Isaac Ray Shoebottom
CS 1083
Assignment 7
3429069

Source Code:

TestSeq.java:

```
import java.text.NumberFormat;
import java.util.Scanner;

/**
 A simple driver that uses the Seq class to compute the
 nth element of the sequence.
 @author Isaac Shoebottom (3429069)
*/

public class TestSeq{

	public static void main(String[] args){

		int n, seqRec, seqMem, seqIter;

		Scanner scan = new Scanner(System.in);
		System.out.print("Enter a positive integer: ");
		n = scan.nextInt();

		seqRec = Seq.seqR(n);
		seqMem = Seq.seqM(n);
		seqIter = Seq.seqI(n);
		System.out.println("seqR(" + n + ") is: " + seqRec);
		System.out.println("seqM(" + n + ") is: " + seqMem);
		System.out.println("seqI(" + n + ") is: " + seqIter);
```

```
            NumberFormat form = NumberFormat.getInstance();
            form.setMaximumFractionDigits(7);
            form.setMinimumFractionDigits(7);
            System.out.println();
            System.out.println("Execution Times in Milliseconds (ms)");
            System.out.println("Seq(n) \tRecursive \tMemoization \tItertive");
            long start, end;
            int seqA;
            double time;
            for(int i = 20; i <= 40; i+=10){
                  start = System.nanoTime();
                  seqA = Seq.seqR(i);
                  end = System.nanoTime();
                  time = (double)(end-start)/1000000;
                  System.out.print(i + "\t" + form.format(time));


                  start = System.nanoTime();
                  seqA = Seq.seqM(i);
                  end = System.nanoTime();
                  time = (double)(end-start)/1000000;
                  System.out.print(i + "\t" + form.format(time));

                  start = System.nanoTime();
                  seqA = Seq.seqI(i);
                  end = System.nanoTime();
                  time = (double)(end-start)/1000000;
                  System.out.print(i + "\t" + form.format(time));

                  System.out.println();
            }
      }
}
```

Seq.java:

```
import java.util.ArrayList;

/**
A utility class that provide methods to compute elements of the
recursive sequence.
@author Leah Bidlake, Isaac Shoebottom (3429069)
*/

public class Seq{

	/**
	Recursively computes seq(n).
	@param n Non-negative integer.
	@return int Element n in the recursive sequence.
	*/

	private static ArrayList<Integer> cache;

	public static int seqR(int n){
		if (n == 0)
			return 1;
		if (n == 1)
			return 5;
		return seqR(n-1) + seqR(n-2);
	}

	/**
	 * Sequence calculator using an memoization cache
	 * @param n The digit the sequence is to be calculated to
	 * @return The number in the sequence n refers to
	 */
	public static int seqM(int n){
```

```
		if (cache == null) {
			cache = new ArrayList<>();
			cache.add(1);
			cache.add(5);
		}

		if (cache.size() <= n)
			cache.add(seqM(n-1) + seqM( n-2));
		return cache.get(n);
	}

	/**
	 * Sequence calculator using an iterative approach
	 * @param n The digit the sequence is to be calculated to
	 * @return The number in the sequence n refers to
	 */
	public static int seqI(int n){
		int[] cache = new int[n+2]; //n+2 so seqI(0) is not out of bounds
		cache[0] = 1;
		cache[1] = 5;

		for(int i = 2; i <= n; i++) { //i = 2 because we need to skip the first two
array items
			cache[i] = cache[i-1] + cache[i-2];
		}
		return cache[n];
	}
}
```

Output:

```
Enter a positive integer: 5

seqR(5) is: 28

seqM(5) is: 28

seqI(5) is: 28


Execution Times in Milliseconds (ms)

Seq(n)  Recursive       Memoization     Itertive

20      0.342400020     0.016900020     0.0015000

30      3.604300030     0.018300030     0.0020000

40      429.767200040   0.024100040     0.0021000
```