

## Part1:

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows tabs for "CustomLinkedList.java", "Node.java", "little\_women.txt", "DuplicateWords.java" (which is the active tab), and "Url".
- Code Editor:** Displays the Java code for the "DuplicateWords" class. The code reads a text file ("little\_women.txt") and prints the count of duplicate and single words.

```
1 package sets;
2 import java.util.*;
3 import java.io.*;
4
5 public class DuplicateWords {
6
7     public static String cleanString(String originalString) {
8         String cleanedString = originalString.replaceAll("\\p{Punct}", "");
9         cleanedString = cleanedString.replaceAll("[‘’]+", "");
10        return cleanedString.toLowerCase();
11    }
12
13    public static void main(String[] args) throws IOException {
14        String path = args.length > 0 ? args[0] : "little_women.txt";
15
16        Set<String> once = new HashSet<>();
17        Set<String> dup = new HashSet<>();
18
19        try (BufferedReader br = new BufferedReader(new FileReader(path))) {
20            String line;
21            while ((line = br.readLine()) != null) {
22                String[] words = line.split("\\s+");
23                for (String w : words) {
24                    w = cleanString(w);
25                    if (w.isEmpty()) continue;
26
27                    if (dup.contains(w)) continue;
28                    if (once.contains(w)) {
29                        once.remove(w);
30                        dup.add(w);
31                    } else {
32                        once.add(w);
33                    }
34                }
35            }
36        }
37
38        System.out.println("number of duplicate words: " + dup.size());
39        System.out.println("number of single words: " + once.size());
40    }
41 }
42
```

- Console View:** Shows the output of the application execution:  
number of duplicate words: 253  
number of single words: 382

## Part2:

The screenshot shows the Eclipse IDE interface with the Java code for `UrlLog.java` in the editor. Below the editor, the Eclipse IDE status bar displays the application name, version, and file path. The output console shows the execution results of the application, which prints user recent logs and URL visit counts.

```
CustomLinkedList.java Node.java little_women.txt UrlLog.java X DuplicateWords.java
```

```
line = line.trim();
if (line.isEmpty()) continue;

// skip header line if present
if (first) {
    first = false;
    if (line.toLowerCase(Locale.ROOT).startsWith("timestamp,")) {
        continue;
    }
}

// split CSV: timestamp,userid,url
String[] parts = line.split(",", 3);
if (parts.length < 3) continue;
String timestamp = parts[0].trim(); // not used but kept
String user = parts[1].trim();
String url = parts[2].trim();

// record in userToRecent
ArrayDeque<String> dq = userToRecent.computeIfAbsent(user, k -> new ArrayDeque<>());
if (dq.size() == MAX_RECENT) dq.pollFirst(); // remove oldest
dq.addLast(url);

// count url frequency
urlCount.merge(url, 1, Integer::sum);
} catch (IOException e) {
    System.err.println("Cannot read file: " + path);
    e.printStackTrace();
    return;
}

// --- print user recent logs ---
System.out.println("Recent URLs per user:");
for (Map.Entry<String, ArrayDeque<String>> e : userToRecent.entrySet()) {
    List<String> list = new ArrayList<>(e.getValue());
    System.out.printf("%s -> %s%n", e.getKey(), list);
}

// --- print URL visit counts ---
System.out.println("\nURL Visit Counts:");
List<Map.Entry<String, Integer>> sorted = urlCount.entrySet()
    .stream()
    .sorted(Comparator.comparing(Map.Entry<String, Integer>::getValue, Comparator.reverseOrder()
        .thenComparing(Map.Entry::getKey)))
    .collect(Collectors.toList());

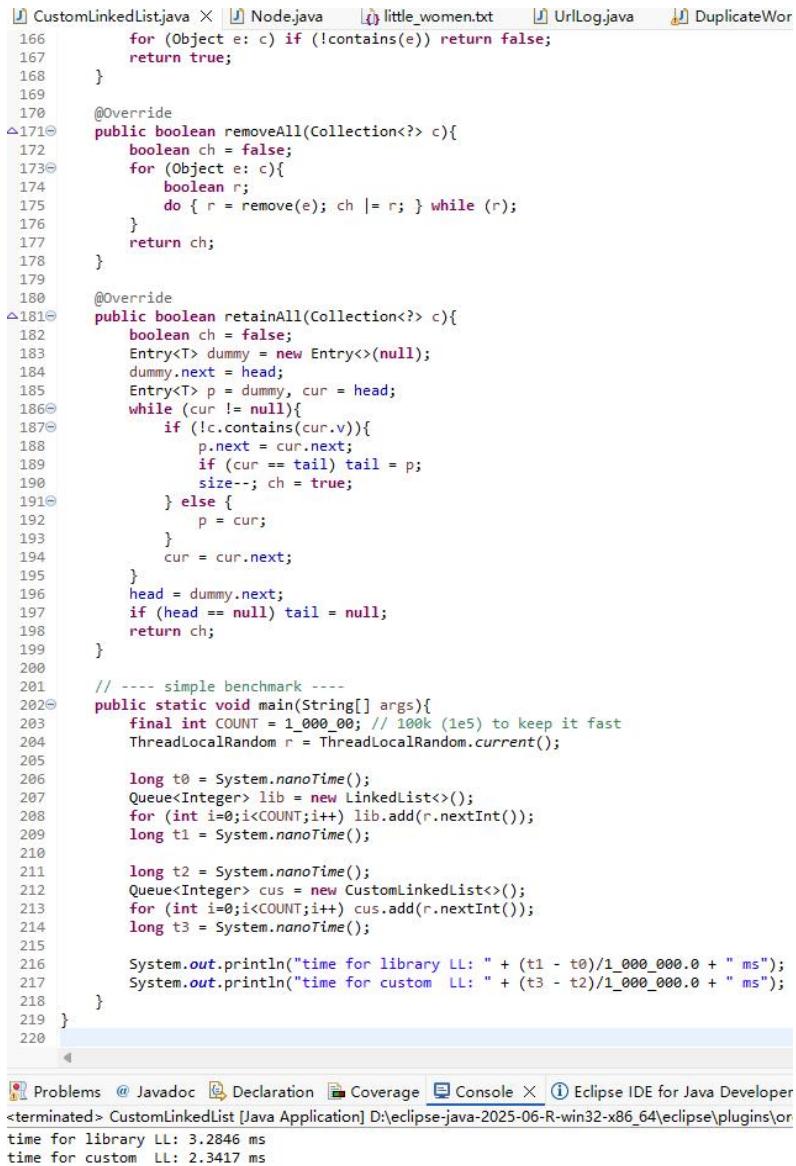
int maxLen = sorted.stream().mapToInt(a -> a.getKey().length()).max().orElse(0);
for (Map.Entry<String, Integer> e : sorted) {
    System.out.printf(" %-" + maxLen + "s : %d%n", e.getKey(), e.getValue());
}
```

```
Problems @ Javadoc Declaration Coverage Console X Eclipse IDE for Java Developers 2025-09 Release
<terminated> UrlLog [Java Application] D:\eclipse-java-2025-06-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot
Recent URLs per user:
u001 -> [/cart, /orders, /logout, /login, /checkout]
u002 -> [/home, /products, /checkout, /orders, /logout]
u003 -> [/checkout, /logout, /login, /home, /products]
u004 -> [/products, /cart, /orders, /logout, /checkout]

URL Visit Counts:
/products : 8
/checkout : 7
/home    : 7
/logout   : 7
/login    : 5
/orders   : 5
/cart     : 4
```

## Part3:

```
CustomLinkedList.java X Node.java little_women.txt UrlLog.java DuplicateWor
166     for (Object e: c) if (!contains(e)) return false;
167     return true;
168 }
169
170 @Override
171 public boolean removeAll(Collection<?> c){
172     boolean ch = false;
173     for (Object e: c){
174         boolean r;
175         do { r = remove(e); ch |= r; } while (r);
176     }
177     return ch;
178 }
179
180 @Override
181 public boolean retainAll(Collection<?> c){
182     boolean ch = false;
183     Entry<T> dummy = new Entry<>(null);
184     dummy.next = head;
185     Entry<T> p = dummy, cur = head;
186     while (cur != null){
187         if (!c.contains(cur.v)){
188             p.next = cur.next;
189             if (cur == tail) tail = p;
190             size--;
191             ch = true;
192         } else {
193             p = cur;
194         }
195         cur = cur.next;
196     }
197     head = dummy.next;
198     if (head == null) tail = null;
199     return ch;
200 }
201 // ---- simple benchmark ----
202 public static void main(String[] args){
203     final int COUNT = 1_000_00; // 100k (1e5) to keep it fast
204     ThreadLocalRandom r = ThreadLocalRandom.current();
205
206     long t0 = System.nanoTime();
207     Queue<Integer> lib = new LinkedList<>();
208     for (int i=0;i<COUNT;i++) lib.add(r.nextInt());
209     long t1 = System.nanoTime();
210
211     long t2 = System.nanoTime();
212     Queue<Integer> cus = new CustomLinkedList<>();
213     for (int i=0;i<COUNT;i++) cus.add(r.nextInt());
214     long t3 = System.nanoTime();
215
216     System.out.println("time for library LL: " + (t1 - t0)/1_000_000.0 + " ms");
217     System.out.println("time for custom LL: " + (t3 - t2)/1_000_000.0 + " ms");
218 }
219 }
220
```



The screenshot shows the Eclipse IDE interface with the Java code for `CustomLinkedList`. The code includes implementations for `removeAll` and `retainAll` methods, and a `main` method for benchmarking. The benchmark results are printed to the console, showing execution times for both the library's `LinkedList` and the custom implementation.

```
Problems @ Javadoc Declaration Coverage Console X Eclipse IDE for Java Developer
<terminated> CustomLinkedList [Java Application] D:\eclipse-java-2025-06-R-win32-x86_64\plugins\or
time for library LL: 2.2846 ms
time for custom LL: 2.3417 ms
```