PA6 – Programming Workflow

Name: Max Rivett
UCSD email: mrivett@ucsd.edu

First 20 Minutes

Screenshot or copy/paste of program:

AveragePositives.java:

```java
class AveragePositives {
    Run | Debug
    public static void main(String[] args) {
        double[] d = new double[args.length];
        double sum = 0;
        double avg = 0;
        int ctr = 0;
        for(String s : args) {
            double num = Double.parseDouble(s);
            if (num > 0) {
                sum += num;
                ctr++;
            }
        }
        avg = sum / ctr;
        if (ctr > 0) {
            System.out.println(avg);
        } else {
            System.out.println(0);
        }
    }
}
```

PairSelect.java:

```java
import tester.*;
class PairSelect {
    static int[] getAs(Pair[] parr) {
        int[] arr = new int[parr.length];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = parr[i].a;
        }
        return arr;
    }
}

class Pair {
    int a,b;
    Pair(int a, int b) {
        this.a=a;
        this.b=b;
    }
}

class ExamplesPairs {
    Pair[] p1 = {new Pair(1,2)};
    int[] a1 = {1};
    void testCase(Tester t) {
        t.checkExpect(PairSelect.getAs(p1), a1);
    }
}
```

Screenshot or copy/paste of ./run or java/javac output (if any):

AveragePositives.java:

```
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives 4 5 6 7 8
6.0
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives 4 -5 6 -7 8
6.0
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives 4 -5 6 -7 -8
5.0
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives -4 -5 -6 -7 -8
0
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives 4.5 5.5 6.5 7.5 8.5
6.5
MacBook-Pro-8:pa6-master max.rivett$ java AveragePositives 4 5 6.66666 7 8
6.133332
```

PairSelect:

N/A

Thoughts on your progress:

I think my progress was pretty solid. I was rushing a little bit so the code doesn't look as clean/organized as I'd like, but it's definitely functional which is what matters. I ended up finishing the first two tasks, although I am still in the process of testing the second task.

Distractions:

Pretty minimal; I've put myself in a quiet environment. There were a couple times when I got distracted a little bit, but I would say that only accounted for around 2 minutes of the 20.

Second 20 minutes:

Screenshot or copy/paste of program:

PairSelect.java:

```
import tester.*;
class PairSelect {
    static int[] getAs(Pair[] parr) {
        int[] arr = new int[parr.length];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = parr[i].a;
        }
        return arr;
    }
}

class Pair {
    int a,b;
    Pair(int a, int b) {
        this.a=a;
        this.b=b;
    }
}

class ExamplesPairs {
    Pair[] p1 = {new Pair(1,2), new Pair(2,2), new Pair(3,0), new Pair(4,4)};
    int[] a1 = {1, 2, 3, 4};
    Pair[] p2 = {new Pair(0,2), new Pair(-1,2), new Pair(3,0), new Pair(10,4)};
    int[] a2 = {0, -1, 3, 10};
    Pair[] p3 = {new Pair(22,-0), new Pair(25/5,10), new Pair(33,0), new Pair(45,4)};
    int[] a3 = {22, 5, 33, 45};
    Pair[] p4 = {new Pair(-1,2), new Pair(-2,-2), new Pair(-3,0), new Pair(-4,-4)};
    int[] a4 = {-1, -2, -3, -4};
    void testCase(Tester t) {
        t.checkExpect(PairSelect.getAs(p1), a1);
        t.checkExpect(PairSelect.getAs(p2), a2);
        t.checkExpect(PairSelect.getAs(p3), a3);
        t.checkExpect(PairSelect.getAs(p4), a4);
    }
}
```

LongStrings.java:

```java
import tester.*;
class PairSelect {
    static int[] getAs(Pair[] parr) {
        int[] arr = new int[parr.length];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = parr[i].a;
        }
        return arr;
    }
}

class Pair {
    int a,b;
    Pair(int a, int b) {
        this.a=a;
        this.b=b;
    }
}

class ExamplesPairs {
    Pair[] p1 = {new Pair(1,2), new Pair(2,2), new Pair(3,0), new Pair(4,4)};
    int[] a1 = {1, 2, 3, 4};
    Pair[] p2 = {new Pair(0,2), new Pair(-1,2), new Pair(3,0), new Pair(10,4)};
    int[] a2 = {0, -1, 3, 10};
    Pair[] p3 = {new Pair(22,-0), new Pair(25/5,10), new Pair(33,0), new Pair(45,4)};
    int[] a3 = {22, 5, 33, 45};
    Pair[] p4 = {new Pair(-1,2), new Pair(-2,-2), new Pair(-3,0), new Pair(-4,-4)};
    int[] a4 = {-1, -2, -3, -4};
    void testCase(Tester t) {
        t.checkExpect(PairSelect.getAs(p1), a1);
        t.checkExpect(PairSelect.getAs(p2), a2);
        t.checkExpect(PairSelect.getAs(p3), a3);
        t.checkExpect(PairSelect.getAs(p4), a4);
    }
}
```

ClosestPoints.java:

```java
class Point {
    int x, y;
    Point(int x, int y) {
        this.x=x;
        this.y=y;
    }
    boolean belowLeftOf(Point p) { return this.x < p.x && this.y < p.y; }
    boolean aboveRightOf(Point p) { return this.x > p.x && this.y > p.y; }

}
```

Screenshot or copy/paste of ./run or java/javac output (if any):

ExamplesPairs:

```
MacBook-Pro-8:pa6-master max.rivett$ ./run ExamplesPairs
Tester Library v.3.0
------------------------------------
Tests defined in the class: ExamplesPairs:
---------------------------
ExamplesPairs:
--------------
new ExamplesPairs:1(
 this.p1 = new Pair[4]:2{
  [0] new Pair:3(
   this.a = 1
   this.b = 2),
  [1] new Pair:4(
   this.a = 2
   this.b = 2),
  [2] new Pair:5(
   this.a = 3
   this.b = 0),
  [3] new Pair:6(
   this.a = 4
   this.b = 4)
 }
 this.a1 = new int[4]:7{
  [0] 1,
  [1] 2,
  [2] 3,
  [3] 4
 }
 this.p2 = new Pair[4]:8{
  [0] new Pair:9(
   this.a = 0
   this.b = 2),
  [1] new Pair:10(
   this.a = -1
   this.b = 2),
  [2] new Pair:11(
   this.a = 3
   this.b = 0),
  [3] new Pair:12(
   this.a = 10
   this.b = 4)
 }
 this.a2 = new int[4]:13{
  [0] 0,
  [1] -1,
  [2] 3,
  [3] 10
 }
 this.p3 = new Pair[4]:14{
  [0] new Pair:15(
   this.a = 22
   this.b = 0),
  [1] new Pair:16(
   this.a = 5
   this.b = 10),
  [2] new Pair:17(
   this.a = 33
   this.b = 0),
  [3] new Pair:18(
   this.a = 45
   this.b = 4)
 }
 this.a3 = new int[4]:19{
  [0] 22,
  [1] 5,
  [2] 33,
  [3] 45
 }
 this.p4 = new Pair[4]:20{
  [0] new Pair:21(
   this.a = -1
   this.b = 2),
  [1] new Pair:22(
   this.a = -2
   this.b = -2),
  [2] new Pair:23(
   this.a = -3
   this.b = 0),
  [3] new Pair:24(
   this.a = -4
   this.b = -4)
 }
 this.a4 = new int[4]:25{
  [0] -1,
  [1] -2,
  [2] -3,
  [3] -4
 })
---------------

Ran 4 tests.
All tests passed.

--- END OF TEST RESULTS ---
```

ExamplesLongStrings:

```
MacBook-Pro-8:pa6-master max.rivett$ ./run ExamplesLongStrings
Tester Library v.3.0
------------------------------------
Tests defined in the class: ExamplesLongStrings:
----------------------------
ExamplesLongStrings:
---------------
new ExamplesLongStrings:1(
 this.s1 = new java.lang.String[10]:2{
  [0] "Hello",
  [1] "my",
  [2] "name",
  [3] "is",
  [4] "Max",
  [5] "which",
  [6] "isn't",
  [7] "short",
  [8] "for",
  [9] "Maximilian"
 }
 this.test1 = new java.lang.String[5]:3{
  [0] "Hello",
  [1] "which",
  [2] "isn't",
  [3] "short",
  [4] "Maximilian"
 }
 this.test2 = new java.lang.String[10]:4{
  [0] "Hello",
  [1] "my",
  [2] "name",
  [3] "is",
  [4] "Max",
  [5] "which",
  [6] "isn't",
  [7] "short",
  [8] "for",
  [9] "Maximilian"
 }
 this.test3 = new java.lang.String[8]:5{
  [0] "Hello",
  [1] "name",
  [2] "Max",
  [3] "which",
  [4] "isn't",
  [5] "short",
  [6] "for",
  [7] "Maximilian"
 }
 this.test4 = new java.lang.String[0]:6{
 })
---------------

Ran 4 tests.
All tests passed.
```

ClosestPoints.java:
N/A

Thoughts on your progress:

Finished the assignment (2 out of 4 tasks) with 13 minutes 17 seconds remaining in the second section.
Got caught up a little bit at the start when trying to use “./run” so I tried a few things and eventually it worked after I copied the “run” files from PA5 and replaced the PA6 ones with them. After that I confirmed that the second task was working, then flew through the third one, and was able to copy the Point class into ClosestPoints.java. To be honest I’m not sure if I’m supposed to do all 4 but at this point I may as well.

Distractions:

Pretty minimal once again. I got distracted a little bit when I hit the “./run” roadblock as one does, but bounced back from it with very few to no distractions after that.

Final 20 minutes:

Screenshot or copy/paste of program:

ClosestPoints.java:

```java
class Point {
    int x, y;
    Point(int x, int y) {
        this.x=x;
        this.y=y;
    }
    boolean belowLeftOf(Point p) { return this.x < p.x && this.y < p.y; }
    boolean aboveRightOf(Point p) { return this.x > p.x && this.y > p.y; }
}

class ClosestPoints {
    Run | Debug
    public static void main(String[] args) {
        Point a = new Point(Integer.parseInt(args[0]), Integer.parseInt(args[1]));
        Point b = new Point(Integer.parseInt(args[2]), Integer.parseInt(args[3]));
        Point c = new Point(Integer.parseInt(args[4]), Integer.parseInt(args[5]));
        double ab = pythagorean(a,b);
        double bc = pythagorean(b, c);
        double ca = pythagorean(c, a);
        double closest = ab;
        Point point1 = a;
        Point point2 = b;
        if (bc < closest) {
            closest = bc;
            point1 = b;
            point2 = c;
        }
        if (ca < closest) {
            closest = ca;
            point1 = c;
            point2 = a;
        }
        System.out.println("The closest points are ("+point1.x+","+point1.y+") and ("+point2.x+","+point2.y+") at distance " + closest);
    }

    static double pythagorean(Point a, Point b) {
        double side1Squared = Math.pow(Math.abs(a.x-b.x), 2);
        double side2Squared = Math.pow(Math.abs(a.y-b.y), 2);
        double hypotenuse = Math.sqrt(side1Squared + side2Squared);
        return hypotenuse;
    }
}
```

Screenshot or copy/paste of ./run or java/javac output (if any):

ClosestPoints.java:

```
MacBook-Pro-8:pa6-master max.rivett$ javac ClosestPoints.java
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints 0 0 3 4 3 900
The closest points are (0,0) and (3,4) at distance 5.0
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints 0 0 3 4 3 2
The closest points are (3,4) and (3,2) at distance 2.0
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints 10 10 20 20 31 31
The closest points are (10,10) and (20,20) at distance 14.142135623730951
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints 10 10 20 20 30 30
The closest points are (10,10) and (20,20) at distance 14.142135623730951
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints 0 0 0 0 30 30
The closest points are (0,0) and (0,0) at distance 0.0
MacBook-Pro-8:pa6-master max.rivett$ -2 -4 -80 -10 -40 -40
bash: -2: command not found
MacBook-Pro-8:pa6-master max.rivett$ java ClosestPoints -2 -4 -80 -10 -40 -40
The closest points are (-80,-10) and (-40,-40) at distance 50.0
```

Thoughts on your progress:

Finished all 4 tasks with 4 minutes 47 seconds remaining in this section.
I thought this task was pretty fun. I chose to make a new method to find the distance between two points which I called “hypotenuse()”. I probably could have been more efficient/organized but I was more focused on getting it done in a way I knew would work.

Distractions:

None.

Overall Reflection

- Where did you spend the most time? You might have different descriptions, some I can think of that I spend a lot of time on are below; you might have others.

  I would say I spent the most time actually programming; the problems weren’t overly complex so I did not need to draw anything out or take too much time thinking it through. In the second section I got caught up a bit with an error I got while trying to use “./run”, but other than that it was pretty smooth. The task that took me the longest was most likely the fourth one.

- What could you do to reduce the time taken in the future?

Maybe find a more efficient way to solve the problems. Although, at the same time, a lot of my speed came from doing the problems the first way that came to my head, if I chose to think them through more the time tradeoff may not have been favourable.

- How did this process compare to how you usually complete PAs?

  But for the reflecting portion, pretty similar. I normally just try to sit down and hunker down for an hour or two, or however long it takes me to work through the PA.

- Overall, what did you learn from this experience about your programming workflow, if anything?

  I learned mainly that I normally like to choose the easy way out in terms of maximizing the efficiency of my programs - I'll look to be more efficient and organized in the way that I write programs in the future.