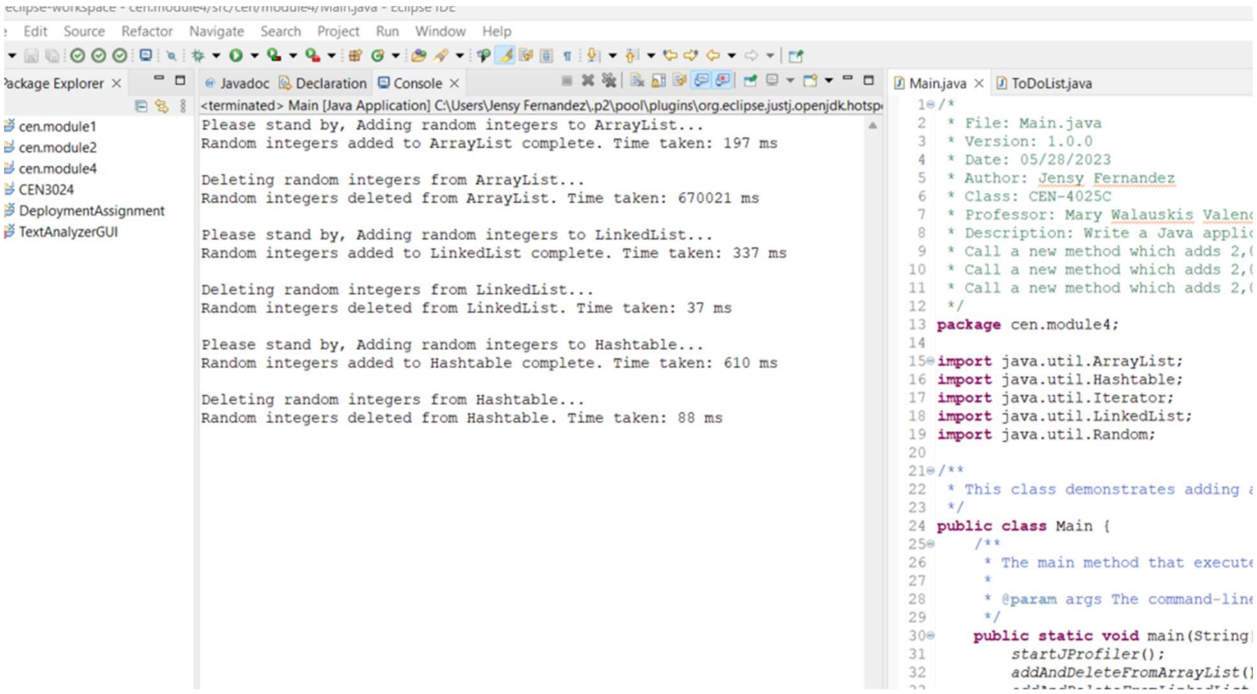
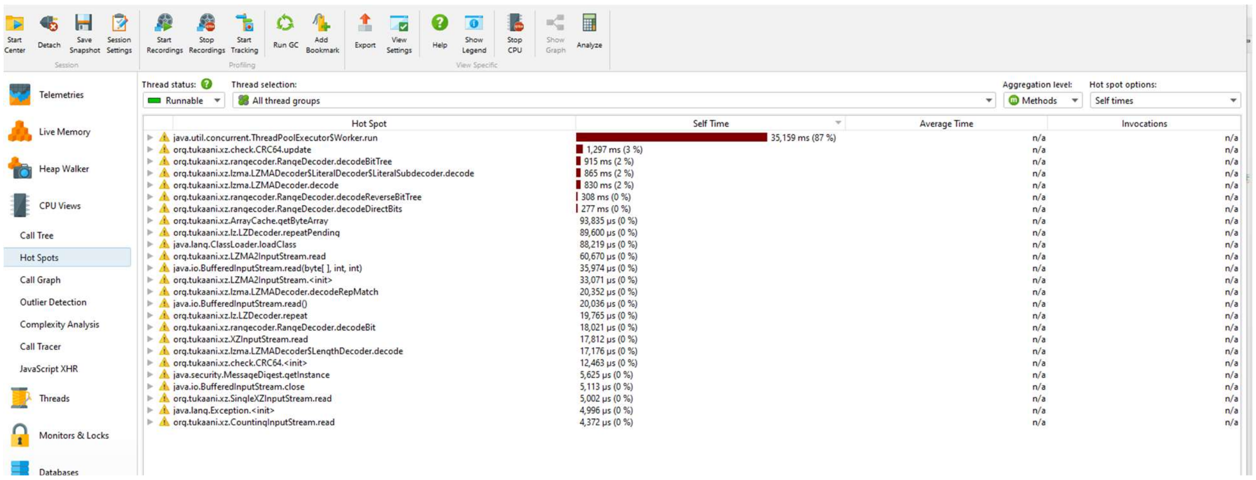
```
/*
 * File: Main.java
 * Version: 1.0.0
 * Date: 05/28/2023
 * Author: Jensy Fernandez
 * Class: CEN-4025C
 * Professor: Mary Walauskis Valencia College
 * Description: Write a Java application that does the following. The Main
method should:
 * Call a new method which adds 2,000,000 random integers into an ArrayList,
then deletes each one from the ArrayList
 * Call a new method which adds 2,000,000 random integers into a LinkedList,
then deletes each one from the LinkedList
 * Call a new method which adds 2,000,000 random integers into a Hashtable,
then deletes each one from the Hashtable
 */
package cen.module4;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Random;

/**
 * This class demonstrates adding and deleting random integers from
JProfiler, ArrayList, LinkedList, and Hashtable.
 */
public class Main {
    /**
     * The main method that executes the program.
     *
     * @param args The command-line arguments.
     */
    public static void main(String[] args) {
      startJProfiler();
      addAndDeleteFromArrayList();
        addAndDeleteFromLinkedList();
        addAndDeleteFromHashtable();
    }

    /**
     * Starts JProfiler by attaching the JProfiler agent to the Java
application.
     */
    public static void startJProfiler() {
        // Specify the JProfiler agent path on my computer
        String agentPath = "C:\\ProgramData\\Microsoft\\Windows\\Start
Menu\\Programs\\JProfiler 13";

        // Specify the JProfiler port this is the most common as per net
        int port = 8849;

        // Start JProfiler
        System.setProperty("java.library.path", agentPath);
        System.setProperty("agentPath", agentPath);
        System.setProperty("jdk.attach.allowAttachSelf", "true");
        System.setProperty("JProfiler.sessionId", "your_session_id");
        System.setProperty("JProfiler.home", agentPath);
        System.setProperty("JProfiler.agentPort", Integer.toString(port));
    }

    /**
     * Adds 2,000,000 random integers to an ArrayList and then deletes each
one from it.
     */
    public static void addAndDeleteFromArrayList() {
        ArrayList<Integer> arrayList = new ArrayList<>();
        Random random = new Random();

        System.out.println("Please stand by, Adding random integers to
ArrayList...");
        long startTime = System.currentTimeMillis();

        // Adding 2,000,000 random integers into the ArrayList
        for (int i = 0; i < 2000000; i++) {
            int randomNumber = random.nextInt();
            arrayList.add(randomNumber);
        }

        long endTime = System.currentTimeMillis();
        System.out.println("Random integers added to ArrayList complete. Time
taken: " + (endTime - startTime) + " ms");

        System.out.println("\nDeleting random integers from ArrayList...");
        startTime = System.currentTimeMillis();

        // Deleting each integer from the ArrayList using an iterator
        Iterator<Integer> iterator = arrayList.iterator();
        while (iterator.hasNext()) {
            iterator.next();
            iterator.remove();
        }

        endTime = System.currentTimeMillis();
        System.out.println("Random integers deleted from ArrayList. Time
taken: " + (endTime - startTime) + " ms");
    }

    /**
     * Adds 2,000,000 random integers to a LinkedList and then deletes each
one from it.
     */
    public static void addAndDeleteFromLinkedList() {
        LinkedList<Integer> linkedList = new LinkedList<>();
        Random random = new Random();

        System.out.println("\nPlease stand by, Adding random integers to
LinkedList...");
        long startTime = System.currentTimeMillis();

        // Adding 2,000,000 random integers into the LinkedList
        for (int i = 0; i < 2000000; i++) {
            int randomNumber = random.nextInt();
            linkedList.add(randomNumber);
        }

        long endTime = System.currentTimeMillis();
        System.out.println("Random integers added to LinkedList complete.
Time taken: " + (endTime - startTime) + " ms");

        System.out.println("\nDeleting random integers from LinkedList...");
        startTime = System.currentTimeMillis();

        // Deleting each integer from the LinkedList using an iterator
        Iterator<Integer> iterator = linkedList.iterator();
        while (iterator.hasNext()) {
            iterator.next();
            iterator.remove();
        }

        endTime = System.currentTimeMillis();
        System.out.println("Random integers deleted from LinkedList. Time
taken: " + (endTime - startTime) + " ms");
    }

    /**
     * Adds 2,000,000 random integers to a Hashtable and then deletes each
one from it.
     */
    public static void addAndDeleteFromHashtable() {
        Hashtable<Integer, Integer> hashtable = new Hashtable<>();
        Random random = new Random();

        System.out.println("\nPlease stand by, Adding random integers to
Hashtable...");
        long startTime = System.currentTimeMillis();

        // Adding 2,000,000 random integers into the Hashtable
        for (int i = 0; i < 2000000; i++) {
            int randomNumber = random.nextInt();
            hashtable.put(randomNumber, randomNumber);
        }

        long endTime = System.currentTimeMillis();
        System.out.println("Random integers added to Hashtable complete. Time
taken: " + (endTime - startTime) + " ms");

        System.out.println("\nDeleting random integers from Hashtable...");
        startTime = System.currentTimeMillis();

        // Deleting each integer from the Hashtable using an iterator
        Iterator<Integer> iterator = hashtable.keySet().iterator();
        while (iterator.hasNext()) {
            iterator.next();
            iterator.remove();
        }

        endTime = System.currentTimeMillis();
        System.out.println("Random integers deleted from Hashtable. Time
taken: " + (endTime - startTime) + " ms");
```

```
        }
}
```



```
<terminated> Main [Java Application] C:\Users\Jensy Fernandez\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
Please stand by, Adding random integers to ArrayList...
Random integers added to ArrayList complete. Time taken: 197 ms

Deleting random integers from ArrayList...
Random integers deleted from ArrayList. Time taken: 670021 ms

Please stand by, Adding random integers to LinkedList...
Random integers added to LinkedList complete. Time taken: 337 ms

Deleting random integers from LinkedList...
Random integers deleted from LinkedList. Time taken: 37 ms

Please stand by, Adding random integers to Hashtable...
Random integers added to Hashtable complete. Time taken: 610 ms

Deleting random integers from Hashtable...
Random integers deleted from Hashtable. Time taken: 88 ms
```

```java
/*
 * File: Main.java
 * Version: 1.0.0
 * Date: 05/28/2023
 * Author: Jensy Fernandez
 * Class: CEN-4025C
 * Professor: Mary Walauskis Valen
 * Description: Write a Java applic
 * Call a new method which adds 2,
 * Call a new method which adds 2,
 * Call a new method which adds 2,
 */
package cen.module4;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Random;

/**
 * This class demonstrates adding
 */
public class Main {
    /**
     * The main method that execute
     *
     * @param args The command-line
     */
    public static void main(String
        startJProfiler();
        addAndDeleteFromArrayList(
```



Thread status: Runnable
Thread selection: All thread groups
Aggregation level: Methods
Hot spot options: Self times

Hot Spot	Self Time	Average Time	Invocations
java.util.concurrent.ThreadPoolExecutor$Worker.run	35,159 ms (87 %)	n/a	n/a
org.tukaani.xz.check.CRC64.update	1,297 ms (3 %)	n/a	n/a
org.tukaani.xz.rangecoder.RangeDecoder.decodeBitTree	915 ms (2 %)	n/a	n/a
org.tukaani.xz.lzma.LZMADecoder$LiteralDecoder$LiteralSubdecoder.decode	865 ms (2 %)	n/a	n/a
org.tukaani.xz.lzma.LZMADecoder.decode	830 ms (2 %)	n/a	n/a
org.tukaani.xz.rangecoder.RangeDecoder.decodeReverseBitTree	308 ms (0 %)	n/a	n/a
org.tukaani.xz.rangecoder.RangeDecoder.decodeDirectBits	277 ms (0 %)	n/a	n/a
org.tukaani.xz.ArrayCache.getByteArray	93,835 µs (0 %)	n/a	n/a
org.tukaani.xz.lz.LZDecoder.repeatPending	89,600 µs (0 %)	n/a	n/a
java.lang.ClassLoader.loadClass	88,219 µs (0 %)	n/a	n/a
org.tukaani.xz.LZMA2InputStream.read	60,670 µs (0 %)	n/a	n/a
java.io.BufferedInputStream.read(byte[ ], int, int)	35,974 µs (0 %)	n/a	n/a
org.tukaani.xz.LZMA2InputStream.<init>	33,071 µs (0 %)	n/a	n/a
org.tukaani.xz.lzma.LZMADecoder.decodeRepMatch	20,352 µs (0 %)	n/a	n/a
java.io.BufferedInputStream.read()	20,036 µs (0 %)	n/a	n/a
org.tukaani.xz.lz.LZDecoder.repeat	19,765 µs (0 %)	n/a	n/a
org.tukaani.xz.rangecoder.RangeDecoder.decodeBit	18,021 µs (0 %)	n/a	n/a
org.tukaani.xz.XZInputStream.read	17,812 µs (0 %)	n/a	n/a
org.tukaani.xz.lzma.LZMADecoder$LengthDecoder.decode	17,176 µs (0 %)	n/a	n/a
org.tukaani.xz.check.CRC64.<init>	12,463 µs (0 %)	n/a	n/a
java.security.MessageDigest.getInstance	5,625 µs (0 %)	n/a	n/a
java.io.BufferedInputStream.close	5,113 µs (0 %)	n/a	n/a
org.tukaani.xz.SingleXZInputStream.read	5,002 µs (0 %)	n/a	n/a
java.lang.Exception.<init>	4,996 µs (0 %)	n/a	n/a
org.tukaani.xz.CountingInputStream.read	4,372 µs (0 %)	n/a	n/a