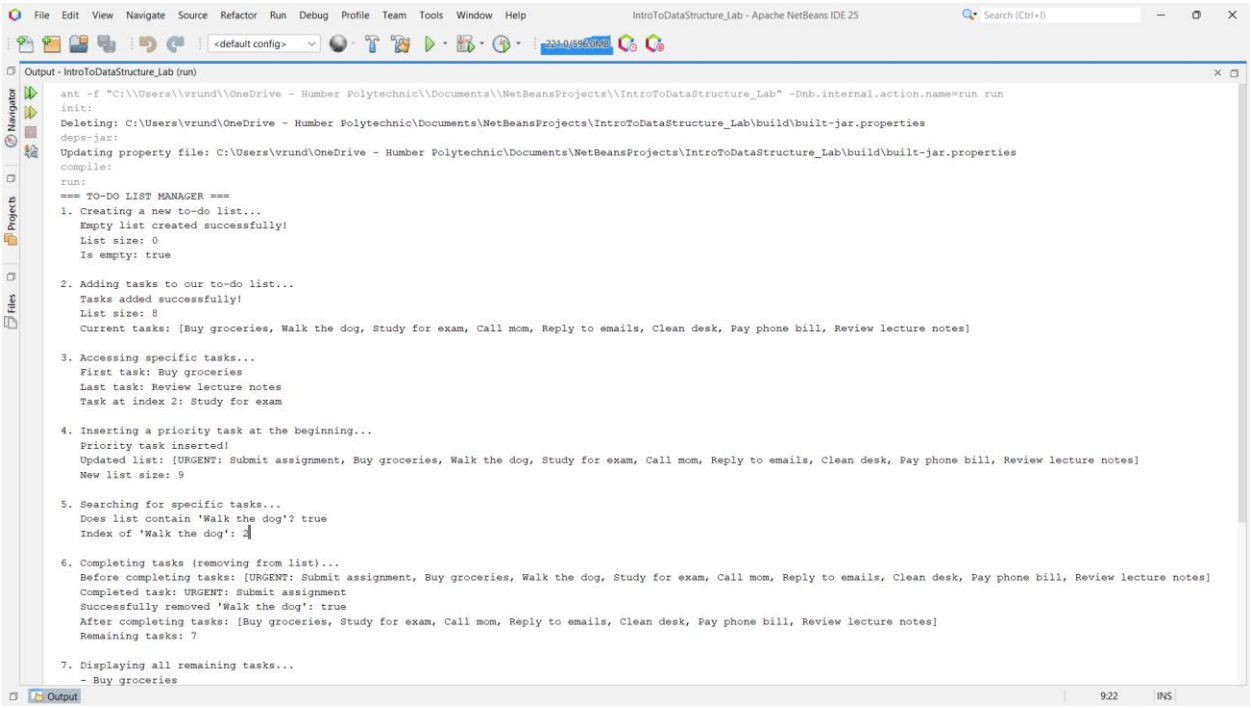
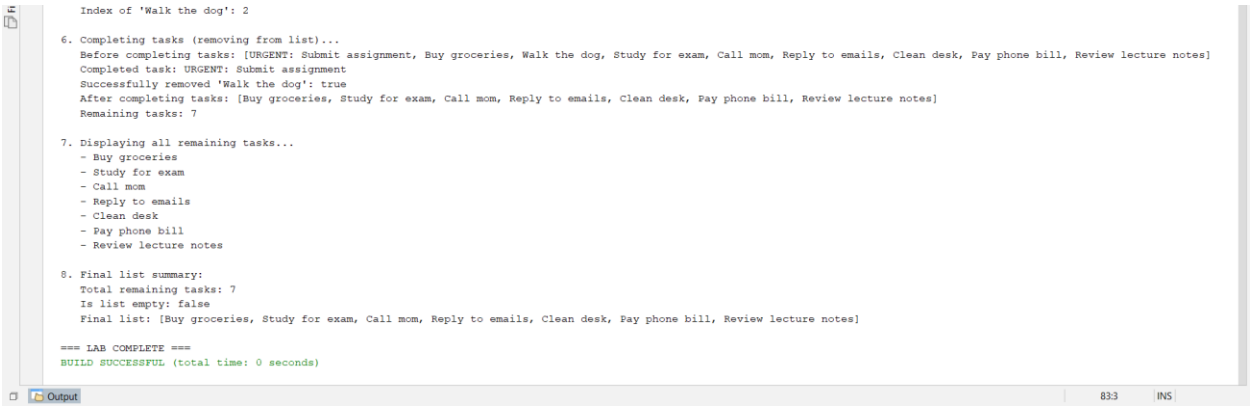
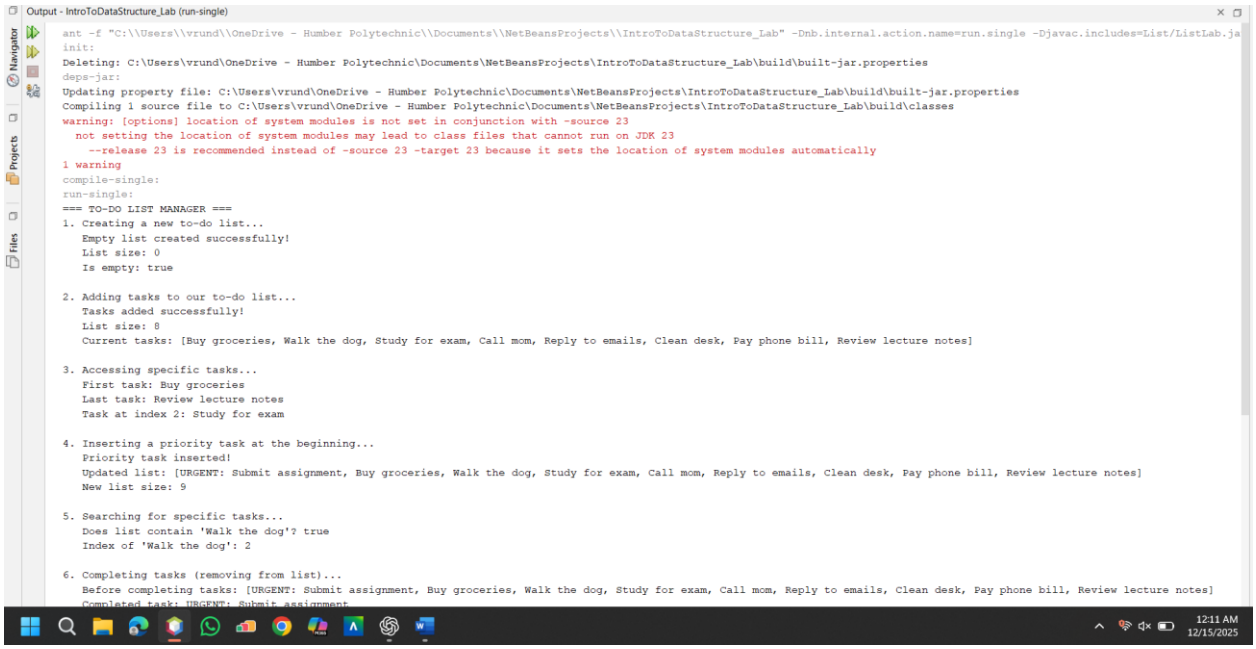
1)

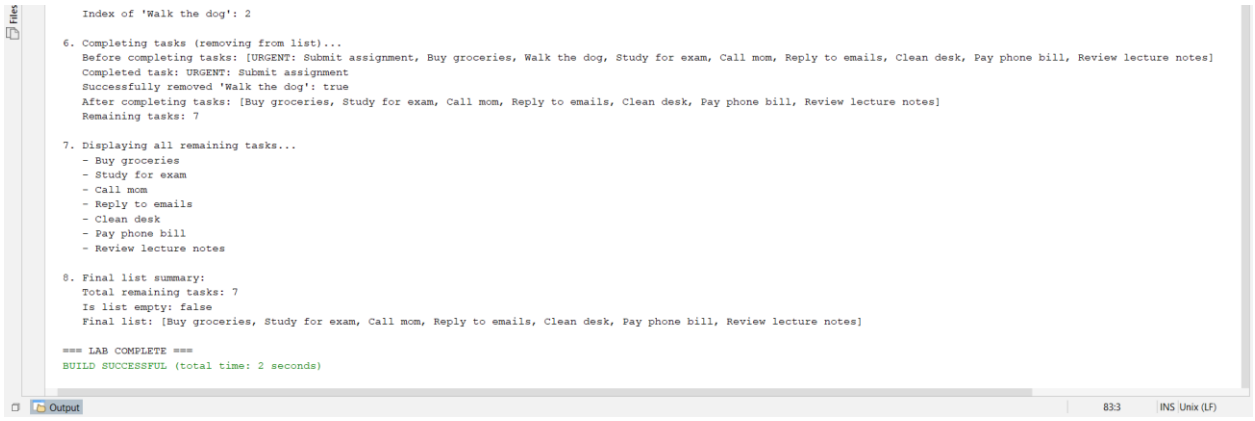

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
IntroToDataStructure_Lab - Apache NetBeans IDE 25
Output - IntroToDataStructure_Lab (run)
ant -f "C:\\Users\\vrund\\OneDrive - Humber Polytechnic\\Documents\\NetBeansProjects\\IntroToDataStructure_Lab" -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
compile:
run:
=== TO-DO LIST MANAGER ===
1. Creating a new to-do list...
Empty list created successfully!
List size: 0
Is empty: true
2. Adding tasks to our to-do list...
Tasks added successfully!
List size: 8
Current tasks: [Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
3. Accessing specific tasks...
First task: Buy groceries
Last task: Review lecture notes
Task at index 2: Study for exam
4. Inserting a priority task at the beginning...
Priority task inserted!
Updated list: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
New list size: 9
5. Searching for specific tasks...
Does list contain 'Walk the dog'? true
Index of 'Walk the dog': 2
6. Completing tasks (removing from list)...
Before completing tasks: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
Completed task: URGENT: Submit assignment
Successfully removed 'Walk the dog': true
After completing tasks: [Buy groceries, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
Remaining tasks: 7
7. Displaying all remaining tasks...
- Buy groceries
Output
9:22
INS


Index of 'Walk the dog': 2
6. Completing tasks (removing from list)...
Before completing tasks: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
Completed task: URGENT: Submit assignment
Successfully removed 'Walk the dog': true
After completing tasks: [Buy groceries, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
Remaining tasks: 7
7. Displaying all remaining tasks...
- Buy groceries
- Study for exam
- Call mom
- Reply to emails
- Clean desk
- Pay phone bill
- Review lecture notes
8. Final list summary:
Total remaining tasks: 7
Is list empty: false
Final list: [Buy groceries, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
=== LAB COMPLETE ===
BUILD SUCCESSFUL (total time: 0 seconds)
Output
83:3
INS

2) ListLab

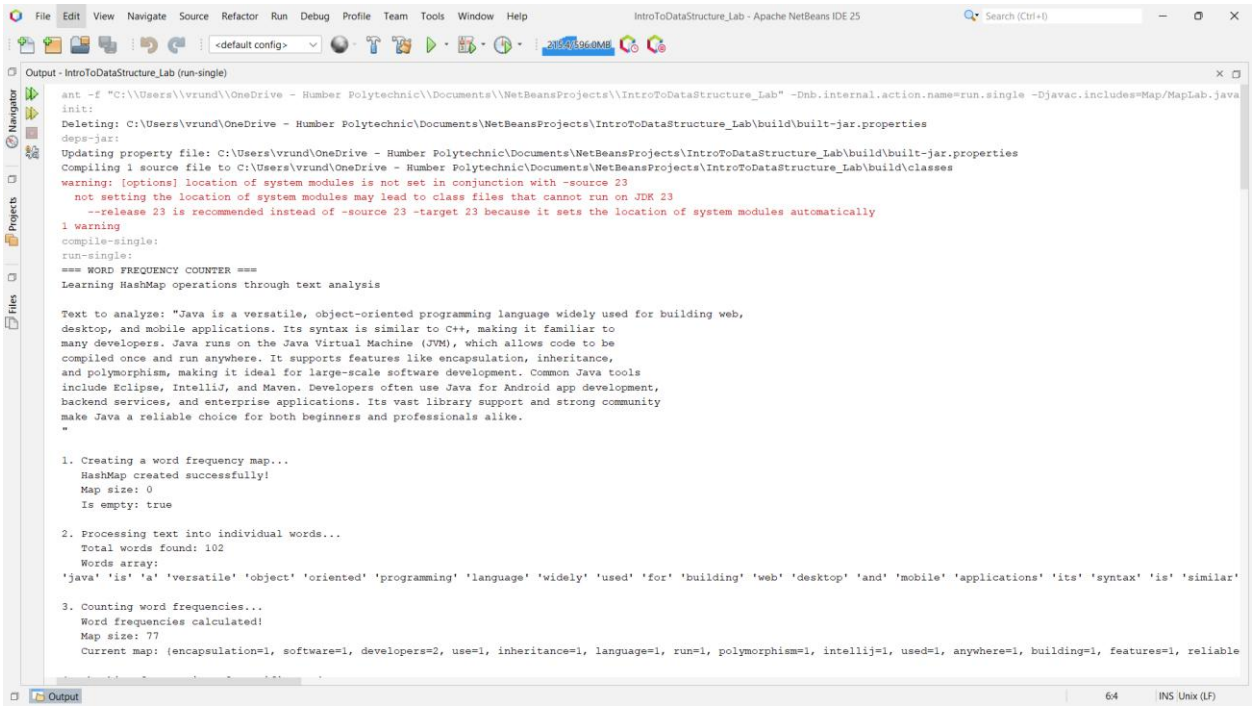


```
Output - IntroToDataStructure_Lab (run-single)
ant -f "C:\\Users\\vrund\\OneDrive - Humber Polytechnic\\Documents\\NetBeansProjects\\IntroToDataStructure_Lab" -Dnb.internal.action.name=run.single -Djavac.includes=List/ListLab.ja
init:
Deleting: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
Compiling 1 source file to C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\classes
warning: [options] location of system modules is not set in conjunction with -source 23
  not setting the location of system modules may lead to class files that cannot run on JDK 23
    --release 23 is recommended instead of -source 23 -target 23 because it sets the location of system modules automatically
1 warning
compile-single:
run-single:
=== TO-DO LIST MANAGER ===
1. Creating a new to-do list...
   Empty list created successfully!
   List size: 0
   Is empty: true

2. Adding tasks to our to-do list...
   Tasks added successfully!
   List size: 8
   Current tasks: [Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]

3. Accessing specific tasks...
   First task: Buy groceries
   Last task: Review lecture notes
   Task at index 2: Study for exam

4. Inserting a priority task at the beginning...
   Priority task inserted!
   Updated list: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
   New list size: 9

5. Searching for specific tasks...
   Does list contain 'Walk the dog'? true
   Index of 'Walk the dog': 2

6. Completing tasks (removing from list)...
   Before completing tasks: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
   Completed task: URGENT: Submit assignment
```

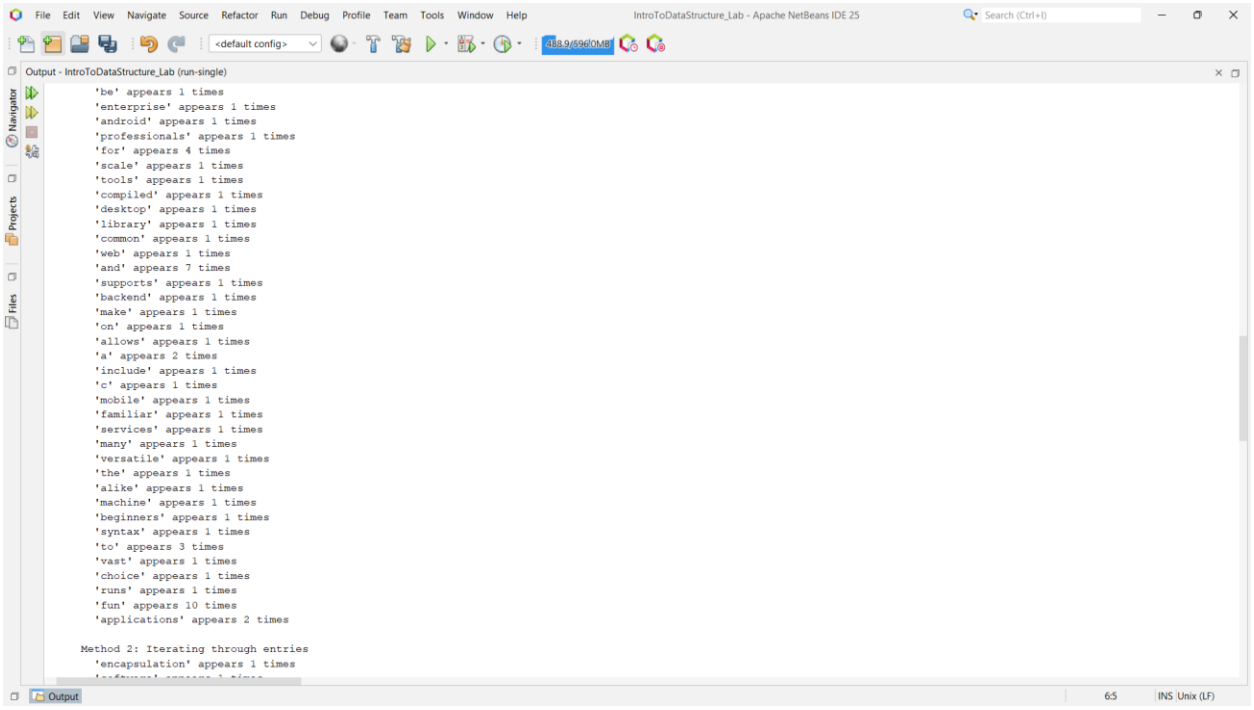


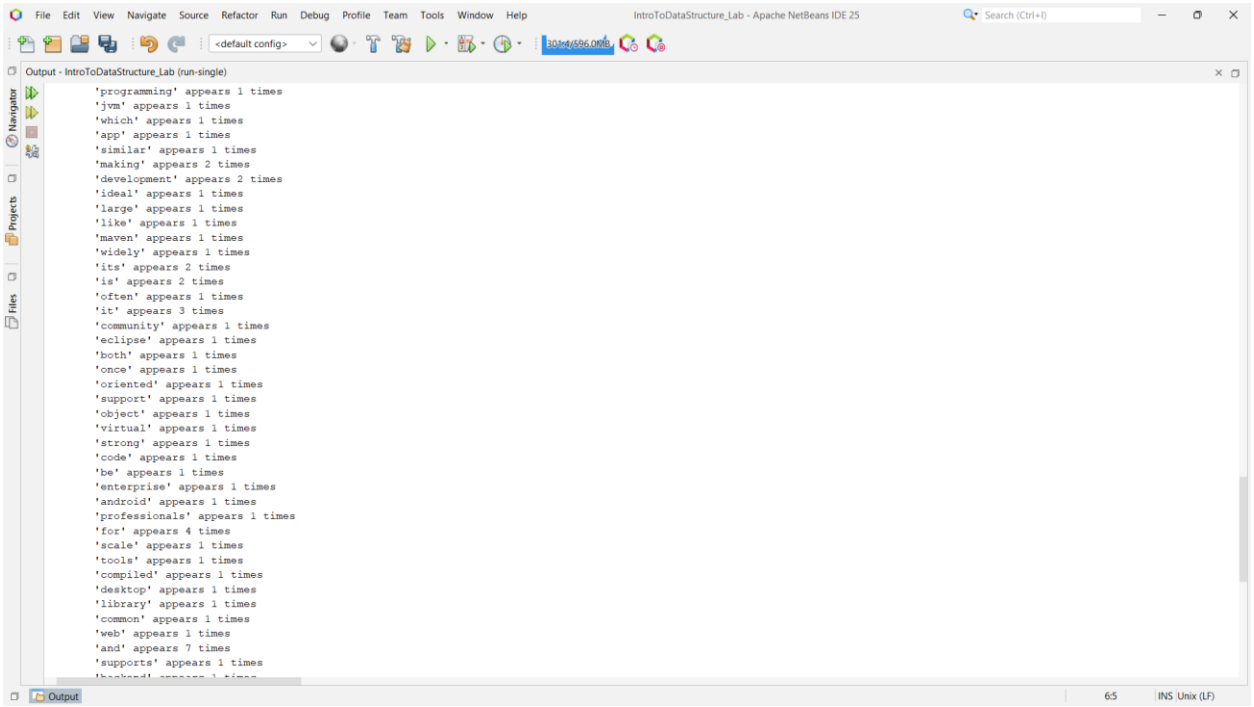
```
   Index of 'Walk the dog': 2

6. Completing tasks (removing from list)...
   Before completing tasks: [URGENT: Submit assignment, Buy groceries, Walk the dog, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
   Completed task: URGENT: Submit assignment
   Successfully removed 'Walk the dog': true
   After completing tasks: [Buy groceries, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]
   Remaining tasks: 7

7. Displaying all remaining tasks...
   - Buy groceries
   - Study for exam
   - Call mom
   - Reply to emails
   - Clean desk
   - Pay phone bill
   - Review lecture notes

8. Final list summary:
   Total remaining tasks: 7
   Is list empty: false
   Final list: [Buy groceries, Study for exam, Call mom, Reply to emails, Clean desk, Pay phone bill, Review lecture notes]

=== LAB COMPLETE ===
BUILD SUCCESSFUL (total time: 2 seconds)
Output
83:3
INS Unix (LF)
```

3) MapLab:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
IntroToDataStructure_Lab - Apache NetBeans IDE 25
Output - IntroToDataStructure_Lab (run-single)
ant -f "C:\\Users\\vrund\\OneDrive - Humber Polytechnic\\Documents\\NetBeansProjects\\IntroToDataStructure_Lab" -Dnb.internal.action.name=run.single -Djavac.includes=Map/MapLab.java
init:
Deleting: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
Compiling 1 source file to C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\classes
warning: [options] location of system modules is not set in conjunction with -source 23
not setting the location of system modules may lead to class files that cannot run on JDK 23
--release 23 is recommended instead of -source 23 -target 23 because it sets the location of system modules automatically
1 warning
compile-single:
run-single:
=== WORD FREQUENCY COUNTER ===
Learning HashMap operations through text analysis
Text to analyze: "Java is a versatile, object-oriented programming language widely used for building web,
desktop, and mobile applications. Its syntax is similar to C++, making it familiar to
many developers. Java runs on the Java Virtual Machine (JVM), which allows code to be
compiled once and run anywhere. It supports features like encapsulation, inheritance,
and polymorphism, making it ideal for large-scale software development. Common Java tools
include Eclipse, IntelliJ, and Maven. Developers often use Java for Android app development,
backend services, and enterprise applications. Its vast library support and strong community
make Java a reliable choice for both beginners and professionals alike.
"
1. Creating a word frequency map...
HashMap created successfully!
Map size: 0
Is empty: true
2. Processing text into individual words...
Total words found: 102
Words array:
'java' 'is' 'a' 'versatile' 'object' 'oriented' 'programming' 'language' 'widely' 'used' 'for' 'building' 'web' 'desktop' 'and' 'mobile' 'applications' 'its' 'syntax' 'is' 'similar'
3. Counting word frequencies...
Word frequencies calculated!
Map size: 77
Current map: {encapsulation=1, software=1, developers=2, use=1, inheritance=1, language=1, run=1, polymorphism=1, intellij=1, used=1, anywhere=1, building=1, features=1, reliable
Output
6:4
INS Unix (LF)

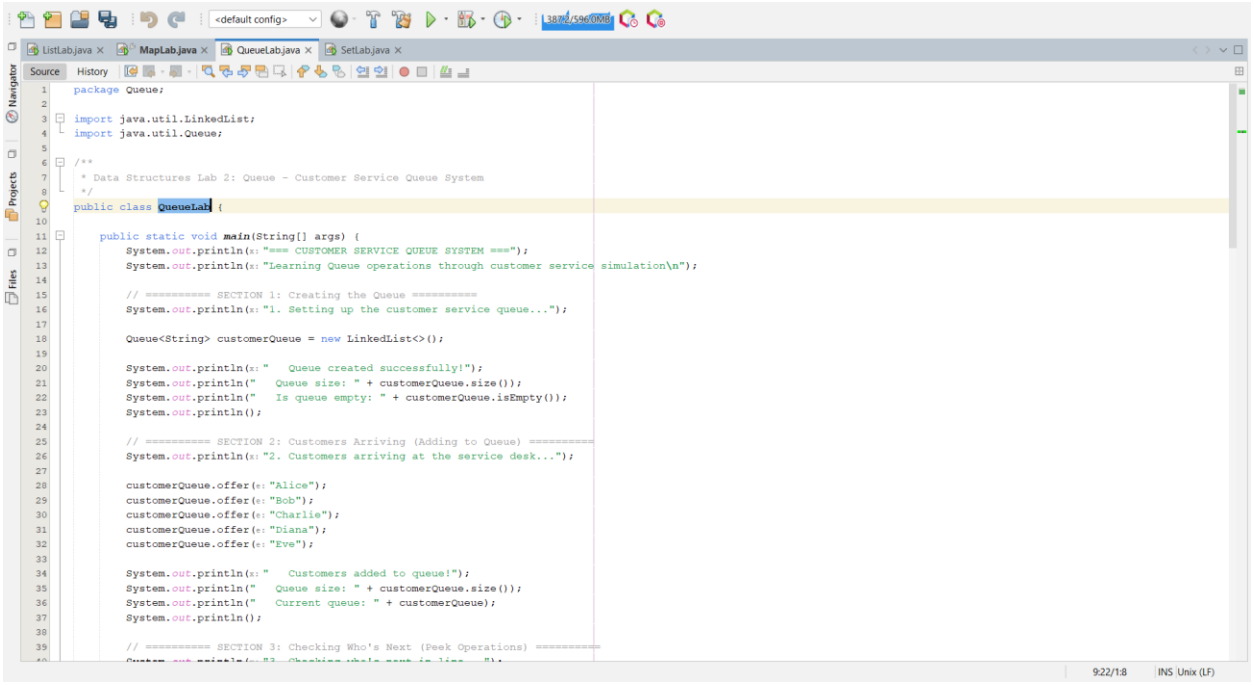

Output - IntroToDataStructure_Lab (run-single)
6. Manually updating a word frequency...
Before update - 'applications' frequency: 2
After update - 'applications' frequency: 2
7. Displaying all word frequencies...
Method 1: Iterating through keys
'encapsulation' appears 1 times
'software' appears 1 times
'developers' appears 2 times
'use' appears 1 times
'inheritance' appears 1 times
'language' appears 1 times
'run' appears 1 times
'polymorphism' appears 1 times
'intellij' appears 1 times
'used' appears 1 times
'anywhere' appears 1 times
'building' appears 1 times
'features' appears 1 times
'reliable' appears 1 times
'java' appears 6 times
'programming' appears 1 times
'jvm' appears 1 times
'which' appears 1 times
'app' appears 1 times
'similar' appears 1 times
'making' appears 2 times
'development' appears 2 times
'ideal' appears 1 times
'large' appears 1 times
'like' appears 1 times
'maven' appears 1 times
'widely' appears 1 times
'its' appears 2 times
'is' appears 2 times
'often' appears 1 times
'it' appears 3 times
'community' appears 1 times
'eclipse' appears 1 times
'both' appears 1 times
Output
6:4
INS Unix (LF)



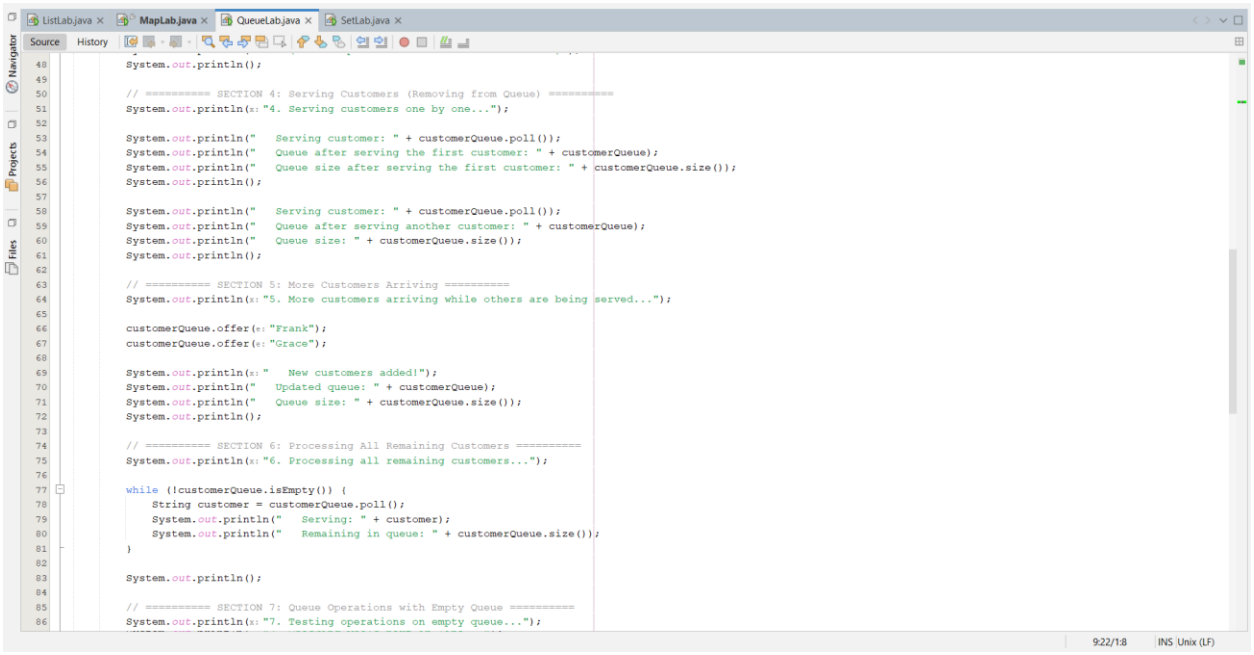



```
8. Finding most and least frequent words...
   Most frequent word: 'fun' (10 times)
   Least frequent word: 'encapsulation' (1 times)

9. Final statistics...
   Total unique words: 78
   All unique words: [encapsulation, software, developers, use, inheritance, language, run, polymorphism, intellij, used, anywhere, building, features, reliable, java, programming,
   All frequency values: [1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1,
   Final map: {encapsulation=1, software=1, developers=2, use=1, inheritance=1, language=1, run=1, polymorphism=1, intellij=1, used=1, anywhere=1, building=1, features=1, reliable=1

=== LAB COMPLETE ===
BUILD SUCCESSFUL (total time: 2 seconds)
```

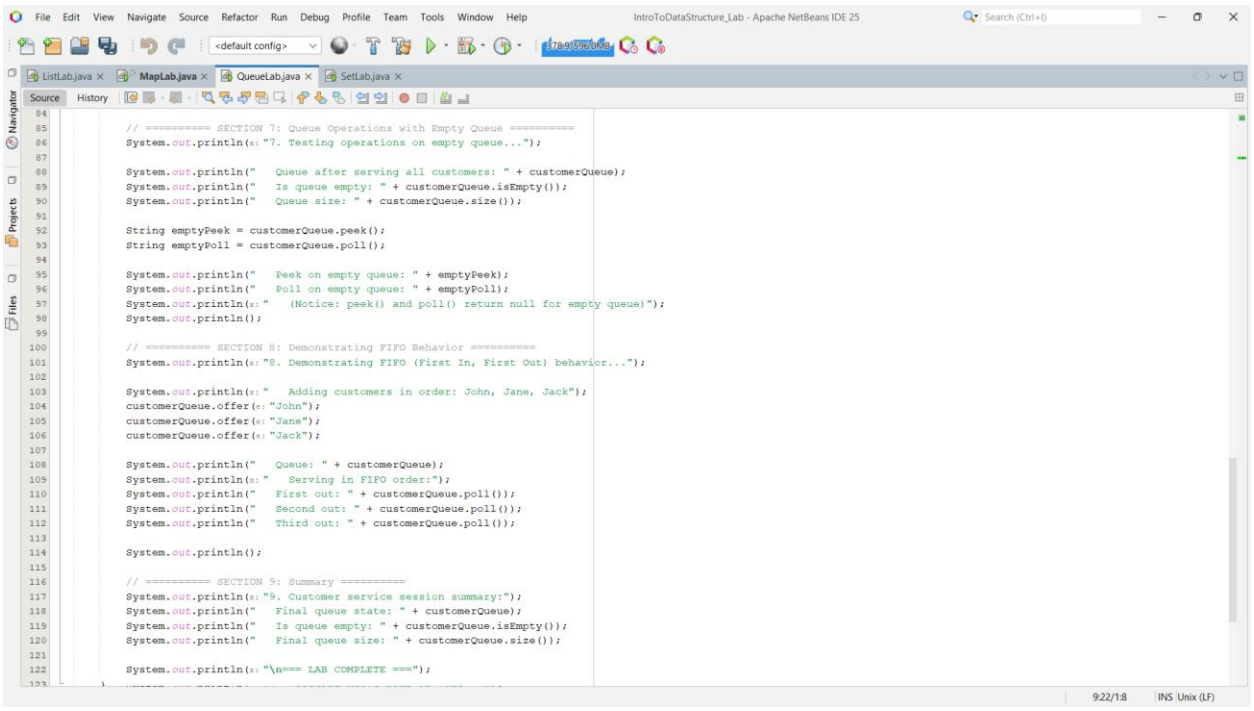
4) QueueLab:



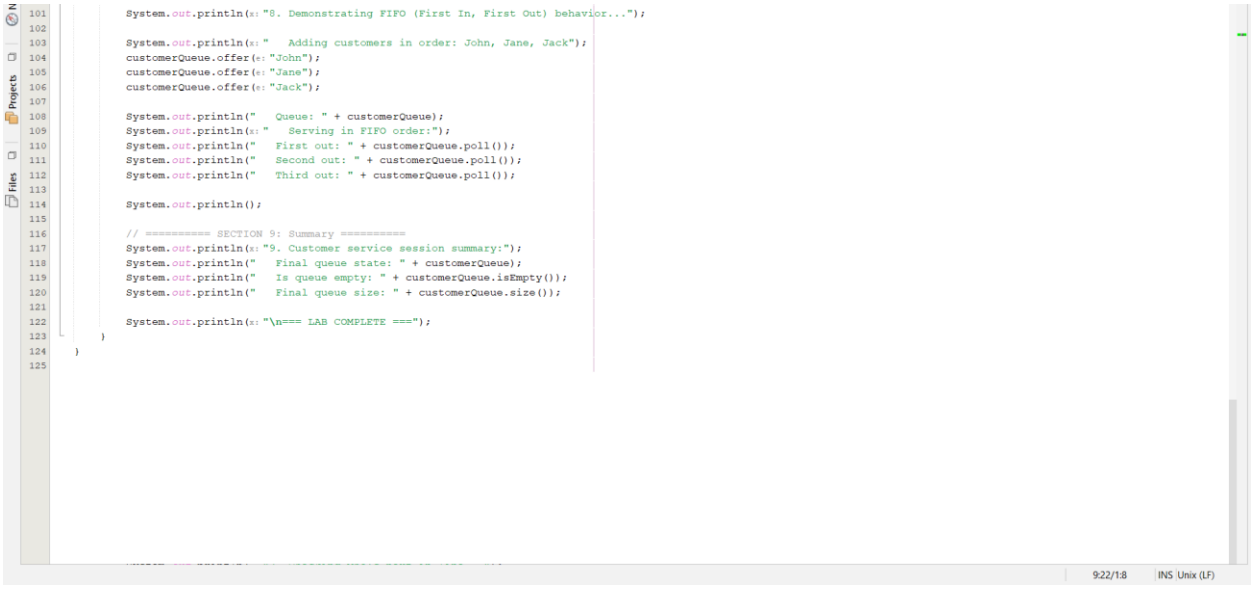
```java
package Queue;

import java.util.LinkedList;
import java.util.Queue;

/**
 * Data Structures Lab 2: Queue - Customer Service Queue System
 */
public class QueueLab {

    public static void main(String[] args) {
        System.out.println("=== CUSTOMER SERVICE QUEUE SYSTEM ===");
        System.out.println("Learning Queue operations through customer service simulation\n");

        // ========== SECTION 1: Creating the Queue ==========
        System.out.println("1. Setting up the customer service queue...");

        Queue<String> customerQueue = new LinkedList<>();

        System.out.println("   Queue created successfully!");
        System.out.println("   Queue size: " + customerQueue.size());
        System.out.println("   Is queue empty: " + customerQueue.isEmpty());
        System.out.println();

        // ========== SECTION 2: Customers Arriving (Adding to Queue) ==========
        System.out.println("2. Customers arriving at the service desk...");

        customerQueue.offer("Alice");
        customerQueue.offer("Bob");
        customerQueue.offer("Charlie");
        customerQueue.offer("Diana");
        customerQueue.offer("Eve");

        System.out.println("   Customers added to queue!");
        System.out.println("   Queue size: " + customerQueue.size());
        System.out.println("   Current queue: " + customerQueue);
        System.out.println();

        // ========== SECTION 3: Checking Who's Next (Peek Operations) ==========
```



```java
        System.out.println();

        // ========== SECTION 4: Serving Customers (Removing from Queue) ==========
        System.out.println("4. Serving customers one by one...");

        System.out.println("   Serving customer: " + customerQueue.poll());
        System.out.println("   Queue after serving the first customer: " + customerQueue);
        System.out.println("   Queue size after serving the first customer: " + customerQueue.size());
        System.out.println();

        System.out.println("   Serving customer: " + customerQueue.poll());
        System.out.println("   Queue after serving another customer: " + customerQueue);
        System.out.println("   Queue size: " + customerQueue.size());
        System.out.println();

        // ========== SECTION 5: More Customers Arriving ==========
        System.out.println("5. More customers arriving while others are being served...");

        customerQueue.offer("Frank");
        customerQueue.offer("Grace");

        System.out.println("   New customers added!");
        System.out.println("   Updated queue: " + customerQueue);
        System.out.println("   Queue size: " + customerQueue.size());
        System.out.println();

        // ========== SECTION 6: Processing All Remaining Customers ==========
        System.out.println("6. Processing all remaining customers...");

        while (!customerQueue.isEmpty()) {
            String customer = customerQueue.poll();
            System.out.println("   Serving: " + customer);
            System.out.println("   Remaining in queue: " + customerQueue.size());
        }

        System.out.println();

        // ========== SECTION 7: Queue Operations with Empty Queue ==========
        System.out.println("7. Testing operations on empty queue...");
```

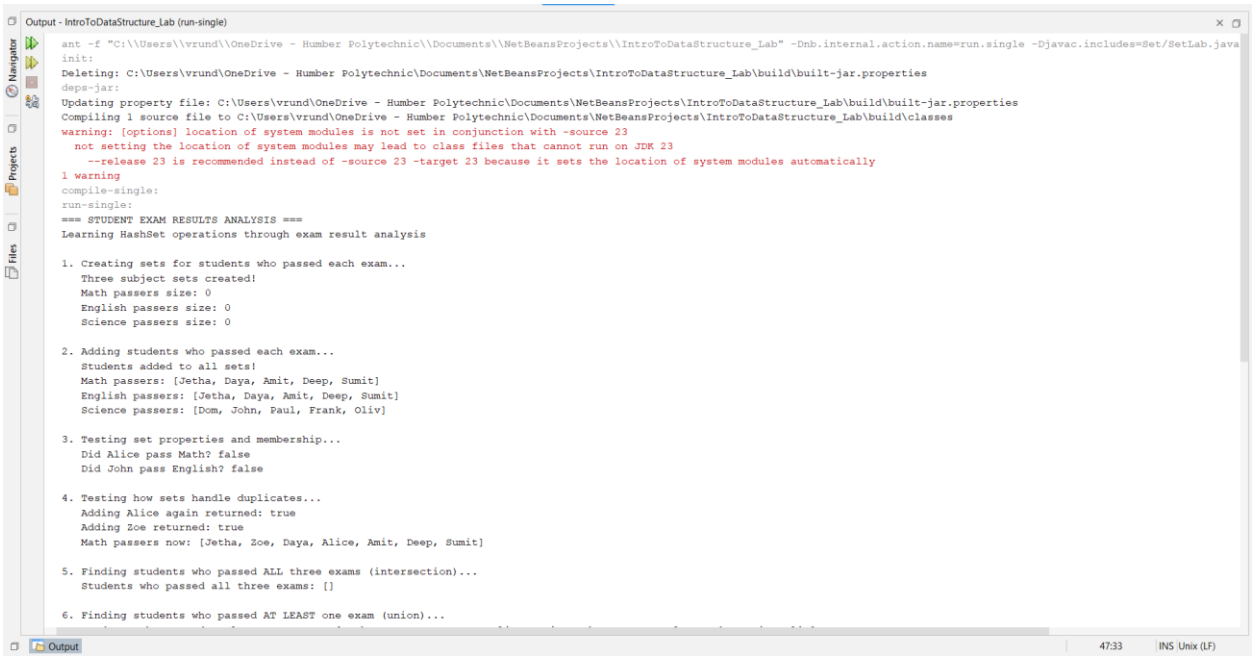


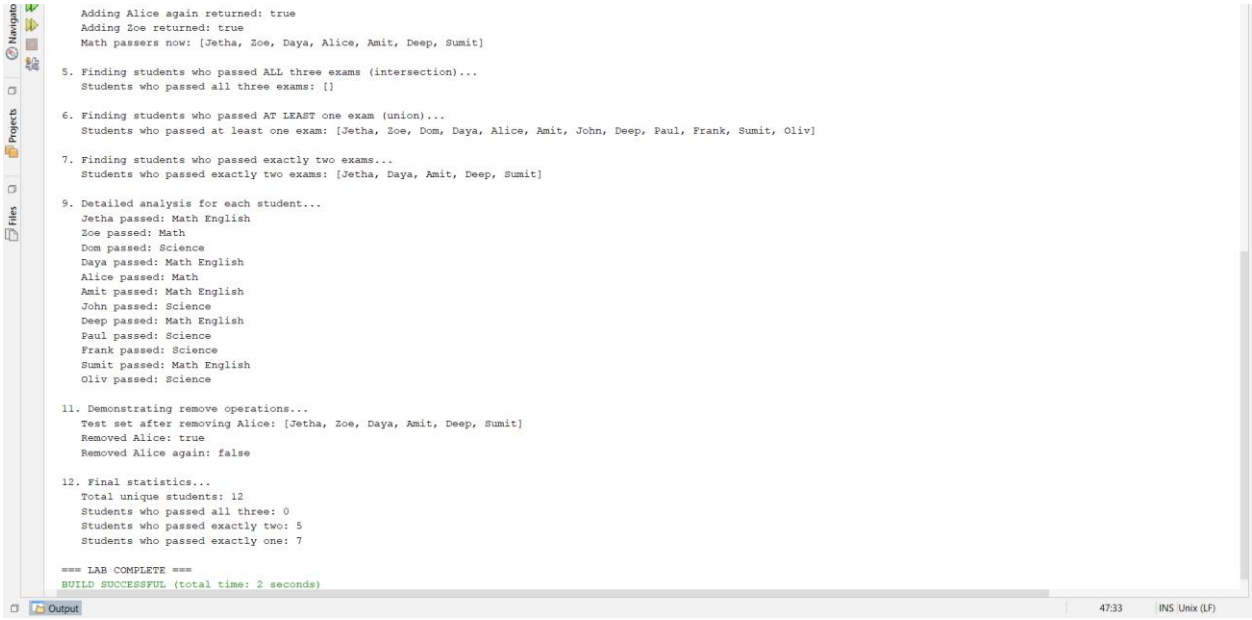
```
        // ========== SECTION 7: Queue Operations with Empty Queue ==========
        System.out.println("7. Testing operations on empty queue...");

        System.out.println("   Queue after serving all customers: " + customerQueue);
        System.out.println("   Is queue empty: " + customerQueue.isEmpty());
        System.out.println("   Queue size: " + customerQueue.size());

        String emptyPeek = customerQueue.peek();
        String emptyPoll = customerQueue.poll();

        System.out.println("   Peek on empty queue: " + emptyPeek);
        System.out.println("   Poll on empty queue: " + emptyPoll);
        System.out.println("   (Notice: peek() and poll() return null for empty queue)");
        System.out.println();

        // ========== SECTION 8: Demonstrating FIFO Behavior ==========
        System.out.println("8. Demonstrating FIFO (First In, First Out) behavior...");

        System.out.println("   Adding customers in order: John, Jane, Jack");
        customerQueue.offer("John");
        customerQueue.offer("Jane");
        customerQueue.offer("Jack");

        System.out.println("   Queue: " + customerQueue);
        System.out.println("   Serving in FIFO order:");
        System.out.println("   First out: " + customerQueue.poll());
        System.out.println("   Second out: " + customerQueue.poll());
        System.out.println("   Third out: " + customerQueue.poll());

        System.out.println();

        // ========== SECTION 9: Summary ==========
        System.out.println("9. Customer service session summary:");
        System.out.println("   Final queue state: " + customerQueue);
        System.out.println("   Is queue empty: " + customerQueue.isEmpty());
        System.out.println("   Final queue size: " + customerQueue.size());

        System.out.println("\n=== LAB COMPLETE ===");
```

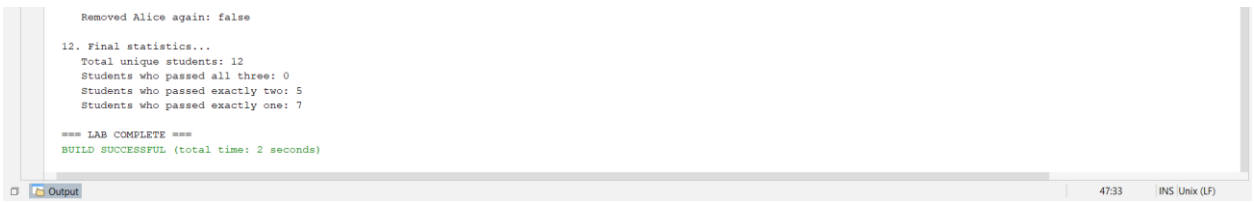


```
        System.out.println("8. Demonstrating FIFO (First In, First Out) behavior...");

        System.out.println("   Adding customers in order: John, Jane, Jack");
        customerQueue.offer("John");
        customerQueue.offer("Jane");
        customerQueue.offer("Jack");

        System.out.println("   Queue: " + customerQueue);
        System.out.println("   Serving in FIFO order:");
        System.out.println("   First out: " + customerQueue.poll());
        System.out.println("   Second out: " + customerQueue.poll());
        System.out.println("   Third out: " + customerQueue.poll());

        System.out.println();

        // ========== SECTION 9: Summary ==========
        System.out.println("9. Customer service session summary:");
        System.out.println("   Final queue state: " + customerQueue);
        System.out.println("   Is queue empty: " + customerQueue.isEmpty());
        System.out.println("   Final queue size: " + customerQueue.size());

        System.out.println("\n=== LAB COMPLETE ===");
    }
}
```

5) SetLab:


Output - IntroToDataStructure_Lab (run-single)
ant -f "C:\\Users\\vrund\\OneDrive - Humber Polytechnic\\Documents\\NetBeansProjects\\IntroToDataStructure_Lab" -Dnb.internal.action.name=run.single -Djavac.includes=Set/SetLab.java
init:
Deleting: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\built-jar.properties
Compiling 1 source file to C:\Users\vrund\OneDrive - Humber Polytechnic\Documents\NetBeansProjects\IntroToDataStructure_Lab\build\classes
warning: [options] location of system modules is not set in conjunction with -source 23
not setting the location of system modules may lead to class files that cannot run on JDK 23
--release 23 is recommended instead of -source 23 -target 23 because it sets the location of system modules automatically
1 warning
compile-single:
run-single:
=== STUDENT EXAM RESULTS ANALYSIS ===
Learning HashSet operations through exam result analysis
1. Creating sets for students who passed each exam...
Three subject sets created!
Math passers size: 0
English passers size: 0
Science passers size: 0
2. Adding students who passed each exam...
Students added to all sets!
Math passers: [Jetha, Daya, Amit, Deep, Sumit]
English passers: [Jetha, Daya, Amit, Deep, Sumit]
Science passers: [Dom, John, Paul, Frank, Oliv]
3. Testing set properties and membership...
Did Alice pass Math? false
Did John pass English? false
4. Testing how sets handle duplicates...
Adding Alice again returned: true
Adding Zoe returned: true
Math passers now: [Jetha, Zoe, Daya, Alice, Amit, Deep, Sumit]
5. Finding students who passed ALL three exams (intersection)...
Students who passed all three exams: []
6. Finding students who passed AT LEAST one exam (union)...
Output
47:33
INS Unix (LF)


Adding Alice again returned: true
Adding Zoe returned: true
Math passers now: [Jetha, Zoe, Daya, Alice, Amit, Deep, Sumit]
5. Finding students who passed ALL three exams (intersection)...
Students who passed all three exams: []
6. Finding students who passed AT LEAST one exam (union)...
Students who passed at least one exam: [Jetha, Zoe, Dom, Daya, Alice, Amit, John, Deep, Paul, Frank, Sumit, Oliv]
7. Finding students who passed exactly two exams...
Students who passed exactly two exams: [Jetha, Daya, Amit, Deep, Sumit]
9. Detailed analysis for each student...
Jetha passed: Math English
Zoe passed: Math
Dom passed: Science
Daya passed: Math English
Alice passed: Math
Amit passed: Math English
John passed: Science
Deep passed: Math English
Paul passed: Science
Frank passed: Science
Sumit passed: Math English
Oliv passed: Science
11. Demonstrating remove operations...
Test set after removing Alice: [Jetha, Zoe, Daya, Amit, Deep, Sumit]
Removed Alice: true
Removed Alice again: false
12. Final statistics...
Total unique students: 12
Students who passed all three: 0
Students who passed exactly two: 5
Students who passed exactly one: 7
=== LAB COMPLETE ===
BUILD SUCCESSFUL (total time: 2 seconds)
Output
47:33
INS Unix (LF)


Removed Alice again: false
12. Final statistics...
Total unique students: 12
Students who passed all three: 0
Students who passed exactly two: 5
Students who passed exactly one: 7
=== LAB COMPLETE ===
BUILD SUCCESSFUL (total time: 2 seconds)
Output
47:33
INS Unix (LF)