```
import java.util.LinkedList;
import java.util.List;
import java.util.Random;

/**
 * EarlyExit.java
 * Illustrates the average case running time difference
 * between a linear search with an "early exit" and one
 * without.
 *
 * Note: Average case is defined as a search for a list
 * element that is at an arbitrary list location, other
 * than the first or last.
 *
 */
public class EarlyExit {
    // size of the list to be searched
    private static final int N = 1_000_000;

    // number of timing runs
    private static final int NUM_RUNS = 100;

    // 1.0E9 ns per second
    private static final double SECONDS = 1_000_000_000d;

    /** Drives execution. */
    public static void main(String[] args) {

      // generate timing data for "average case" searches
        long start;
        long elapsedTime;
        double avgTimeA = 0d;
        double avgTimeB = 0d;
        Integer target = 42;

        for (int i = 0; i < NUM_RUNS; i++) {
            List<Integer> list = getRandomListWithSeed(N, 42);

            start = System.nanoTime();
            EarlyExit.searchA(list, target);
            elapsedTime = System.nanoTime() - start;
            avgTimeA += elapsedTime;

            start = System.nanoTime();
            EarlyExit.searchB(list, target);
            elapsedTime = System.nanoTime() - start;
            avgTimeB += elapsedTime;
        }

        avgTimeA = (avgTimeA / NUM_RUNS) / SECONDS;
        avgTimeB = (avgTimeB / NUM_RUNS) / SECONDS;

        System.out.printf("%s%4.3f%s\n","Without early exit: ", avgTimeA, " seconds");
        System.out.printf("%s%4.3f%s\n","With early exit:    ", avgTimeB, " seconds");
    }

    /** A linear search that exits only after scanning the entire list. */
    private static <T> boolean searchA(List<T> list, T target) {
        boolean found = false;
        for (T element : list) {
            if (element.equals(target)) {
                found = true;
            }
        }
        return found;
    }

    /** A linear search that exists as soon as the result of the search is known. */
```

```
    private static <T> boolean searchB(List<T> list, T target) {
        for (T element : list) {
            if (element.equals(target)) {
                return true;
            }
        }
        return false;
    }

    /**
     * Fills a list with n random integer values, then places the specified
     * seed value at a random index i, 0 < i < n.
     *
     * @param  n    number of random values to put in the list
     * @param  seed a specific value to ensure that is in the list
     * @return      a list with n - 1 random values plus seed
     */
    private static List<Integer> getRandomListWithSeed(int n, int seed) {
        Random rng = new Random();
        List<Integer> list = new LinkedList<Integer>();
        for (int i = 0; i < n; i++) {
            list.add(i, Integer.valueOf(rng.nextInt()));
        }
        int index = rng.nextInt(n - 1) + 1;
        list.set(index, seed);
        return list;
    }

}
```