Assignment Eight

Student's Name: Tara Aida **Course Name: Computer Science 01100101**

Teacher's Name: Joe Puccio

Write a program that approximates the root of a function by applying Newton's method 100 times. The function can take the following input: the function and the function's derivative (in terms of a variable, 'x'), and a starting x point.

Example of recursion (I recommend copying and pasting the code into Eclipse for easier reading): This program takes in an ordered array of numbers and when you give it an integer, it'll find out where that number falls in the array (so it may be between two values in the array, or it may be a value in the array, or it'll return an error if it's not either of those two cases).

This is the class that contains the bulk of the code (the method that we're using to do our dirty work):

```
public class recursionClass {

	public void finder(int numberList[],int inputValue, int currentIndex, int previousIndex, int penultimateIndex){

		int firstIndex = 0;
		int lastIndex = numberList.length-1;

		if(currentIndex<firstIndex||currentIndex>lastIndex||currentIndex==lastIndex||currentIndex==firstIndex){ //if the index being checked is a boundary point or outside the boundary of the array, then it terminates because it can't be tested.

System.out.println("It may be the case that "+ inputValue +" does not have a home in our array :(. The problem may be that your starting index is out of or on the bounds of the array.");

		}else if(currentIndex==penultimateIndex){ //we're bouncing between two values which means we know that the inputValue falls between what we're bouncing between

	System.out.println("Good news! The value you inputted lies between the value "+ numberList[previousIndex] +" and the value "+ numberList[currentIndex] +" in the array.");

		}

		else{ //if none of the above conditions are true, then we haven't settled down and so we have to continue searching

			int valueToTest = numberList[currentIndex];

			if(valueToTest>inputValue){

				int newIndexToTest = currentIndex-1;

				finder(numberList,inputValue,newIndexToTest,currentIndex,previousIndex); //the method calls itself. This gives it the recursive property.

			}else if(valueToTest<inputValue){

				int newIndexToTest = currentIndex+1;

				finder(numberList,inputValue,newIndexToTest,currentIndex,previousIndex); //the method calls itself. This gives it the recursive property.

			}else{//the inputValue is equal to something in the array

System.out.println("Good news! The value you inputted is equal to something in the array, so your input value lies between the value "+ numberList[currentIndex-1] +" and the value "+ numberList[currentIndex+1] +" in the array.");

			}


		}

	}

}
```

Assignment Eight

This is the main method class (the do-er class), that will actually use what we have written in the other class.

```
public class recursionExample {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub

		int numberList[] = new int[10]; //this is setting up the array that we're using to test
		numberList[0]=0;
		numberList[1]=2;
		numberList[2]=3;
		numberList[3]=5;
		numberList[4]=7;
		numberList[5]=13;
		numberList[6]=17;
		numberList[7]=19;
		numberList[8]=23;
		numberList[9]=29;

		recursionClass instanceOfRecursionClass = new recursionClass(); //this is creating an instance of the class we created

		instanceOfRecursionClass.finder(numberList, 5, 6, 0, 0); //this is starting the method that's found in the recursionClass class.


	}
}
```

Running the main method that's pasted above will yield the following output: `Good news! The value you inputted is equal to something in the array, so your input value lies between the value 3 and the value 7 in the array.`

Running `instanceOfRecursionClass.finder(numberList, 10, 6, 0, 0);` instead will yield the following output: `Good news! The value you inputted lies between the value 7 and the value 13 in the array.`

Send your source code to the following email address for grading:

Joe@pooch.us