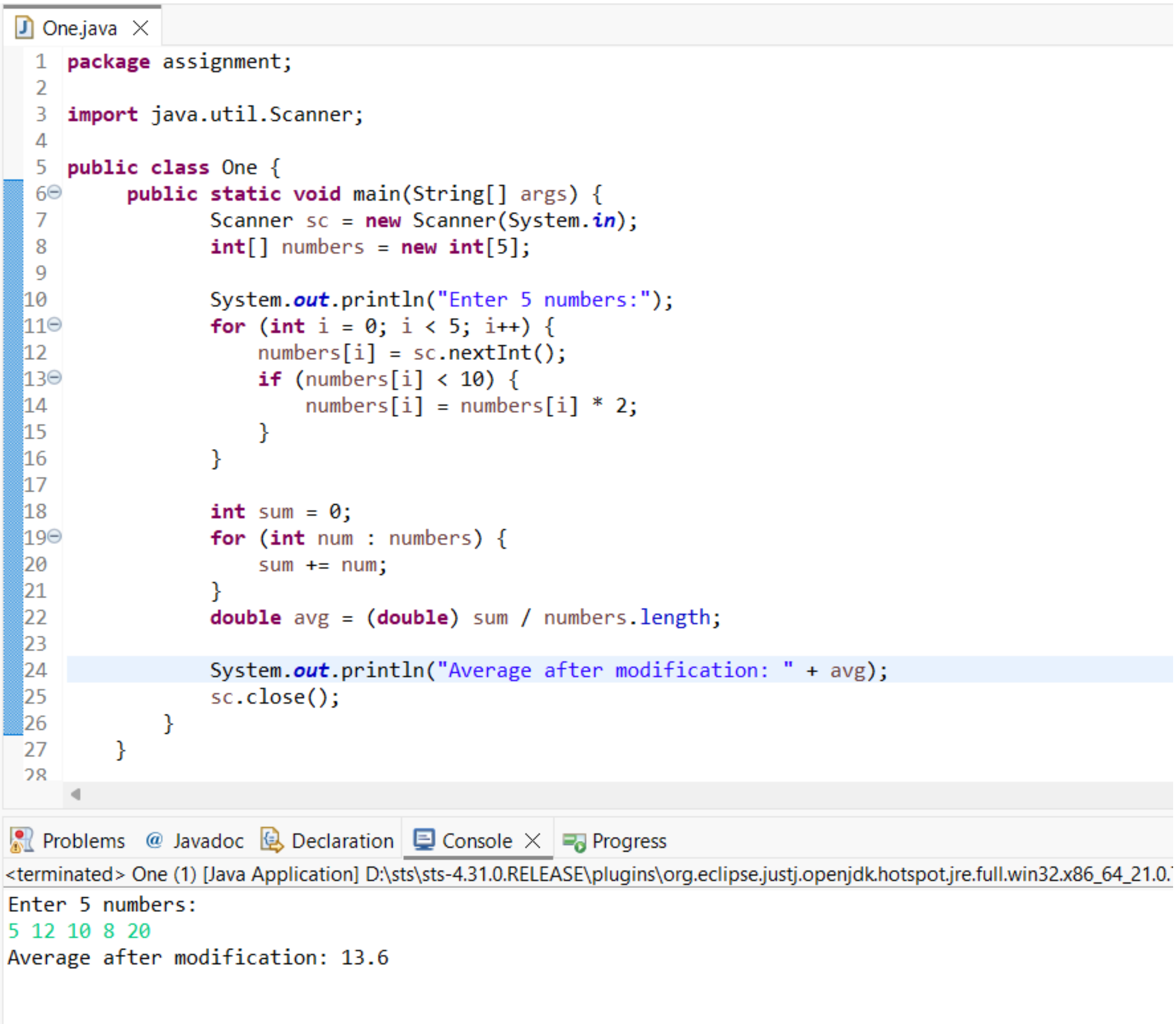
Java

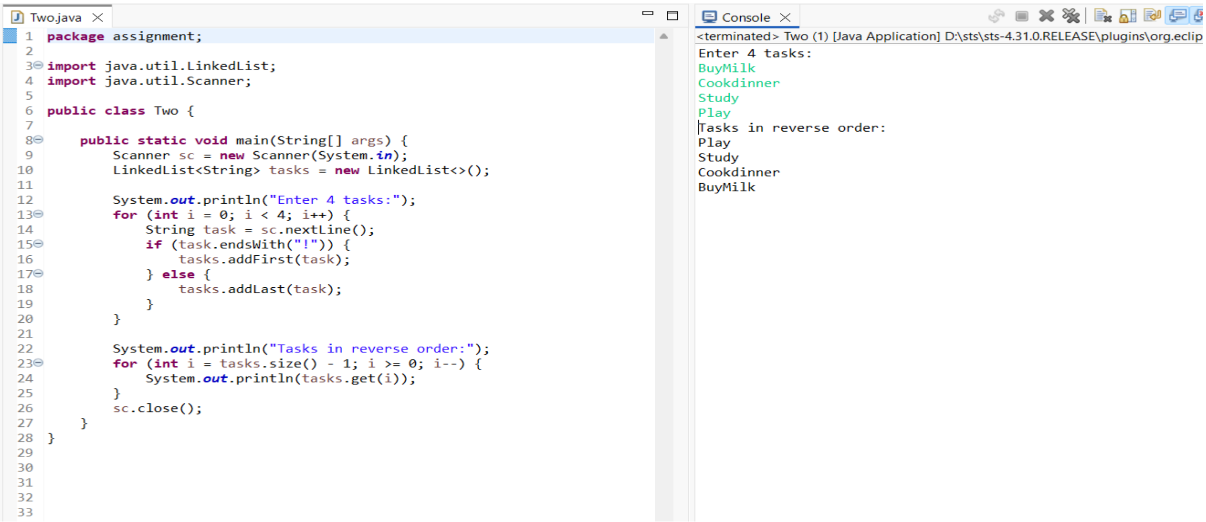
One:Average Confusion (using int[])



```java
package assignment;

import java.util.Scanner;

public class One {
     public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int[] numbers = new int[5];

            System.out.println("Enter 5 numbers:");
            for (int i = 0; i < 5; i++) {
                numbers[i] = sc.nextInt();
                if (numbers[i] < 10) {
                    numbers[i] = numbers[i] * 2;
                }
            }

            int sum = 0;
            for (int num : numbers) {
                sum += num;
            }
            double avg = (double) sum / numbers.length;

            System.out.println("Average after modification: " + avg);
            sc.close();
        }
    }
```

```
<terminated> One (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.
Enter 5 numbers:
5 12 10 8 20
Average after modification: 13.6
```

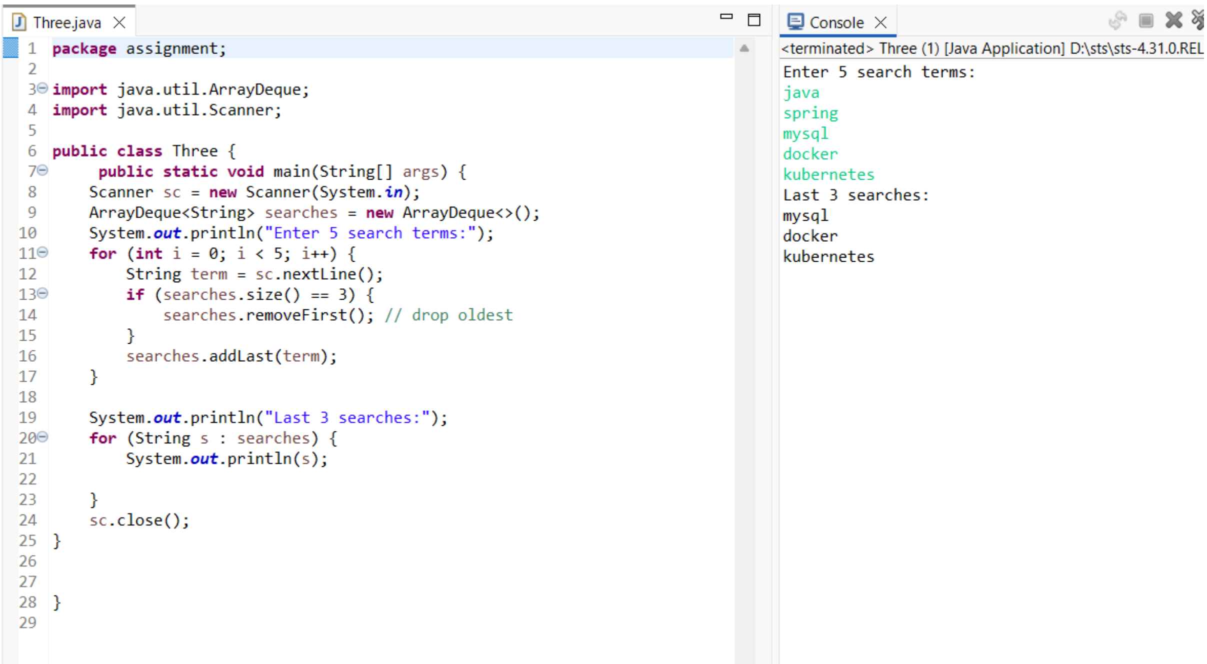
Two :Reversed Task Queue (using LinkedList<String>)



```java
package assignment;

import java.util.LinkedList;
import java.util.Scanner;

public class Two {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<String> tasks = new LinkedList<>();

        System.out.println("Enter 4 tasks:");
        for (int i = 0; i < 4; i++) {
            String task = sc.nextLine();
            if (task.endsWith("!")) {
                tasks.addFirst(task);
            } else {
                tasks.addLast(task);
            }
        }

        System.out.println("Tasks in reverse order:");
        for (int i = tasks.size() - 1; i >= 0; i--) {
            System.out.println(tasks.get(i));
        }
        sc.close();
    }
}
```

```
<terminated> Two (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclip
Enter 4 tasks:
BuyMilk
Cookdinner
Study
Play
Tasks in reverse order:
Play
Study
Cookdinner
BuyMilk
```

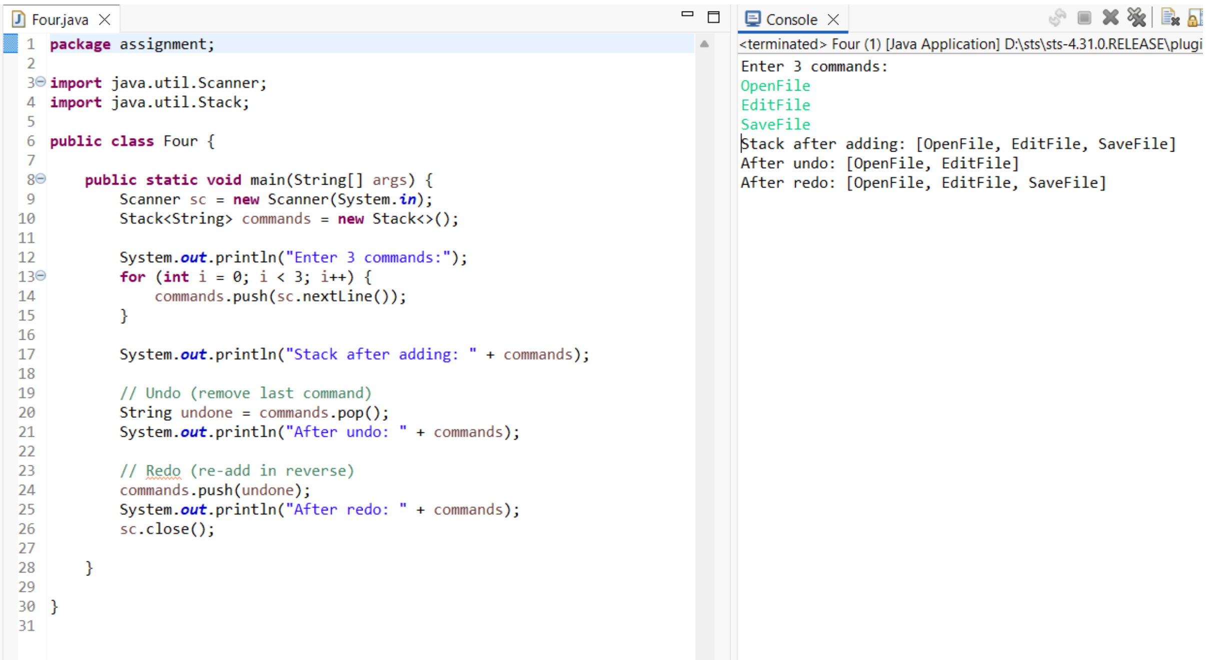
Three:Last 3 Searches(using ArrayDeque<String>)



```java
package assignment;

import java.util.ArrayDeque;
import java.util.Scanner;

public class Three {
     public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    ArrayDeque<String> searches = new ArrayDeque<>();
    System.out.println("Enter 5 search terms:");
    for (int i = 0; i < 5; i++) {
        String term = sc.nextLine();
        if (searches.size() == 3) {
            searches.removeFirst(); // drop oldest
        }
        searches.addLast(term);
    }

    System.out.println("Last 3 searches:");
    for (String s : searches) {
        System.out.println(s);

    }
    sc.close();
}


}
```

```
<terminated> Three (1) [Java Application] D:\sts\sts-4.31.0.REL
Enter 5 search terms:
java
spring
mysql
docker
kubernetes
Last 3 searches:
mysql
docker
kubernetes
```

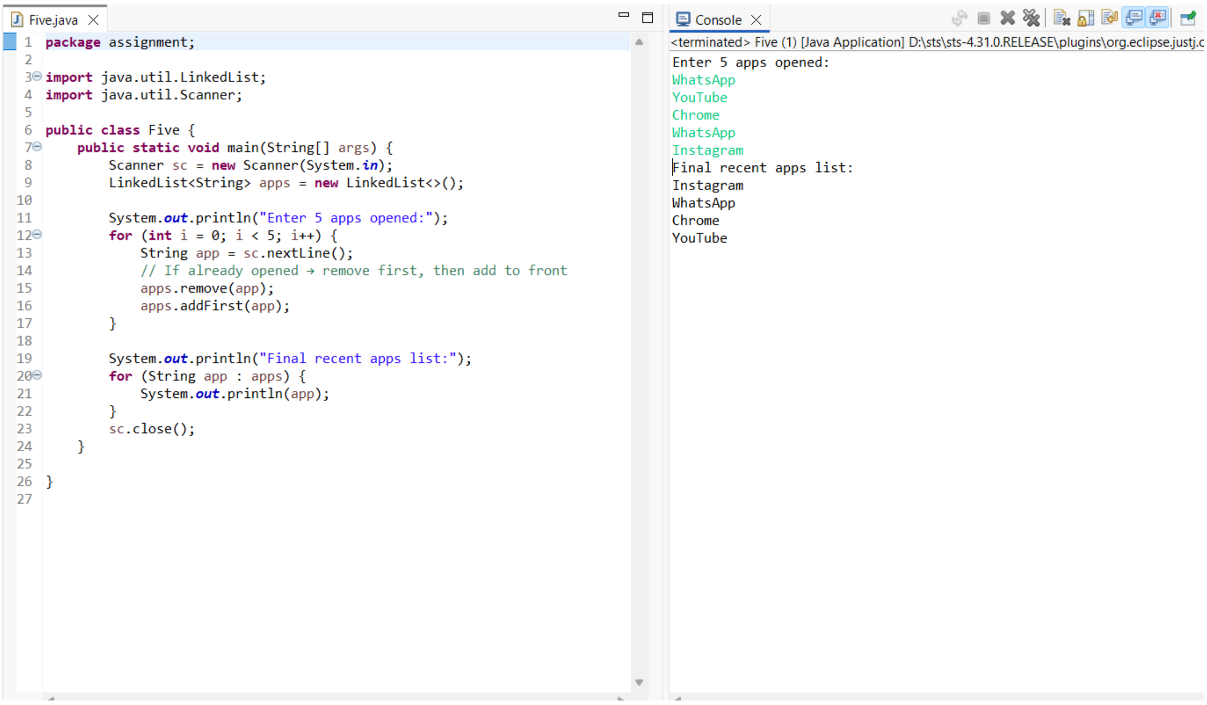
Four:Undo Stack



```java
package assignment;

import java.util.Scanner;
import java.util.Stack;

public class Four {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack<String> commands = new Stack<>();

        System.out.println("Enter 3 commands:");
        for (int i = 0; i < 3; i++) {
            commands.push(sc.nextLine());
        }

        System.out.println("Stack after adding: " + commands);

        // Undo (remove last command)
        String undone = commands.pop();
        System.out.println("After undo: " + commands);

        // Redo (re-add in reverse)
        commands.push(undone);
        System.out.println("After redo: " + commands);
        sc.close();

    }

}
```

```
<terminated> Four (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugi
Enter 3 commands:
OpenFile
EditFile
SaveFile
Stack after adding: [OpenFile, EditFile, SaveFile]
After undo: [OpenFile, EditFile]
After redo: [OpenFile, EditFile, SaveFile]
```

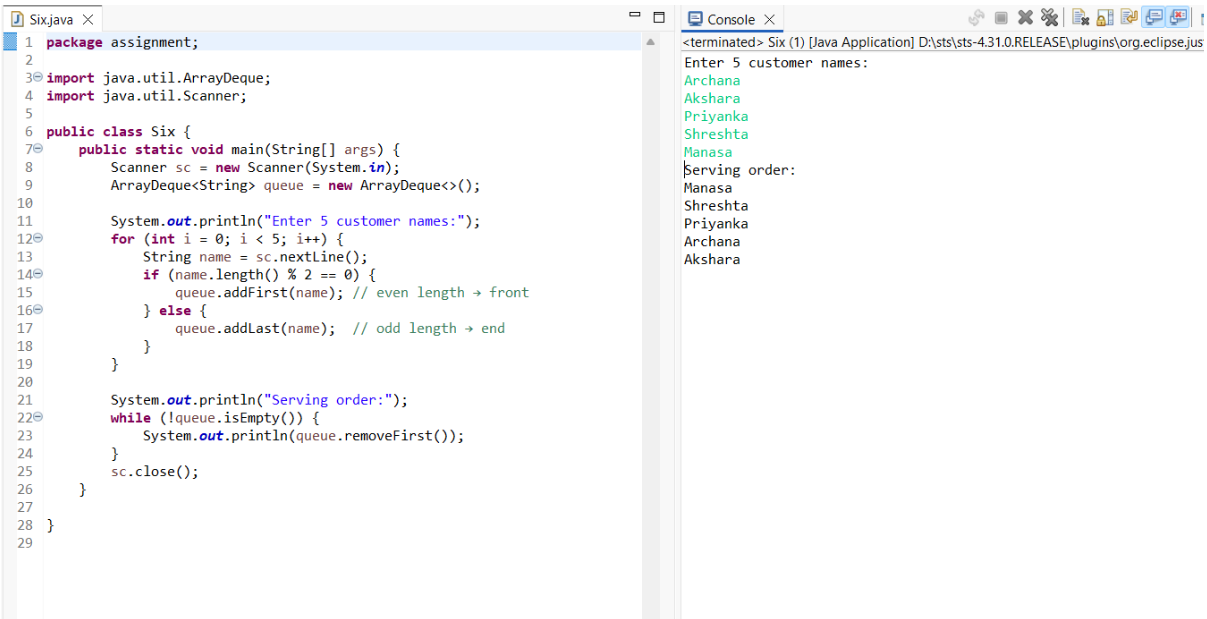
Five:Recent App Memory



```java
package assignment;

import java.util.LinkedList;
import java.util.Scanner;

public class Five {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<String> apps = new LinkedList<>();

        System.out.println("Enter 5 apps opened:");
        for (int i = 0; i < 5; i++) {
            String app = sc.nextLine();
            // If already opened → remove first, then add to front
            apps.remove(app);
            apps.addFirst(app);
        }

        System.out.println("Final recent apps list:");
        for (String app : apps) {
            System.out.println(app);
        }
        sc.close();
    }

}
```

```
<terminated> Five (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.c
Enter 5 apps opened:
WhatsApp
YouTube
Chrome
WhatsApp
Instagram
Final recent apps list:
Instagram
WhatsApp
Chrome
YouTube
```

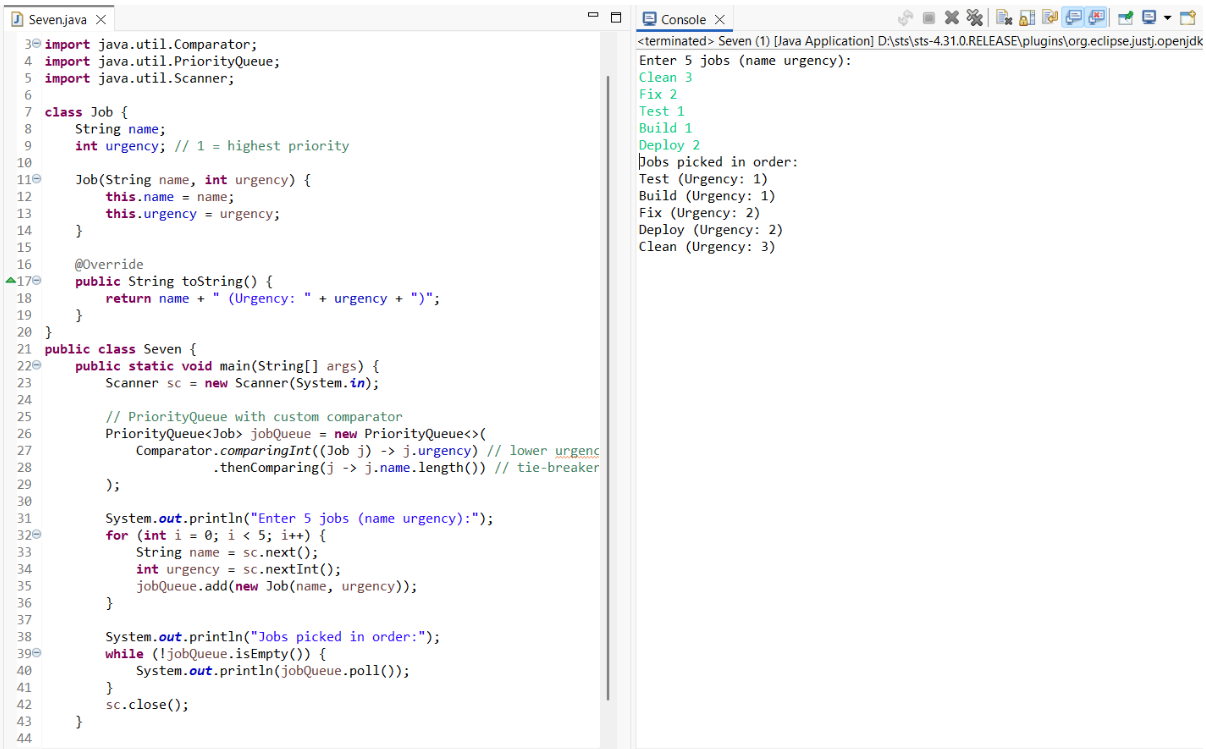
Six: Grocery Line Shuffle



```java
package assignment;

import java.util.ArrayDeque;
import java.util.Scanner;

public class Six {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayDeque<String> queue = new ArrayDeque<>();

        System.out.println("Enter 5 customer names:");
        for (int i = 0; i < 5; i++) {
            String name = sc.nextLine();
            if (name.length() % 2 == 0) {
                queue.addFirst(name); // even length → front
            } else {
                queue.addLast(name);  // odd length → end
            }
        }

        System.out.println("Serving order:");
        while (!queue.isEmpty()) {
            System.out.println(queue.removeFirst());
        }
        sc.close();
    }

}
```

```
<terminated> Six (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.jus
Enter 5 customer names:
Archana
Akshara
Priyanka
Shreshta
Manasa
Serving order:
Manasa
Shreshta
Priyanka
Archana
Akshara
```

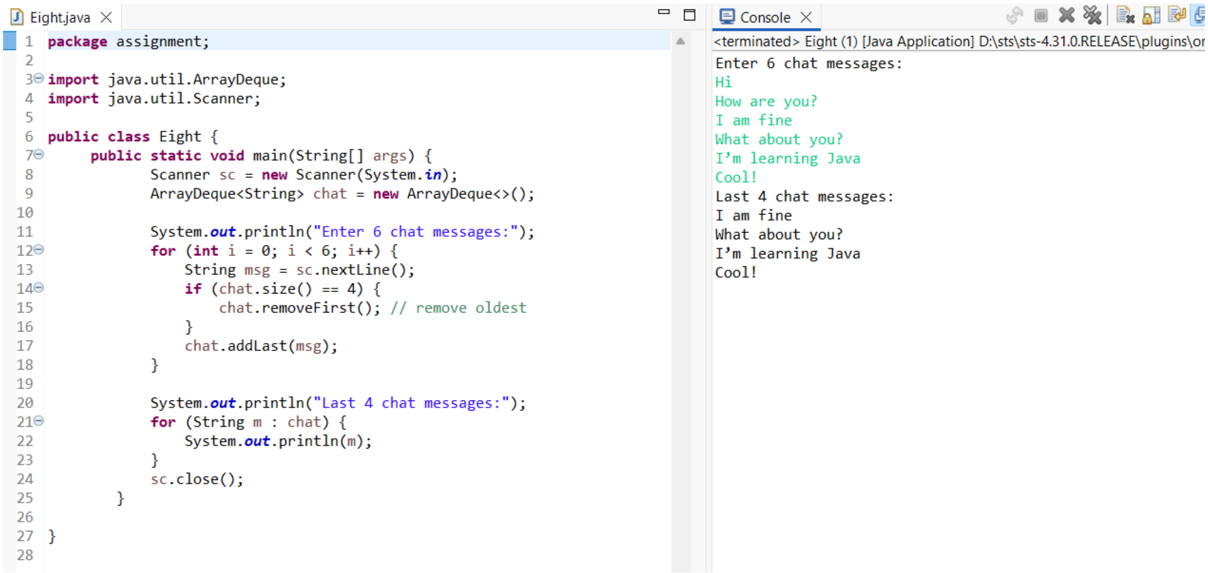
Seven: Smart Job Picker



```java
import java.util.Comparator;
import java.util.PriorityQueue;
import java.util.Scanner;

class Job {
    String name;
    int urgency; // 1 = highest priority

    Job(String name, int urgency) {
        this.name = name;
        this.urgency = urgency;
    }

    @Override
    public String toString() {
        return name + " (Urgency: " + urgency + ")";
    }
}
public class Seven {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // PriorityQueue with custom comparator
        PriorityQueue<Job> jobQueue = new PriorityQueue<>(
            Comparator.comparingInt((Job j) -> j.urgency) // lower urgenc
                      .thenComparing(j -> j.name.length()) // tie-breaker
        );

        System.out.println("Enter 5 jobs (name urgency):");
        for (int i = 0; i < 5; i++) {
            String name = sc.next();
            int urgency = sc.nextInt();
            jobQueue.add(new Job(name, urgency));
        }

        System.out.println("Jobs picked in order:");
        while (!jobQueue.isEmpty()) {
            System.out.println(jobQueue.poll());
        }
        sc.close();
    }

```

```
<terminated> Seven (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk
Enter 5 jobs (name urgency):
Clean 3
Fix 2
Test 1
Build 1
Deploy 2
Jobs picked in order:
Test (Urgency: 1)
Build (Urgency: 1)
Fix (Urgency: 2)
Deploy (Urgency: 2)
Clean (Urgency: 3)
```

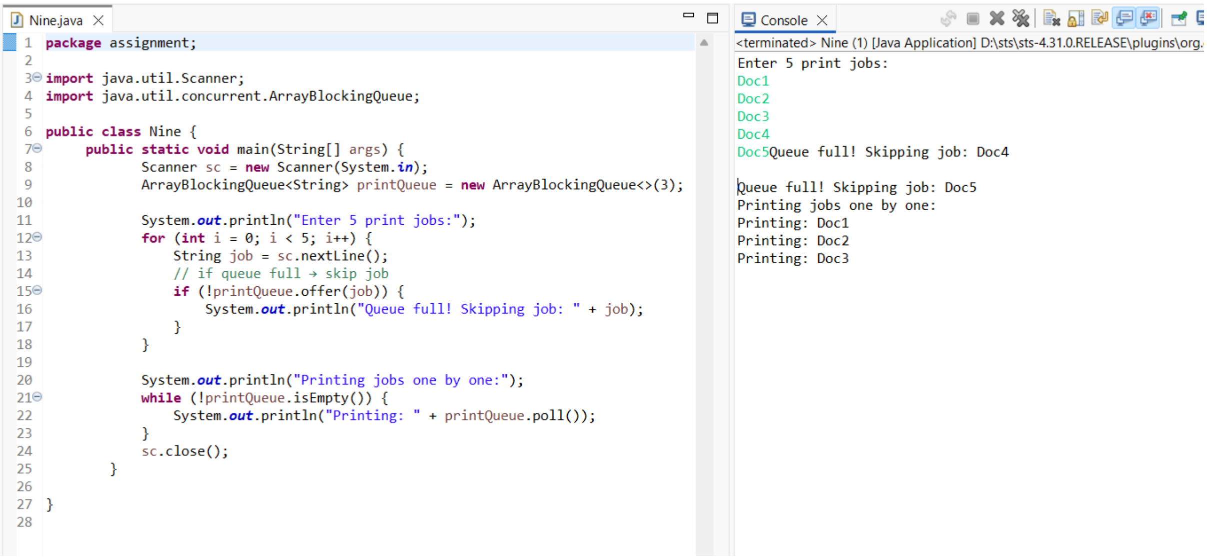
Eight: Limited Chat History



```java
package assignment;

import java.util.ArrayDeque;
import java.util.Scanner;

public class Eight {
     public static void main(String[] args) {
             Scanner sc = new Scanner(System.in);
             ArrayDeque<String> chat = new ArrayDeque<>();

             System.out.println("Enter 6 chat messages:");
             for (int i = 0; i < 6; i++) {
                 String msg = sc.nextLine();
                 if (chat.size() == 4) {
                     chat.removeFirst(); // remove oldest
                 }
                 chat.addLast(msg);
             }

             System.out.println("Last 4 chat messages:");
             for (String m : chat) {
                 System.out.println(m);
             }
             sc.close();
         }

}

```

```
<terminated> Eight (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\or
Enter 6 chat messages:
Hi
How are you?
I am fine
What about you?
I’m learning Java
Cool!
Last 4 chat messages:
I am fine
What about you?
I’m learning Java
Cool!
```

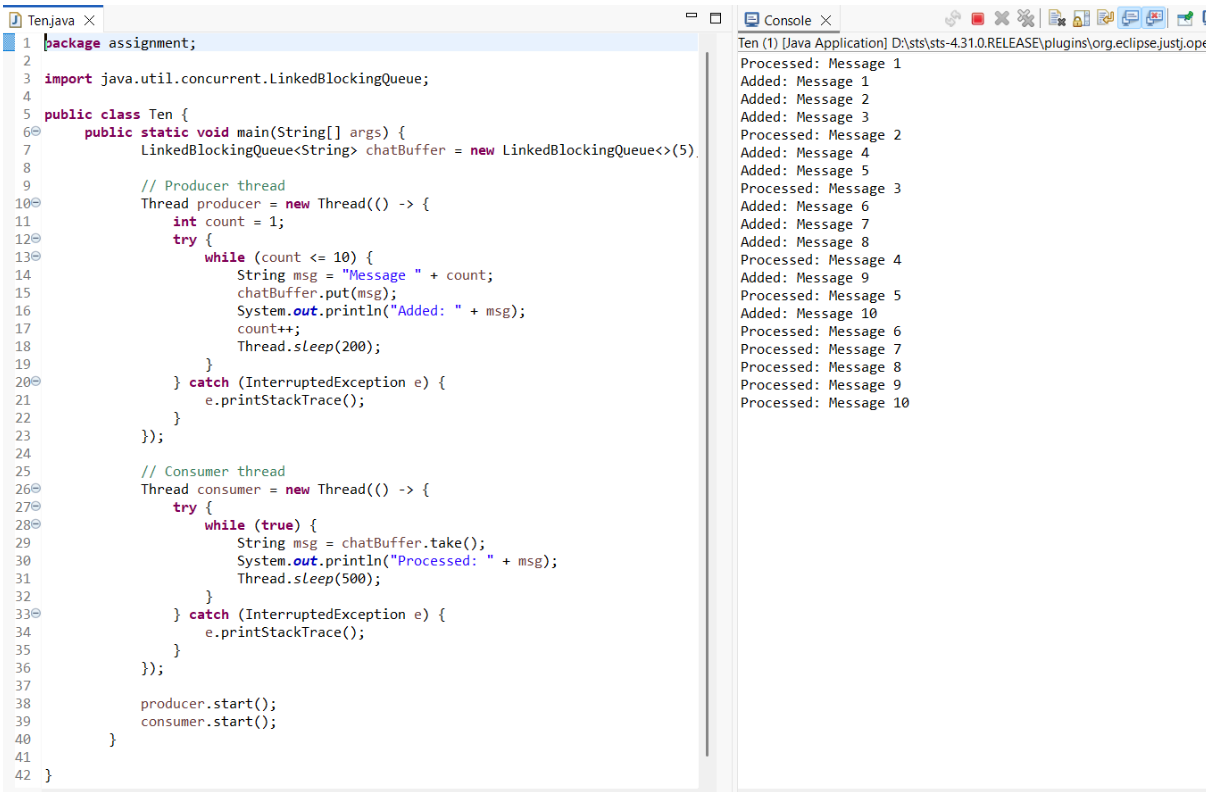
Nine: Print Manager



```java
package assignment;

import java.util.Scanner;
import java.util.concurrent.ArrayBlockingQueue;

public class Nine {
     public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            ArrayBlockingQueue<String> printQueue = new ArrayBlockingQueue<>(3);

            System.out.println("Enter 5 print jobs:");
            for (int i = 0; i < 5; i++) {
                String job = sc.nextLine();
                // if queue full → skip job
                if (!printQueue.offer(job)) {
                    System.out.println("Queue full! Skipping job: " + job);
                }
            }

            System.out.println("Printing jobs one by one:");
            while (!printQueue.isEmpty()) {
                System.out.println("Printing: " + printQueue.poll());
            }
            sc.close();
        }

}

```

```
<terminated> Nine (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.
Enter 5 print jobs:
Doc1
Doc2
Doc3
Doc4
Doc5Queue full! Skipping job: Doc4

Queue full! Skipping job: Doc5
Printing jobs one by one:
Printing: Doc1
Printing: Doc2
Printing: Doc3
```

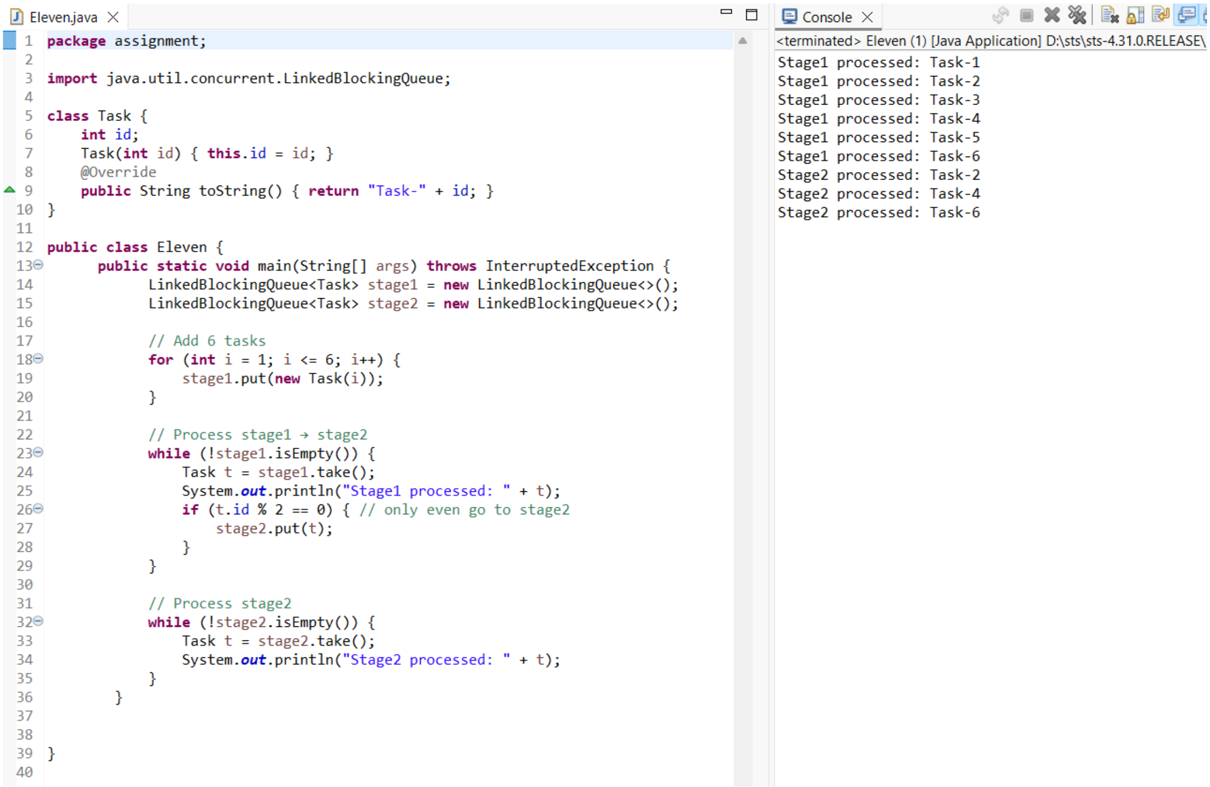
Ten: Chat Processor



```java
package assignment;

import java.util.concurrent.LinkedBlockingQueue;

public class Ten {
     public static void main(String[] args) {
            LinkedBlockingQueue<String> chatBuffer = new LinkedBlockingQueue<>(5)

            // Producer thread
            Thread producer = new Thread(() -> {
                int count = 1;
                try {
                    while (count <= 10) {
                        String msg = "Message " + count;
                        chatBuffer.put(msg);
                        System.out.println("Added: " + msg);
                        count++;
                        Thread.sleep(200);
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            });

            // Consumer thread
            Thread consumer = new Thread(() -> {
                try {
                    while (true) {
                        String msg = chatBuffer.take();
                        System.out.println("Processed: " + msg);
                        Thread.sleep(500);
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            });

            producer.start();
            consumer.start();
        }

}
```

```
Ten (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.ope
Processed: Message 1
Added: Message 1
Added: Message 2
Added: Message 3
Processed: Message 2
Added: Message 4
Added: Message 5
Processed: Message 3
Added: Message 6
Added: Message 7
Added: Message 8
Processed: Message 4
Added: Message 9
Processed: Message 5
Added: Message 10
Processed: Message 6
Processed: Message 7
Processed: Message 8
Processed: Message 9
Processed: Message 10
```

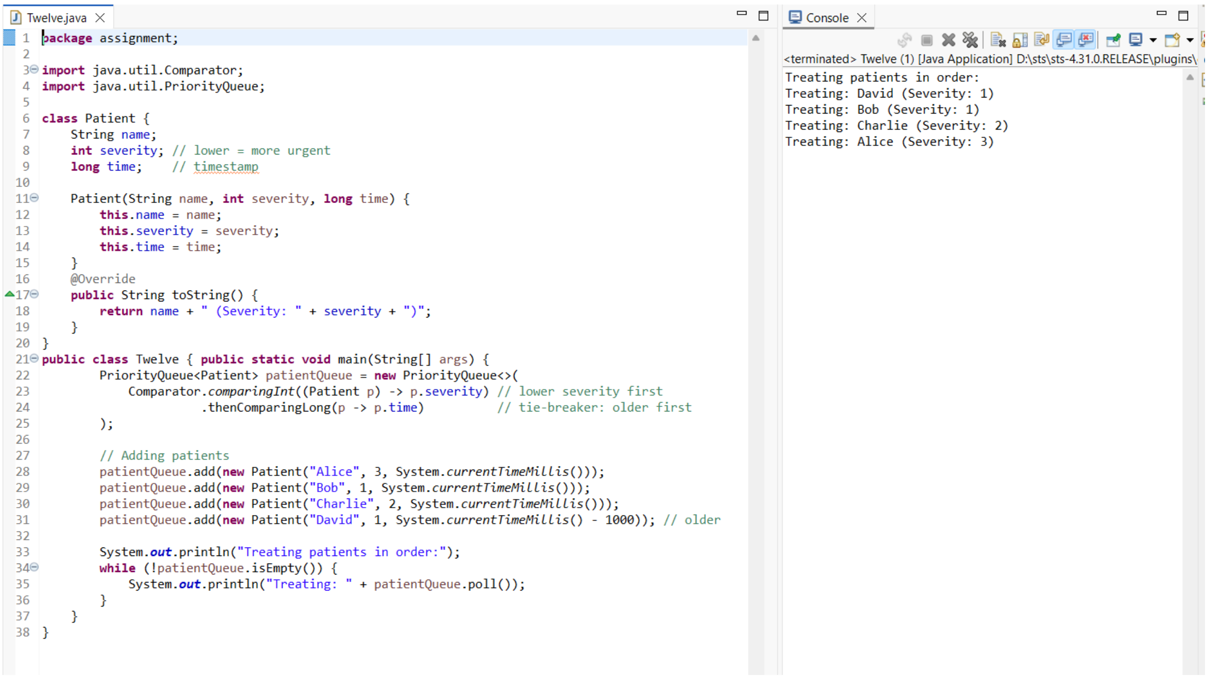
Eleven: Stage-Based Task Runner



```java
package assignment;

import java.util.concurrent.LinkedBlockingQueue;

class Task {
    int id;
    Task(int id) { this.id = id; }
    @Override
    public String toString() { return "Task-" + id; }
}

public class Eleven {
      public static void main(String[] args) throws InterruptedException {
            LinkedBlockingQueue<Task> stage1 = new LinkedBlockingQueue<>();
            LinkedBlockingQueue<Task> stage2 = new LinkedBlockingQueue<>();

            // Add 6 tasks
            for (int i = 1; i <= 6; i++) {
                stage1.put(new Task(i));
            }

            // Process stage1 → stage2
            while (!stage1.isEmpty()) {
                Task t = stage1.take();
                System.out.println("Stage1 processed: " + t);
                if (t.id % 2 == 0) { // only even go to stage2
                    stage2.put(t);
                }
            }

            // Process stage2
            while (!stage2.isEmpty()) {
                Task t = stage2.take();
                System.out.println("Stage2 processed: " + t);
            }
        }


}

```

```
<terminated> Eleven (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\
Stage1 processed: Task-1
Stage1 processed: Task-2
Stage1 processed: Task-3
Stage1 processed: Task-4
Stage1 processed: Task-5
Stage1 processed: Task-6
Stage2 processed: Task-2
Stage2 processed: Task-4
Stage2 processed: Task-6
```

Twelve: Emergency Patient Tracker



```java
package assignment;

import java.util.Comparator;
import java.util.PriorityQueue;

class Patient {
    String name;
    int severity; // lower = more urgent
    long time;    // timestamp

    Patient(String name, int severity, long time) {
        this.name = name;
        this.severity = severity;
        this.time = time;
    }
    @Override
    public String toString() {
        return name + " (Severity: " + severity + ")";
    }
}
public class Twelve { public static void main(String[] args) {
        PriorityQueue<Patient> patientQueue = new PriorityQueue<>(
            Comparator.comparingInt((Patient p) -> p.severity) // lower severity first
                      .thenComparingLong(p -> p.time)          // tie-breaker: older first
        );

        // Adding patients
        patientQueue.add(new Patient("Alice", 3, System.currentTimeMillis()));
        patientQueue.add(new Patient("Bob", 1, System.currentTimeMillis()));
        patientQueue.add(new Patient("Charlie", 2, System.currentTimeMillis()));
        patientQueue.add(new Patient("David", 1, System.currentTimeMillis() - 1000)); // older

        System.out.println("Treating patients in order:");
        while (!patientQueue.isEmpty()) {
            System.out.println("Treating: " + patientQueue.poll());
        }
    }
}
```

```
<terminated> Twelve (1) [Java Application] D:\sts\sts-4.31.0.RELEASE\plugins\
Treating patients in order:
Treating: David (Severity: 1)
Treating: Bob (Severity: 1)
Treating: Charlie (Severity: 2)
Treating: Alice (Severity: 3)
```