements P, C, and X.

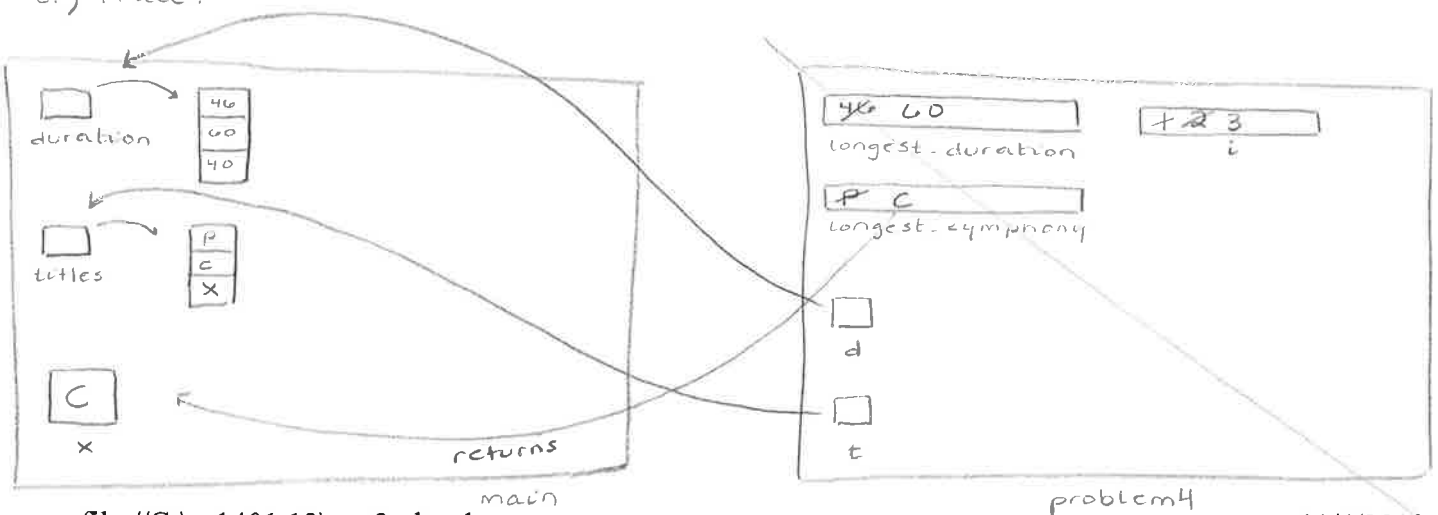
12/10

4c. Trace step-by-step how the piece of code you wrote in Part 4a will find the title of the longest symphony.

a.)
```
public static String problem4(int[] d, String[] t){
    int longest_duration;
    longest_duration = d[0];
    String longest_symphony;
    longest_symphony = t[0];
    for(int i = 1; i < d.length; i++){
      if(d[i] > longest_duration){
        d[i] = longest_duration;
        t[i] = longest_symphony;}
      }
    return longest_symphony;}
```

b.) In main method:
```
int[] duration = {46, 60, 40};
String[] titles = {"P", "C", "X"};
String x = problem4(duration, titles);
```

c.) Trace:



5a. Describe what is white-box and black-box testing.

5b. Describe the main rules for testing.

8/10

a.) White-box testing is a strategy for testing a program in which the tester is fully aware of the internal workings of the program's code.

Black-box testing is an alternate strategy for testing a program in which the tester has no knowledge of the program's code.

b.) For white-box testing, the rule is to trace the program step-by-step.

For black-box testing, the tester simply evaluates the accuracy of outputs for given inputs.

- test all branches
- test on boundary values
- on simple & typical values
- on random values