Assignment 2 Assessment:

I copied the method directly from my program, it might be easier to read in Eclipse. I will summarize the final analysis:

Run time for the method: line 67-77: O(1), 78-80: O(n), 88-121: O(nm), 123-135 O(n).

Final math: C(all the constant time operations) + O(n) (2 loops) + O(mn) = **O(n + nm).**

The result of my analysis is that my day method should have a big O of (n + mn).

```
67-77: O(1), 78-80: O(n),  88-121: O(nm) 123-135 O(n).
             * Final math: C(all the constant time operations) + 2O(n) (2
loops) + O(mn) = O(n + nm).


/**
       * Method to complete a breeding cycle.  Update's mostFit to the Genome
with lowest fitness
       * score.  Re-populates myPop with the 50 most fit Genome's of the
previous day's cycle.
       * Creates new Genome's to bring the total of myPop back to 100 using 2
options: clone a
       * random Genome and mutate it, clone a random genome and crossover
with another random
       * Genome and mutate the result.
       *
       * Run time for method: line 67-77: O(1), 78-80: O(n),  88-121: O(nm)
123-135 O(n).
       * Final math: C(all the constant time operations) + 2O(n) (2 loops) +
O(mn) = O(n + nm).
       */
      public void day() {
            //Declare a random to use in the method.
            Random rand = new Random();

            //Declare an array of MAX_POP size to hold the top Genome's and
newly created Genome's.
            final Genome[] temp = new Genome[MAX_POP];

            /*
             * Create a temporary PriorityQueue to hold the top 50 best
Genome's.  Transfer the
             * top 50 back to myPop.
             *
             * Run time is the size of the target minimum population: O(n).
             */
            for (int i = 0; i < MIN_POP; i++) {
                  temp[i] = myPop.remove();
            }

            /*
```

```
             * Re-populate the queue.
             *
             * Run time for the for loop is: 87-100: O(1), 101 O(m), 102-114:
O(1), 115: O(m), 116: O(m),
             * 117-121: O(1).  Final math: C + (O(n) x 3O(m)) = O(nm).
             */
            for (int i = MIN_POP; i < MAX_POP; i++) {

                  //Make a 'coin' to flip and randomly choose one of the 2
options.
                  final int coin = rand.nextInt(2);

                  //Flip the coin.
                  if (coin == 0) {
                        /*
                         * Pick random index within the bounds of 0 to i - 1.
Create a new Genome by cloning
                         * the Genome at the random index.  Mutate the new
Genome. Add it to temp at index i.
                         */
                        final int index = rand.nextInt(i - 1);
                        final Genome target = new Genome(temp[index]);
                        target.mutate(); //O(m).
                        temp[i] = target;
                        //Enable S.O.P for test cases.
//                      System.out.println("We are Mutating!");
                  } else {
                        /*
                         * Pick 2 random index's and clone the Genome's at
those indexes.  Crossover the first
                         * Genome with the second and then mutate the result.
Add the result to temp at
                         * index i.
                         */
                        final int index1 = rand.nextInt(i - 1);
                        final int index2 = rand.nextInt(i - 1);
                        final Genome parentMom = new Genome(temp[index1]);
                        final Genome parentDad = new Genome(temp[index2]);
                        parentMom.crossover(parentDad); //O(m)
                        parentMom.mutate(); //O(m)
                        temp[i] = parentMom;
                        //Enable S.O.P for test cases.
//                      System.out.println("We are Crossing Over!");
                  }
            }

            /*
             * First run fitness on each Genome in temp to update the fitness
score for that Genome,
             * then transfer the Genome into myPop.
             *
             * Run time for a for loop is O(n).
             */
            myPop = new PriorityQueue<Genome>();

            for (int i = 0; i < MAX_POP; i++) {
                  temp[i].fitness();
```

```
                myPop.add(temp[i]);
            }
        }
```