E26

Designs	Advantages	Disadvantages
Design1	- Efficient memory use	- Inefficient in terms of creating instances - Inefficient doing computations requiring both coordinate systems - Confusing code (not as straightforward)
Design 2	- Efficient in terms of creating instances - Efficient doing computations requiring both coordinate systems - Simple code	- Inefficient memory use - Confusing code (not as straightforward)
Design 3	- Efficient in terms of creating instances - Efficient doing computations requiring both coordinate systems - Simple code	- Inefficient memory use
Design 4	- Efficient doing computations requiring both coordinate systems - Simple code	- Inefficient in terms of creating instances - Inefficient memory use
Design 5	- Efficient in terms of creating instances - Efficient doing computations requiring both coordinate systems - Simple code	- Inefficient memory use

E28-E29-E30 (Running all methods)

Designs	Run 1	Run 2	Run 3	Run 4	Run 5
Design 1 (Pointcp)	9.565 seconds	9.746 seconds	10.770 seconds	9.501 seconds	9.491 seconds
Design 5 (Pointcp 2)	8.057 seconds	7.684 seconds	7.864 seconds	7.830 seconds	7.855 seconds
Design 5 (Pointcp 3)	6.934 seconds	6.808 seconds	6.827 seconds	6.757 seconds	6.816 seconds

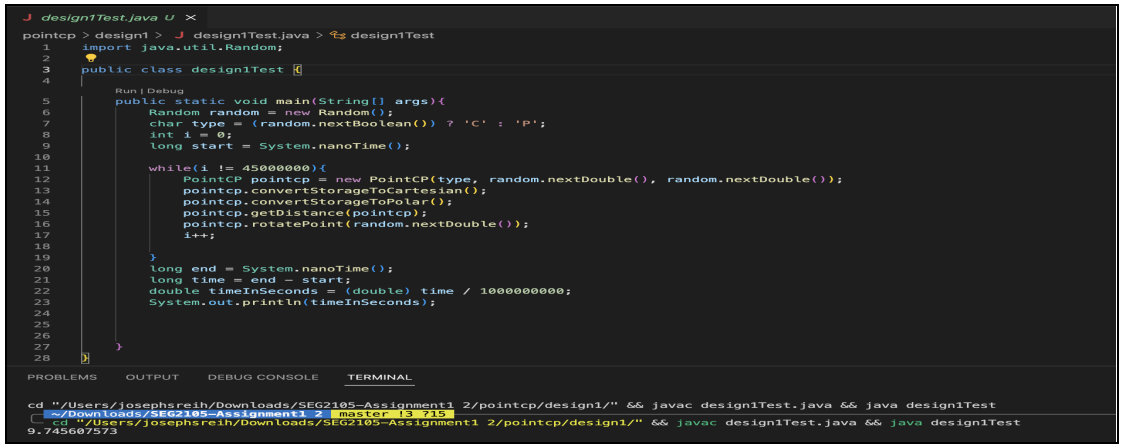
The average runtime for each design when calling all methods
Design 1: 9.815 seconds

Design 2: 7.822 seconds

Design 3: 6.720 seconds

Conclusion: Pointcp seems to be inefficient in terms of creating instances compared to Pointcp 2 and 3.This is because *typecoord* does not need to be assigned a value in its constructor for Pointcp2 and 3, compared to Pointcp which assigns the type value to *typecoord.* As for memory space, Pointcp 2 and 3 will end up using more memory space since calling the convert methods might result in a new instance being created. Regarding computation, Pointcp will be the most inefficient, since it will end up calling math functions almost always for the convert methods, *getDistance* method and *rotatePoint* method. Pointcp 2 will be a bit more efficient in that regard because if it calls the *convertToPolar* method, it will simply return what is already stored instead of using math functions. Even so, Pointcp 3 will still run the fastest when doing computations. Not only is it faster when doing *convertToCartesian* since the instance stores values as x and y, the get methods will simply return the already stored values so when the math is being done for *getDistance* and *rotatePoint*, the computation will be minimal.

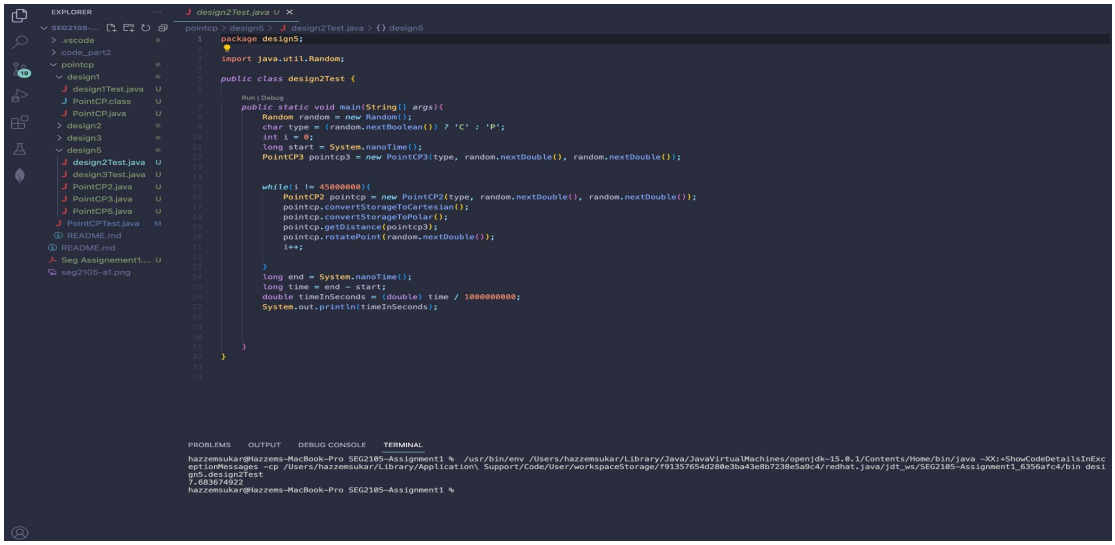
Sample outputs underneath



```
design1Test.java U
pointcp > design1 > design1Test.java > design1Test
```

```java
import java.util.Random;

public class design1Test {

    Run | Debug
    public static void main(String[] args){
        Random random = new Random();
        char type = (random.nextBoolean()) ? 'C' : 'P';
        int i = 0;
        long start = System.nanoTime();

        while(i != 45000000){
            PointCP pointcp = new PointCP(type, random.nextDouble(), random.nextDouble());
            pointcp.convertStorageToCartesian();
            pointcp.convertStorageToPolar();
            pointcp.getDistance(pointcp);
            pointcp.rotatePoint(random.nextDouble());
            i++;

        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println(timeInSeconds);


    }
}
```

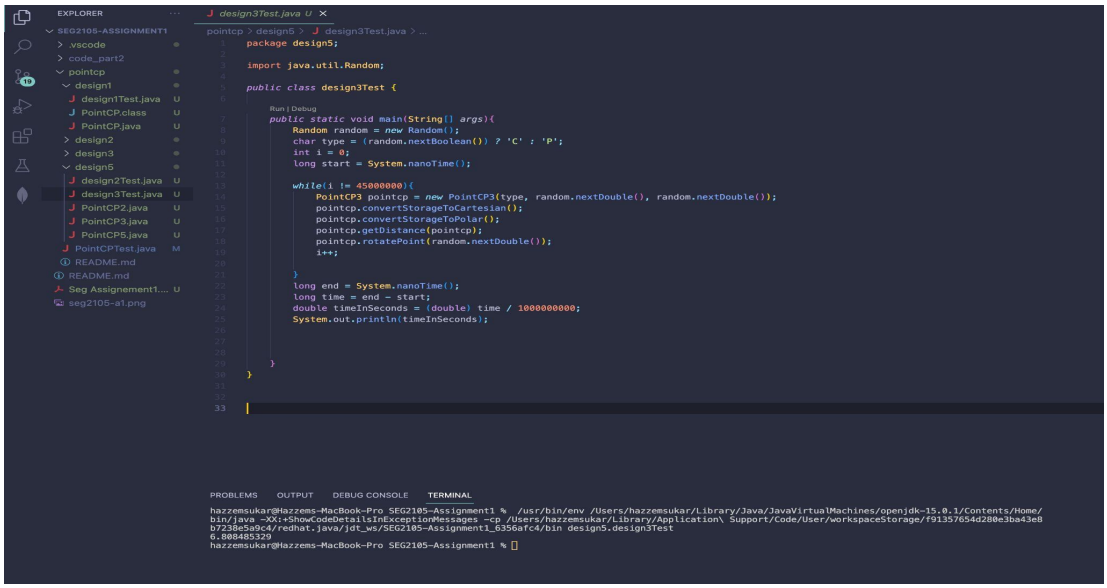
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

cd "/Users/josephsreih/Downloads/SEG2105-Assignment1 2/pointcp/design1/" && javac design1Test.java && java design1Test
 ~/Downloads/SEG2105-Assignment1 2  master !3 ?15
 cd "/Users/josephsreih/Downloads/SEG2105-Assignment1 2/pointcp/design1/" && javac design1Test.java && java design1Test
9.745607573
```



```
design2Test.java U
pointcp > design5 > design2Test.java > design5
```

```java
package design5;

import java.util.Random;

public class design2Test {

    Run | Debug
    public static void main(String[] args){
        Random random = new Random();
        char type = (random.nextBoolean()) ? 'C' : 'P';
        int i = 0;
        long start = System.nanoTime();
        PointCP3 pointcp3 = new PointCP3(type, random.nextDouble(), random.nextDouble());


        while(i != 45000000){
            PointCP2 pointcp = new PointCP2(type, random.nextDouble(), random.nextDouble());
            pointcp.convertStorageToCartesian();
            pointcp.convertStorageToPolar();
            pointcp.getDistance(pointcp3);
            pointcp.rotatePoint(random.nextDouble());
            i++;

        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println(timeInSeconds);


    }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

hazzemsukar@Hazzems-MacBook-Pro SEG2105-Assignment1 %  /usr/bin/env /Users/hazzemsukar/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/hazzemsukar/Library/Application\ Support/Code/User/workspaceStorage/f91357654d280e3ba43e8b7238e5a9c4/redhat.java/jdt_ws/SEG2105-Assignment1_6356afc4/bin design5.design2Test
7.60367422
hazzemsukar@Hazzems-MacBook-Pro SEG2105-Assignment1 %
```



```
design3Test.java U
pointcp > design5 > design3Test.java > ...
```

```java
package design5;

import java.util.Random;

public class design3Test {

    Run | Debug
    public static void main(String[] args){
        Random random = new Random();
        char type = (random.nextBoolean()) ? 'C' : 'P';
        int i = 0;
        long start = System.nanoTime();

        while(i != 45000000){
            PointCP3 pointcp = new PointCP3(type, random.nextDouble(), random.nextDouble());
            pointcp.convertStorageToCartesian();
            pointcp.convertStorageToPolar();
            pointcp.getDistance(pointcp);
            pointcp.rotatePoint(random.nextDouble());
            i++;

        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println(timeInSeconds);


    }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

hazzemsukar@Hazzems-MacBook-Pro SEG2105-Assignment1 %  /usr/bin/env /Users/hazzemsukar/Library/Java/JavaVirtualMachines/openjdk-15.0.1/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/hazzemsukar/Library/Application\ Support/Code/User/workspaceStorage/f91357654d280e3ba43e8b7238e5a9c4/redhat.java/jdt_ws/SEG2105-Assignment1_6356afc4/bin design5.design3Test
6.808485329
hazzemsukar@Hazzems-MacBook-Pro SEG2105-Assignment1 %
```

a) Adding to arrays

After doing some testing, we deduced that calling the add function 400 000 000 (400 million) times would result in a near 10 second runtime (in between 0.3 and 0.9 seconds off).

For the first test, we created an Array of size 400 million, while we let the Vector and ArrayList grow dynamically. Then we proceeded to add 400 million elements to each array.

Arrays	Run 1	Run 2	Run 3	Run 4
Array	5.56 seconds	5.73 seconds	5.65 seconds	6.69 seconds
ArrayList	11.29 seconds	10.15 seconds	9.67 seconds	9.90 seconds
Vector	N/A	N/A	N/A	N/A

See note at the end of a) for information on vector runtime.

The average runtime for each array
Array: 5.91 seconds.

ArrayList: 10.25 seconds.

Vector: N/A (the vector ran into an out of memory error).

For the second test, we created an Array, Vector and ArrayList of size 400 million. Then we proceeded to add 400 million elements to each array.

Arrays	Run 1	Run 2	Run 3	Run 4
Array	5.56 seconds	5.73 seconds	5.65 seconds	6.69 seconds
ArrayList	6.30 seconds	6.55 seconds	6.27 seconds	6.73 seconds
Vector	N/A	N/A	N/A	N/A

See note at the end of a) for information on vector runtime.

The average runtime for each of the arrays
Array: 5.91 seconds.

ArrayList: 6.46 seconds.

Vector: N/A (the vector ran into an out of memory error).

Note: We came to the conclusion that even if the size was feasible for the vector array, it would still consistently run slower than the 2 other arrays.

b) Sum of elements in arrays

For this experiment, we decided to use a size of 200 million, since the vector array would not throw an error if we kept it at this size.

For this test, we created an Array, Vector and ArrayList of size 200 million (runtime for creating the arrays was negligible). We then used a for loop to get the sum of the Array, whilst opting for iterators to get the sum of the elements in Vector and ArrayList.

Arrays	Run 1	Run 2	Run 3	Run 4	Run 5
Array	0.087 seconds	0.087 seconds	0.087 seconds	0.096 seconds	0.086 seconds
ArrayList	0.215 seconds	0.209 seconds	0.216 seconds	0.220 seconds	0.250 seconds
Vector	4.880 seconds	4.662 seconds	4.850 seconds	5.100seconds	4.960 seconds

The average runtime for each of the arrays
Array: 0.089 seconds.

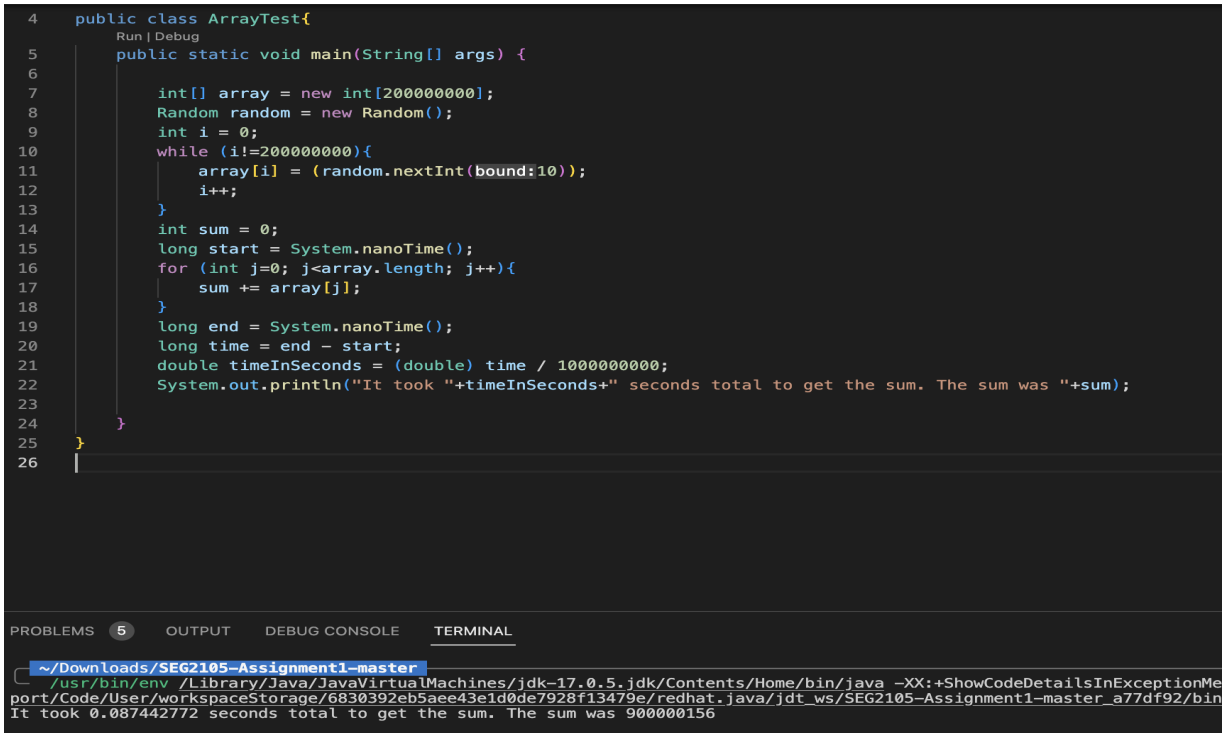
ArrayList: 0.222 seconds.

Vector: 4.89 seconds.

Conclusion: When testing the run time for the different types of collections, we noticed that vector collection, no matter what the operation was, would always run considerably slower than the other 2 collections. The reason for this is because the vector class is synchronized, which increases runtime all together. As for the other 2 collections, we deduced that a regular array would be more efficient in both adding elements to the array and adding all the elements of the array together to get the sum. Although the array is faster than arrayList, the size needs to be predefined, whereas the size of arrayList is dynamic, and will expand when it needs to. In essence, if you have a fixed amount of elements that you know will not

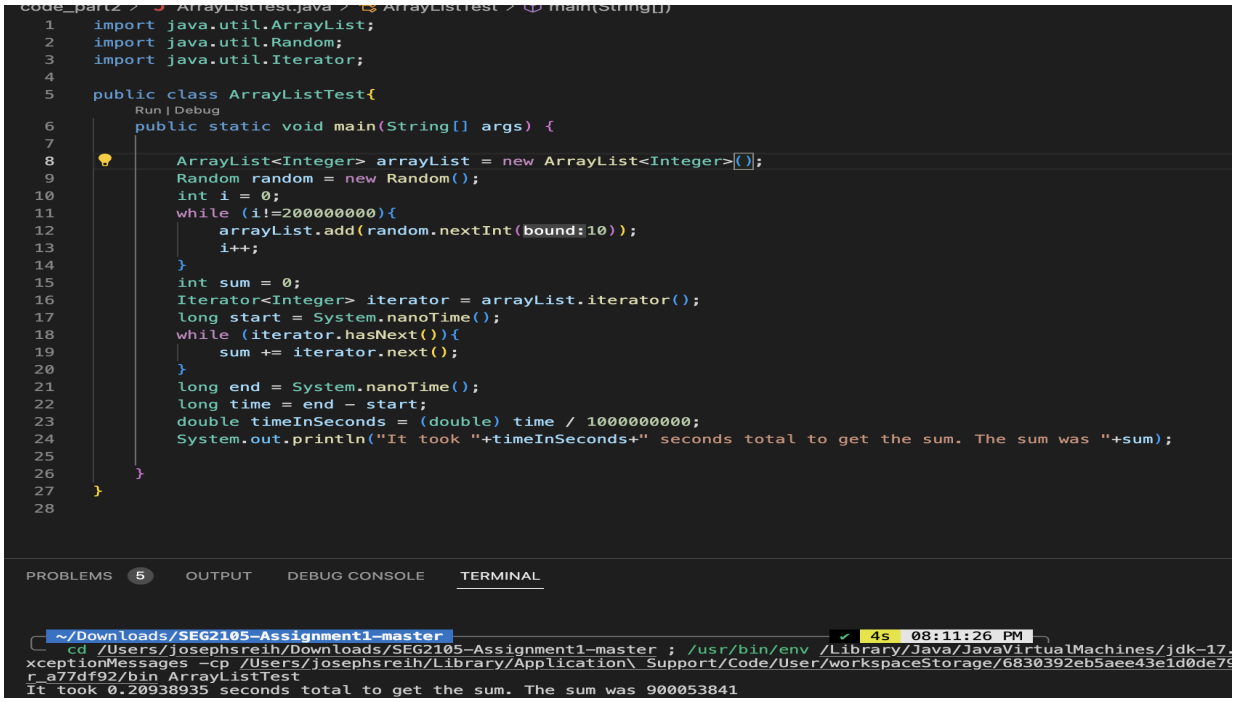
expand, then using arrays would be wiser. However, if the array is prone to expansion and you can predict the maximum size it will reach, then using arrayList will be a more viable option.

Sample outputs underneath



```
public class ArrayTest{
    Run | Debug
    public static void main(String[] args) {

        int[] array = new int[200000000];
        Random random = new Random();
        int i = 0;
        while (i!=200000000){
            array[i] = (random.nextInt(bound:10));
            i++;
        }
        int sum = 0;
        long start = System.nanoTime();
        for (int j=0; j<array.length; j++){
            sum += array[j];
        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println("It took "+timeInSeconds+" seconds total to get the sum. The sum was "+sum);

    }
}
```

```
PROBLEMS 5  OUTPUT  DEBUG CONSOLE  TERMINAL

~/Downloads/SEG2105-Assignment1-master
  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMe
port/Code/User/workspaceStorage/6830392eb5aee43e1d0de7928f13479e/redhat.java/jdt_ws/SEG2105-Assignment1-master_a77df92/bin
It took 0.087442772 seconds total to get the sum. The sum was 900000156
```



```
import java.util.ArrayList;
import java.util.Random;
import java.util.Iterator;

public class ArrayListTest{
    Run | Debug
    public static void main(String[] args) {

        ArrayList<Integer> arrayList = new ArrayList<Integer>();
        Random random = new Random();
        int i = 0;
        while (i!=200000000){
            arrayList.add(random.nextInt(bound:10));
            i++;
        }
        int sum = 0;
        Iterator<Integer> iterator = arrayList.iterator();
        long start = System.nanoTime();
        while (iterator.hasNext()){
            sum += iterator.next();
        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println("It took "+timeInSeconds+" seconds total to get the sum. The sum was "+sum);

    }
}
```

```
PROBLEMS 5  OUTPUT  DEBUG CONSOLE  TERMINAL

~/Downloads/SEG2105-Assignment1-master  4s  08:11:26 PM
  cd /Users/josephsreih/Downloads/SEG2105-Assignment1-master ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-17.
xceptionMessages -cp /Users/josephsreih/Library/Application\ Support/Code/User/workspaceStorage/6830392eb5aee43e1d0de79
r_a77df92/bin ArrayListTest
It took 0.20938935 seconds total to get the sum. The sum was 900053841
```

code_part2 > VectorTest.java > VectorTest > main(String[])

```java
import java.util.Vector;
import java.util.Random;
import java.util.Iterator;

public class VectorTest{
    Run | Debug
    public static void main(String[] args) {

        Vector<Integer> vector = new Vector<Integer>();
        Random random = new Random();
        int i = 0;
        while (i!=200000000){
            vector.add(random.nextInt(bound:10));
            i++;
        }
        int sum = 0;
        Iterator<Integer> iterator = vector.iterator();
        long start = System.nanoTime();
        while (iterator.hasNext()){
            sum += iterator.next();
        }
        long end = System.nanoTime();
        long time = end - start;
        double timeInSeconds = (double) time / 1000000000;
        System.out.println("It took "+timeInSeconds+" seconds total to get the sum. The sum was "+sum);

    }
}
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

```
~/Downloads/SEG2105-Assignment1-master
  /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptio
port/Code/User/workspaceStorage/6830392eb5aee43e1d0de7928f13479e/redhat.java/jdt_ws/SEG2105-Assignment1-master_a77df92/
It took 4.662093665 seconds total to get the sum. The sum was 900008859
```