**Task 2**

Text

Description automatically generated

**Task 3**

import java.util.LinkedList;

public class LinkedListExperiments {

public static LinkedList<Double> initialize(int size) {

// A LinkedList of random doubles

LinkedList<Double> d = new LinkedList<Double>();

for (int i = 0; i < size; i++) {

d.addLast(StdRandom.uniform()); // add double into the LinkedList.

}

return d;

}

public static long retrieve\_by\_index(LinkedList<Double> d) {

int i = (int) (StdRandom.uniform() \* d.size()); // get a random index within the list bound.

long startTime = System.nanoTime();

double r = d.get(i);

return (System.nanoTime() - startTime);

}

public static long retrieve\_by\_element(LinkedList<Double> d) {

int i = (int) (StdRandom.uniform() \* d.size());

double r = d.get(i);

long startTime = System.nanoTime();

StdOut.println("index: " + d.indexOf(r));

return (System.nanoTime() - startTime);

}

public static void main(String[] args) {

int size = Integer.parseInt(args[0]);

long trials = Long.parseLong(args[1]);

LinkedList<Double> d = initialize(size);

/\* System.out.println("Timing retrieval by index...");

long index\_retrieval = 0;

for (long t = 0; t < trials; t++) {

index\_retrieval += retrieve\_by\_index(d);

}

System.out.println(

"Retrieval by index: " + (index\_retrieval / ((double) trials))

+ " nanoseconds on average");

System.out.println("Sorting array...");

d.sort(Comparator.naturalOrder());

\*/

System.out.println("Timing retrieval by element...");

long element\_retrieval = 0;

for (long t = 0; t < trials; t++) {

element\_retrieval += retrieve\_by\_element(d);

}

System.out.println(

"Retrieval by element: " + element\_retrieval / ((double) trials)

+ " nanoseconds on average");

}

}

**Task 4**

Text, table

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

A picture containing text

Description automatically generated