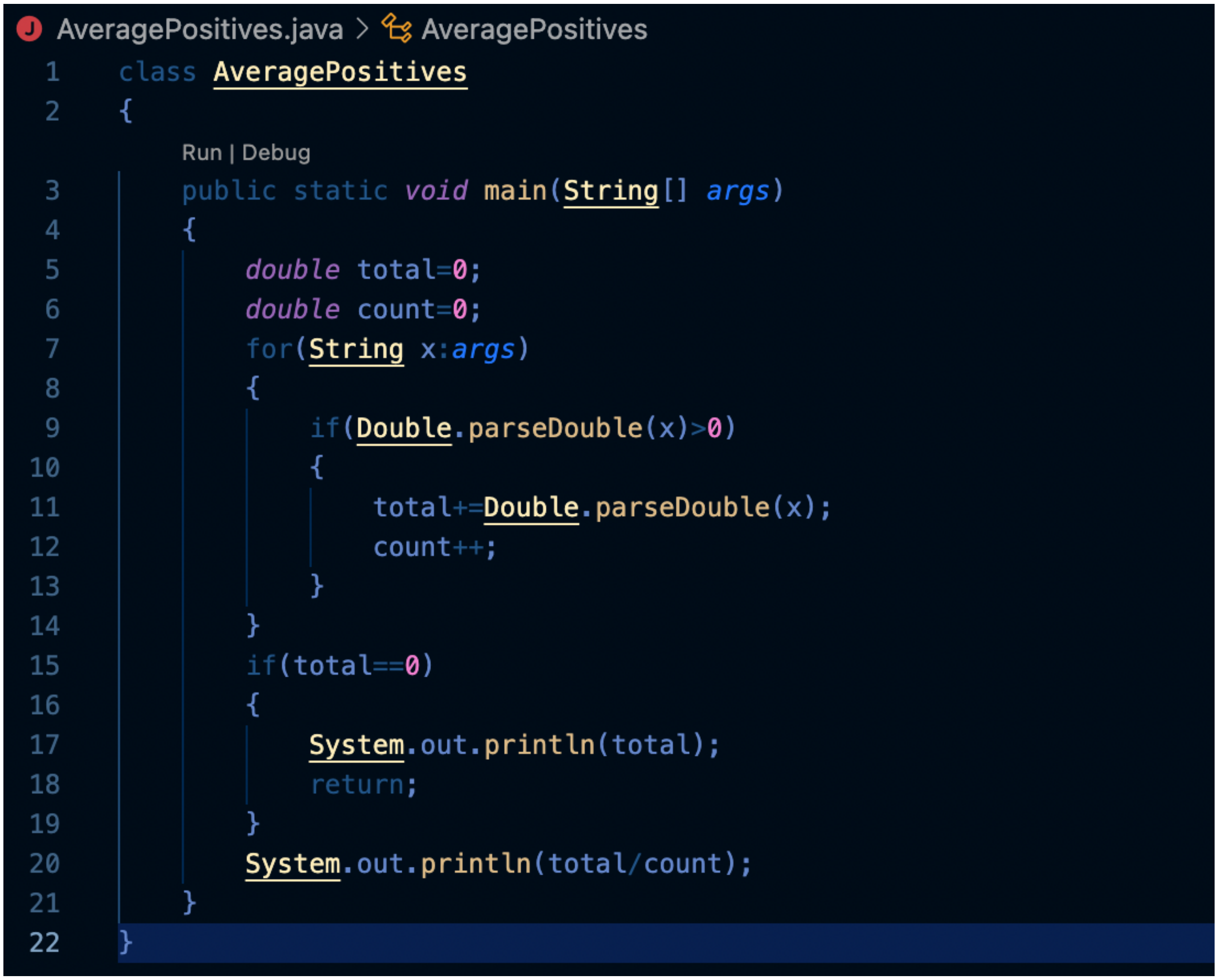
PA6 – Programming Workflow Template

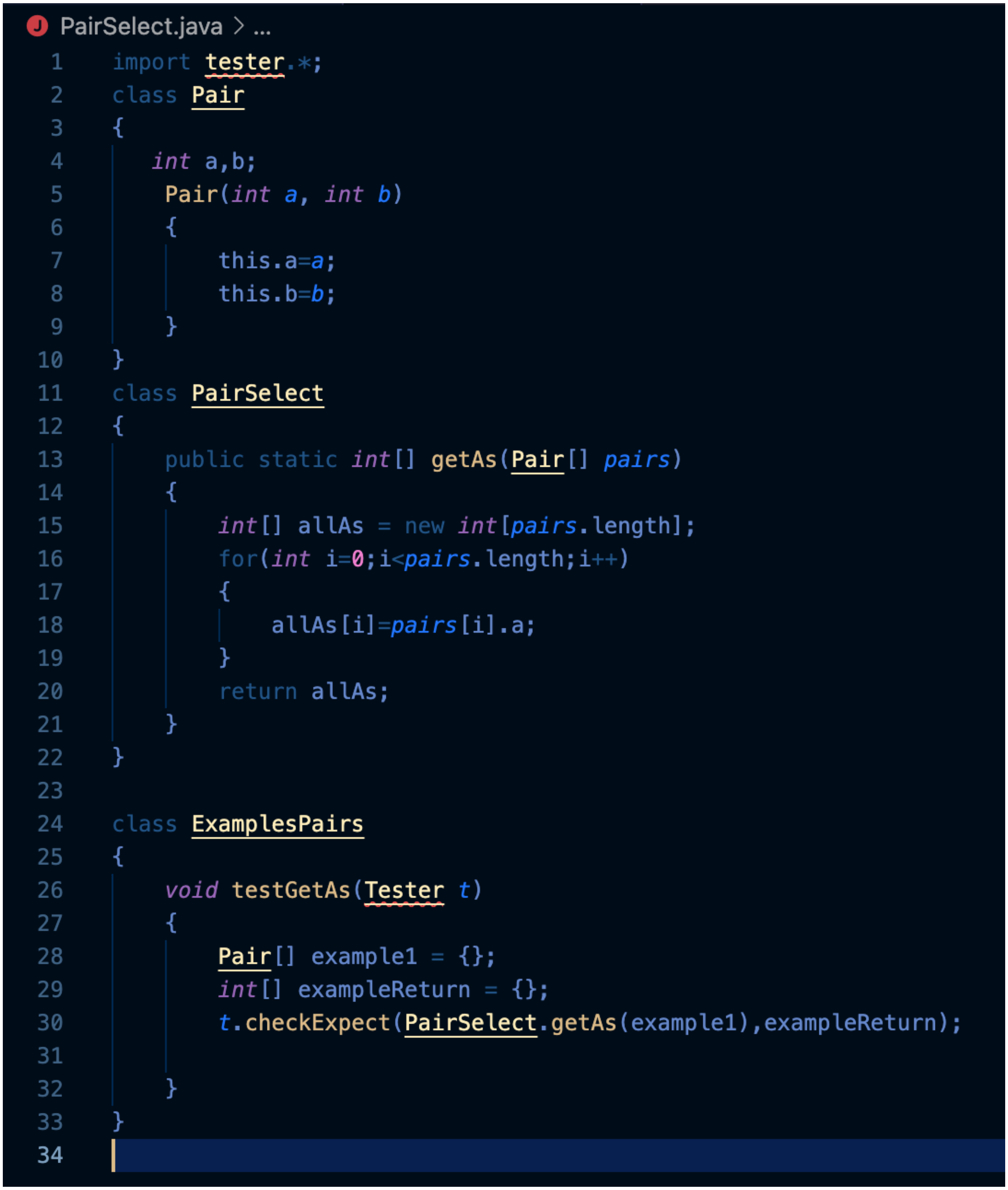
Name: Pranav Vijay
UCSD email: pvijay@ucsd.edu

First 20 Minutes

Screenshot or copy/paste of program:



```java
class AveragePositives
{
    public static void main(String[] args)
    {
        double total=0;
        double count=0;
        for(String x:args)
        {
            if(Double.parseDouble(x)>0)
            {
                total+=Double.parseDouble(x);
                count++;
            }
        }
        if(total==0)
        {
            System.out.println(total);
            return;
        }
        System.out.println(total/count);
    }
}
```



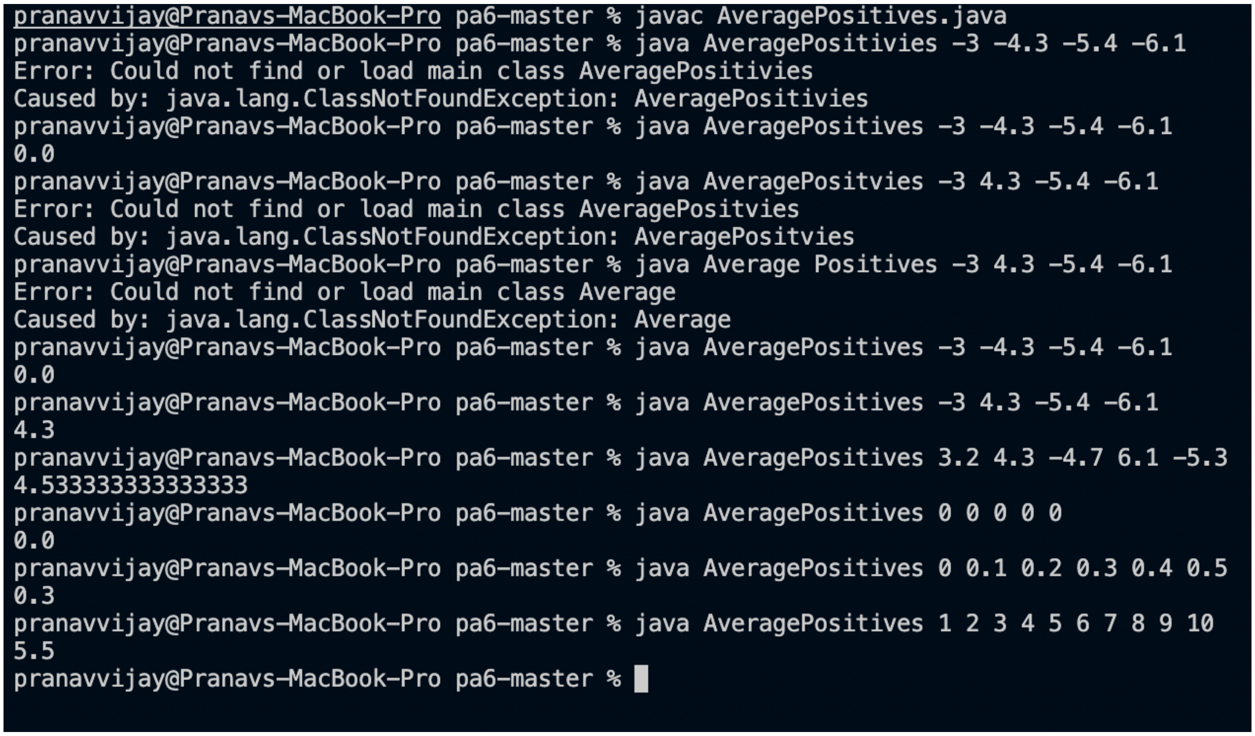
```java
import tester.*;
class Pair
{
   int a,b;
    Pair(int a, int b)
    {
        this.a=a;
        this.b=b;
    }
}
class PairSelect
{
    public static int[] getAs(Pair[] pairs)
    {
        int[] allAs = new int[pairs.length];
        for(int i=0;i<pairs.length;i++)
        {
            allAs[i]=pairs[i].a;
        }
        return allAs;
    }
}

class ExamplesPairs
{
    void testGetAs(Tester t)
    {
        Pair[] example1 = {};
        int[] exampleReturn = {};
        t.checkExpect(PairSelect.getAs(example1),exampleReturn);

    }
}

```

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



```
pranavvijay@Pranavs-MacBook-Pro pa6-master % javac AveragePositives.java
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositivies -3 -4.3 -5.4 -6.1
Error: Could not find or load main class AveragePositivies
Caused by: java.lang.ClassNotFoundException: AveragePositivies
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives -3 -4.3 -5.4 -6.1
0.0
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositvies -3 4.3 -5.4 -6.1
Error: Could not find or load main class AveragePositvies
Caused by: java.lang.ClassNotFoundException: AveragePositvies
pranavvijay@Pranavs-MacBook-Pro pa6-master % java Average Positives -3 4.3 -5.4 -6.1
Error: Could not find or load main class Average
Caused by: java.lang.ClassNotFoundException: Average
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives -3 -4.3 -5.4 -6.1
0.0
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives -3 4.3 -5.4 -6.1
4.3
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives 3.2 4.3 -4.7 6.1 -5.3
4.533333333333333
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives 0 0 0 0 0
0.0
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives 0 0.1 0.2 0.3 0.4 0.5
0.3
pranavvijay@Pranavs-MacBook-Pro pa6-master % java AveragePositives 1 2 3 4 5 6 7 8 9 10
5.5
pranavvijay@Pranavs-MacBook-Pro pa6-master % 
```

Thoughts on your progress:

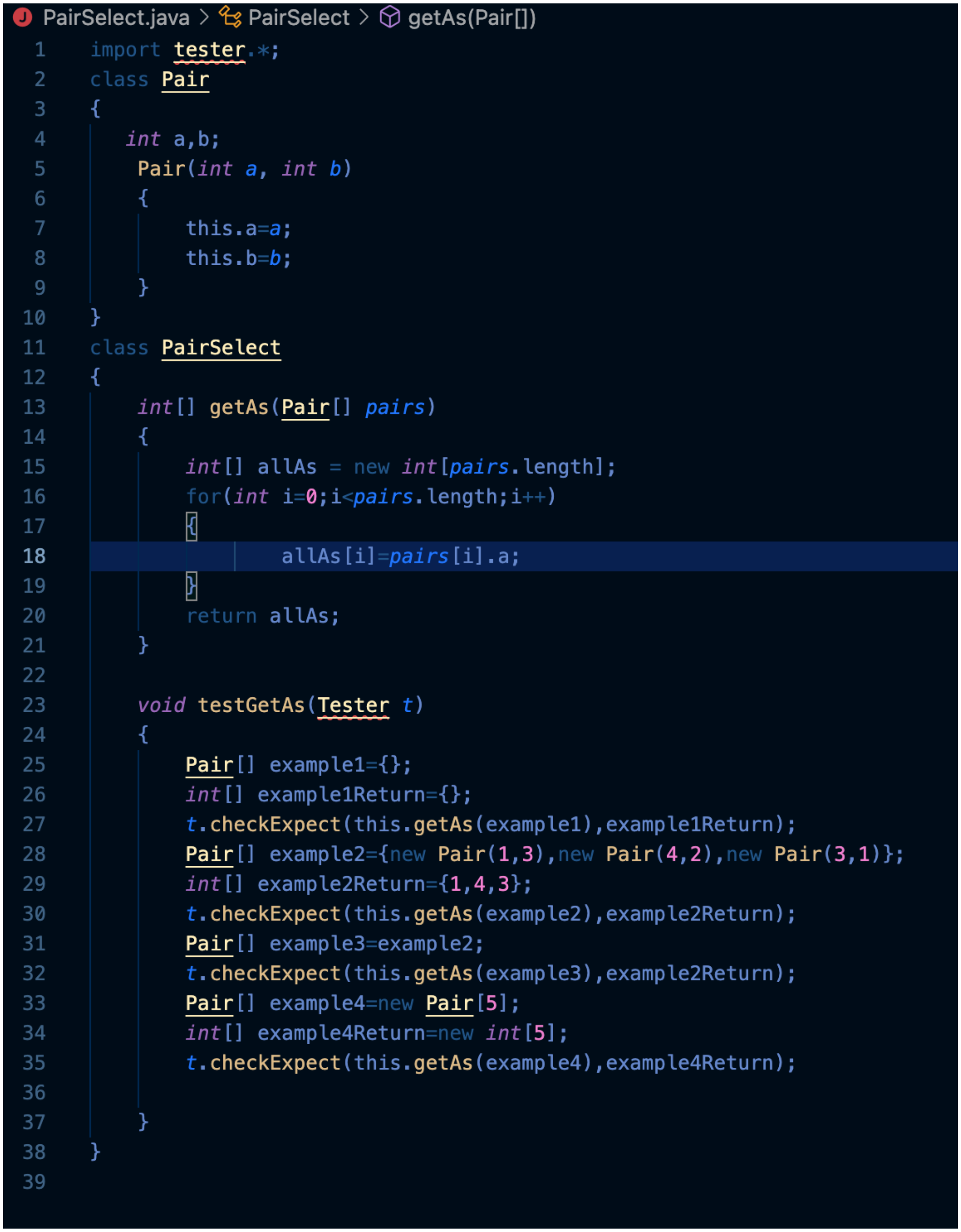
I think I made a good amount of progress. I finished all the classes and examples for the first task and finished the classes for the second task. I am still working on the test examples on that second task.

Distractions:

Yes, about 5 minutes of distractions.

Second 20 minutes:

Screenshot or copy/paste of program:

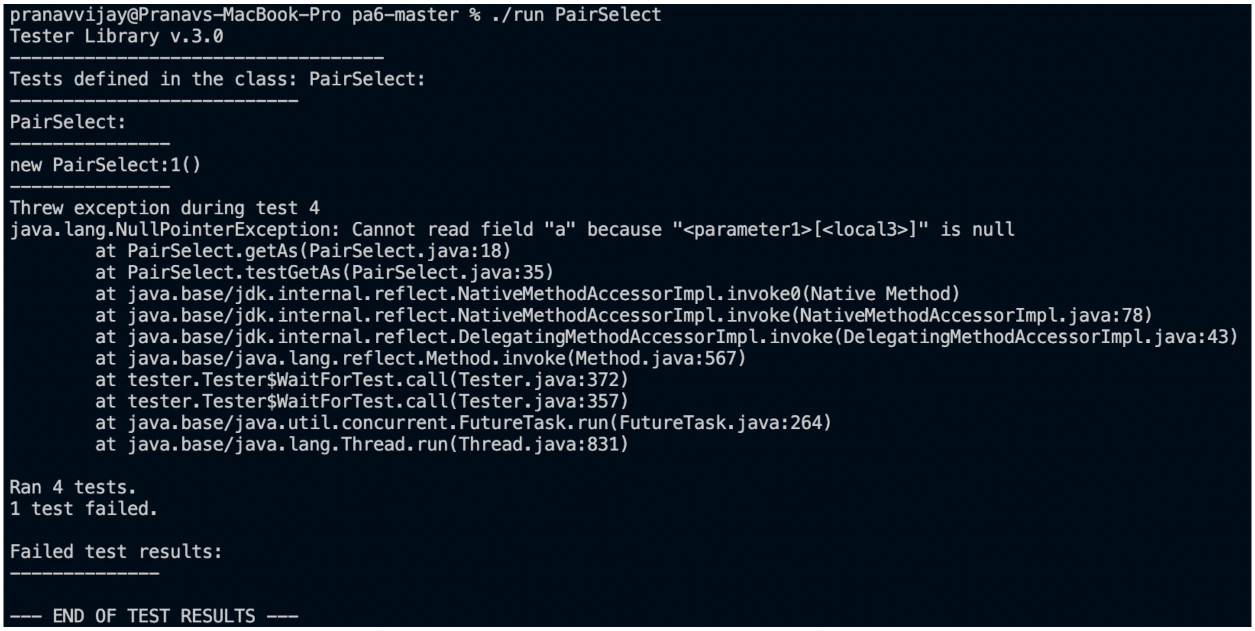


```
import tester.*;
class Pair
{
   int a,b;
    Pair(int a, int b)
    {
        this.a=a;
        this.b=b;
    }
}
class PairSelect
{
    int[] getAs(Pair[] pairs)
    {
        int[] allAs = new int[pairs.length];
        for(int i=0;i<pairs.length;i++)
        {
                allAs[i]=pairs[i].a;
        }
        return allAs;
    }

    void testGetAs(Tester t)
    {
        Pair[] example1={};
        int[] example1Return={};
        t.checkExpect(this.getAs(example1),example1Return);
        Pair[] example2={new Pair(1,3),new Pair(4,2),new Pair(3,1)};
        int[] example2Return={1,4,3};
        t.checkExpect(this.getAs(example2),example2Return);
        Pair[] example3=example2;
        t.checkExpect(this.getAs(example3),example2Return);
        Pair[] example4=new Pair[5];
        int[] example4Return=new int[5];
        t.checkExpect(this.getAs(example4),example4Return);

    }
}
```

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



```
pranavvijay@Pranavs-MacBook-Pro pa6-master % ./run PairSelect
Tester Library v.3.0
-----------------------------------
Tests defined in the class: PairSelect:
---------------------------
PairSelect:
---------------
new PairSelect:1()
---------------
Threw exception during test 4
java.lang.NullPointerException: Cannot read field "a" because "<parameter1>[<local3>]" is null
        at PairSelect.getAs(PairSelect.java:18)
        at PairSelect.testGetAs(PairSelect.java:35)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:78)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:567)
        at tester.Tester$WaitForTest.call(Tester.java:372)
        at tester.Tester$WaitForTest.call(Tester.java:357)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
        at java.base/java.lang.Thread.run(Thread.java:831)

Ran 4 tests.
1 test failed.

Failed test results:
--------------

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

Thoughts on your progress:

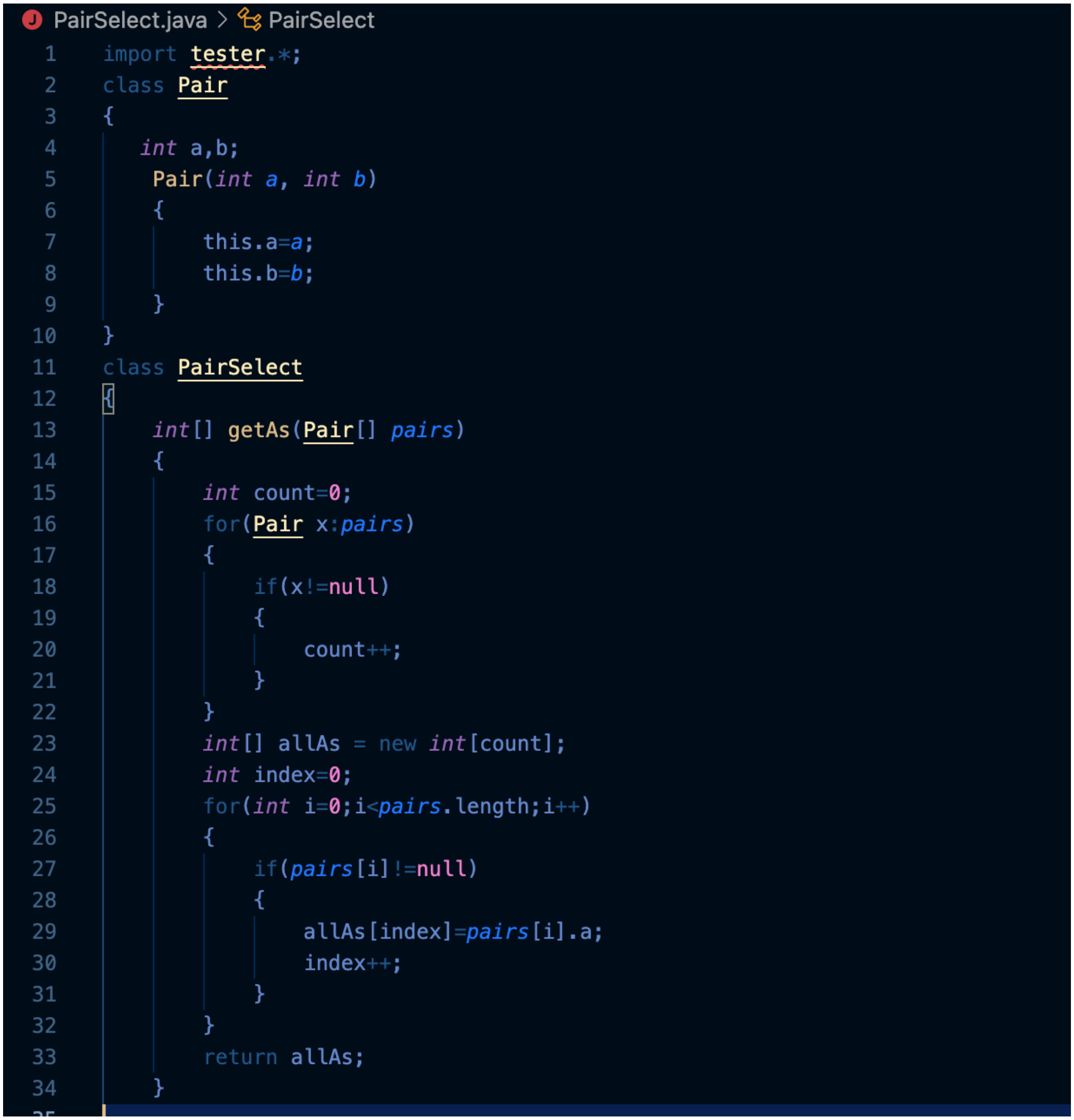
I made significantly less progress during these 20 minutes. I struggled a lot making unique and interesting test cases since this is such a simple class. During the last 20 minutes I need to figure out how to deal with an array with null values for my final example.

Distractions:

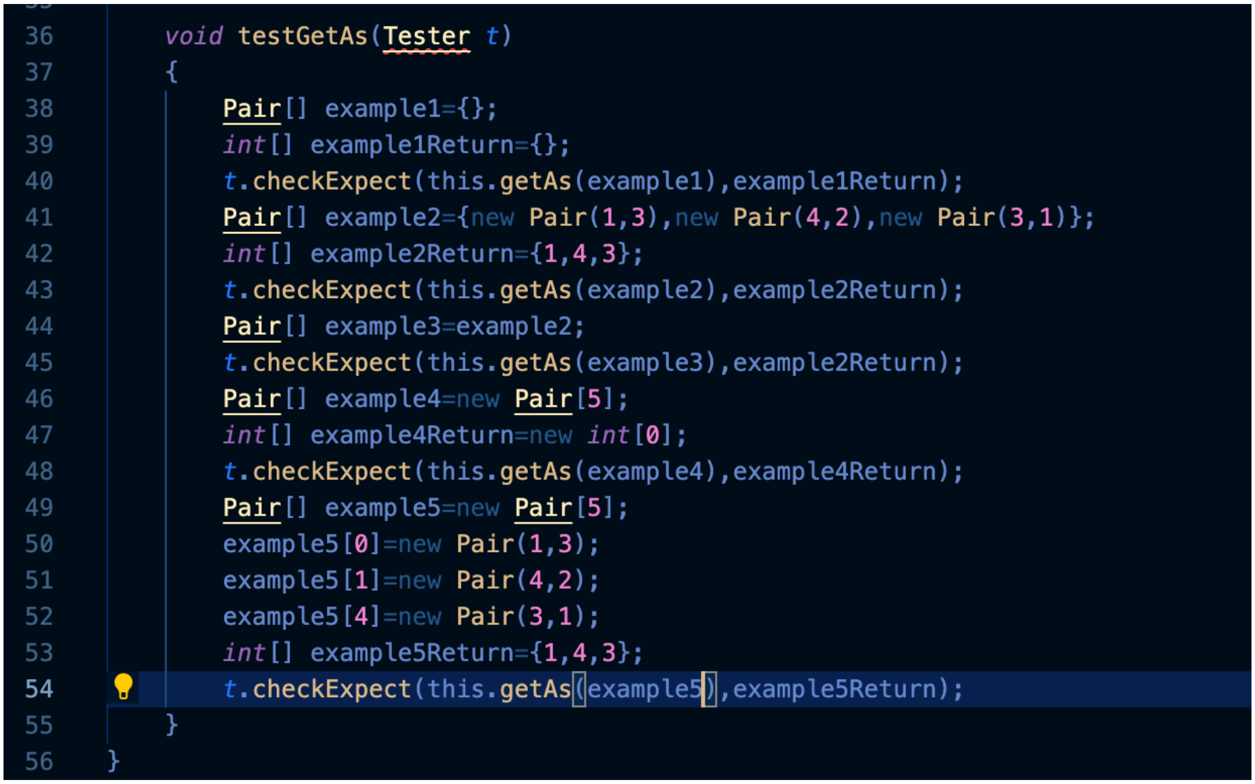
Yes, about 2-3 minutes of distractions.

Final 20 minutes: Finished in 10 minutes

Screenshot or copy/paste of program:

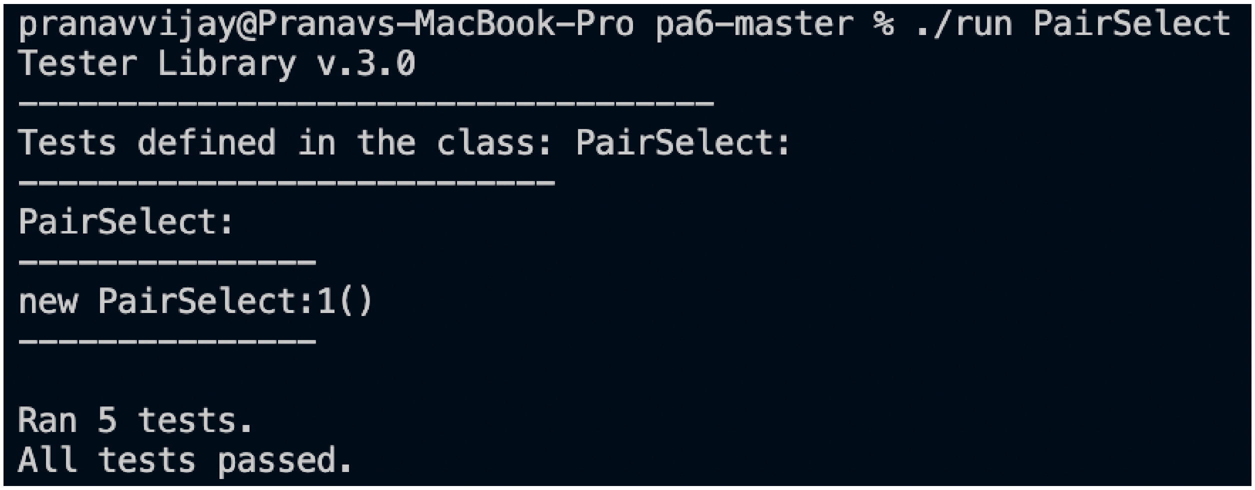


```java
import tester.*;
class Pair
{
   int a,b;
    Pair(int a, int b)
    {
        this.a=a;
        this.b=b;
    }
}
class PairSelect
{
    int[] getAs(Pair[] pairs)
    {
        int count=0;
        for(Pair x:pairs)
        {
            if(x!=null)
            {
                count++;
            }
        }
        int[] allAs = new int[count];
        int index=0;
        for(int i=0;i<pairs.length;i++)
        {
            if(pairs[i]!=null)
            {
                allAs[index]=pairs[i].a;
                index++;
            }
        }
        return allAs;
    }
```



```
    void testGetAs(Tester t)
    {
        Pair[] example1={};
        int[] example1Return={};
        t.checkExpect(this.getAs(example1),example1Return);
        Pair[] example2={new Pair(1,3),new Pair(4,2),new Pair(3,1)};
        int[] example2Return={1,4,3};
        t.checkExpect(this.getAs(example2),example2Return);
        Pair[] example3=example2;
        t.checkExpect(this.getAs(example3),example2Return);
        Pair[] example4=new Pair[5];
        int[] example4Return=new int[0];
        t.checkExpect(this.getAs(example4),example4Return);
        Pair[] example5=new Pair[5];
        example5[0]=new Pair(1,3);
        example5[1]=new Pair(4,2);
        example5[4]=new Pair(3,1);
        int[] example5Return={1,4,3};
        t.checkExpect(this.getAs(example5),example5Return);
    }
}
```

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



```
pranavvijay@Pranavs-MacBook-Pro pa6-master % ./run PairSelect
Tester Library v.3.0
-----------------------------------
Tests defined in the class: PairSelect:
---------------------------
PairSelect:
---------------
new PairSelect:1()
---------------

Ran 5 tests.
All tests passed.
```

Thoughts on your progress:

While coming up with those examples, I figured out that I had to change my class code. So I was able to make improvements on my code from the previous 20 minutes to work with more unique examples dealing with arrays with null values.

Distractions:

No

Overall Reflection

I definitely spent the most time figuring out different and unique test cases to test important examples and aspects of my code. For the most part writing the code is rather easy, but figuring out those unique test cases helped me improve my code so it can be used with more unusual inputs. I also mistype a lot when I am trying to code quickly which can just take me more time to understand where I messed up and if it was the code or my terminal input. To reduce time in the future, I could just limit my distractions by putting my phone away or I could take enough time when I am coding to make sure I typed everything the way I wanted it to be typed. It was pretty similar to how I usually complete PAs. Usually I try to finish in one sitting without really limiting my distractions, but I still get done in a relatively timely manner. I learned that I work best when sitting down and coding in one sitting, but once I come across errors that I am really stuck on, it is beneficial for me to get away from my computer for a bit because sometimes the solutions to my code just come to me while I'm out and about during my normal day.