

The screenshot shows an IDE with four tabs: `AverageConfusion.java`, `ReversedTaskQueue.java`, `RecentAppMemory.java`, and `GroceryLineShuffle.java`. The `AverageConfusion.java` tab is active, displaying the following code:

```
1 public class AverageConfusion {
2     public static void main(String[] args) {
3         int[] numbers = {3, 12, 7, 15, 9};
4         int sum = 0;
5
6         for (int i = 0; i < numbers.length; i++) {
7             if (numbers[i] < 10) {
8                 numbers[i] = numbers[i] * 2;
9             }
10            sum += numbers[i];
11        }
12
13        double average = (double) sum / numbers.length;
14        System.out.println("Average: " + average);
15    }
16 }
17
```

Below the code editor, the `TERMINAL` tab is selected, showing the execution output:

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' -veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\AverageConfusion'
Average: 13.0
PS C:\Assignments>
```

Figure 1AverageConfusion

```
ChatProcessor.java > ...
1 import java.util.concurrent.LinkedBlockingQueue;
2
3 public class ChatProcessor {
4     public static void main(String[] args) {
5         LinkedBlockingQueue<String> buffer = new LinkedBlockingQueue<>(capacity:5);
6
7         Thread writer = new Thread(() -> {
8             int count = 1;
9             try {
10                 while (count <= 10) {
11                     buffer.put("Message " + count);
12                     System.out.println("Added: Message " + count);
13                 }
14             } catch (InterruptedException e) {
15                 e.printStackTrace();
16             }
17         });
18         writer.start();
19
20         while (true) {
21             try {
22                 String message = buffer.take();
23                 System.out.println("Read: " + message);
24             } catch (InterruptedException e) {
25                 e.printStackTrace();
26             }
27         }
28     }
29 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Run: ChatProcess

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\
veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\Assignments_5c079
ssor'
Added: Message 1
Read: Message 1
Added: Message 2
Added: Message 3
Read: Message 2
Added: Message 4
Added: Message 5
Read: Message 3
Added: Message 6
Added: Message 7
Added: Message 8
Read: Message 4
Added: Message 9
Added: Message 10
Read: Message 5
Read: Message 6
Read: Message 7
Read: Message 8
Read: Message 9
Read: Message 10
```

Figure 2 ChatProcessor

```
EmergencyPatientTracker.java
13 }
14
15 public class EmergencyPatientTracker {
16     Run | Debug
17     public static void main(String[] args) {
18         Comparator<Patient> comparator = (a, b) -> {
19             if (a.severity != b.severity) return Integer.compare(a.severity, b.severity);
20             return Long.compare(a.timestamp, b.timestamp);
21         };
22
23         PriorityQueue<Patient> queue = new PriorityQueue<>(initialCapacity:5, comparator);
24
25         addPatient(queue, new Patient(name:"Arun", severity:3));
26         addPatient(queue, new Patient(name:"Bhanu", severity:2));
27         addPatient(queue, new Patient(name:"Charitha", severity:2));
28         addPatient(queue, new Patient(name:"Dinesh", severity:1));
29         addPatient(queue, new Patient(name:"Eshwar", severity:4));
30         addPatient(queue, new Patient(name:"Fathima", severity:1));
31
32         while (!queue.isEmpty()) {
33             Patient p = queue.poll();
34             System.out.println("Treating: " + p.name + " (Severity: " + p.severity + ")");
35         }
36     }
37 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Run: EmergencyPatientTracker

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Use
veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\Assignments_5c07956e\
PatientTracker'
Queue full, cannot add: Fathima
Treating: Dinesh (Severity: 1)
Treating: Bhanu (Severity: 2)
Treating: Charitha (Severity: 2)
Treating: Arun (Severity: 3)
Treating: Eshwar (Severity: 4)
PS C:\Assignments>
```

Figure 3EmergencyPatientTracker

```
J GroceryLineShuffle.java > GroceryLineShuffle > main(String[])
2 import java.util.Deque;
3 import java.util.Scanner;
4
5 public class GroceryLineShuffle {
6     Run | Debug
7     public static void main(String[] args) {
8         Deque<String> queue = new ArrayDeque<>();
9         Scanner scanner = new Scanner(System.in);
10
11         for (int i = 0; i < 5; i++) {
12             String name = scanner.nextLine();
13             if (name.length() % 2 == 0) {
14                 queue.addFirst(name);
15             } else {
16                 queue.addLast(name);
17             }
18         }
19
20         for (String customer : queue) {
21             System.out.println(customer);
22         }
23     }
24 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
neShuffle'
Nikhila
Rajesh
Anu
Akki
Manoj
Akki
Rajesh
Nikhila
Anu
Manoj
PS C:\Assignments>
```

Figure 4 GroceryLineShuffle

```

J LastThreeSearches.java > ...
1  import java.util.ArrayDeque;
2  import java.util.Deque;
3  import java.util.Scanner;
4
5  public class LastThreeSearches {
6      Run | Debug
7      public static void main(String[] args) {
8          Deque<String> searchHistory = new ArrayDeque<>();
9          Scanner scanner = new Scanner(System.in);
10
11          while (true) {
12              String input = scanner.nextLine();
13              if (input.equalsIgnoreCase("exit")) {
14                  break;
15              }
16              if (searchHistory.size() == 3) {
17                  searchHistory.removeFirst();
18              }
19              searchHistory.addLast(input);
20          }
21          for (String search : searchHistory) {

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

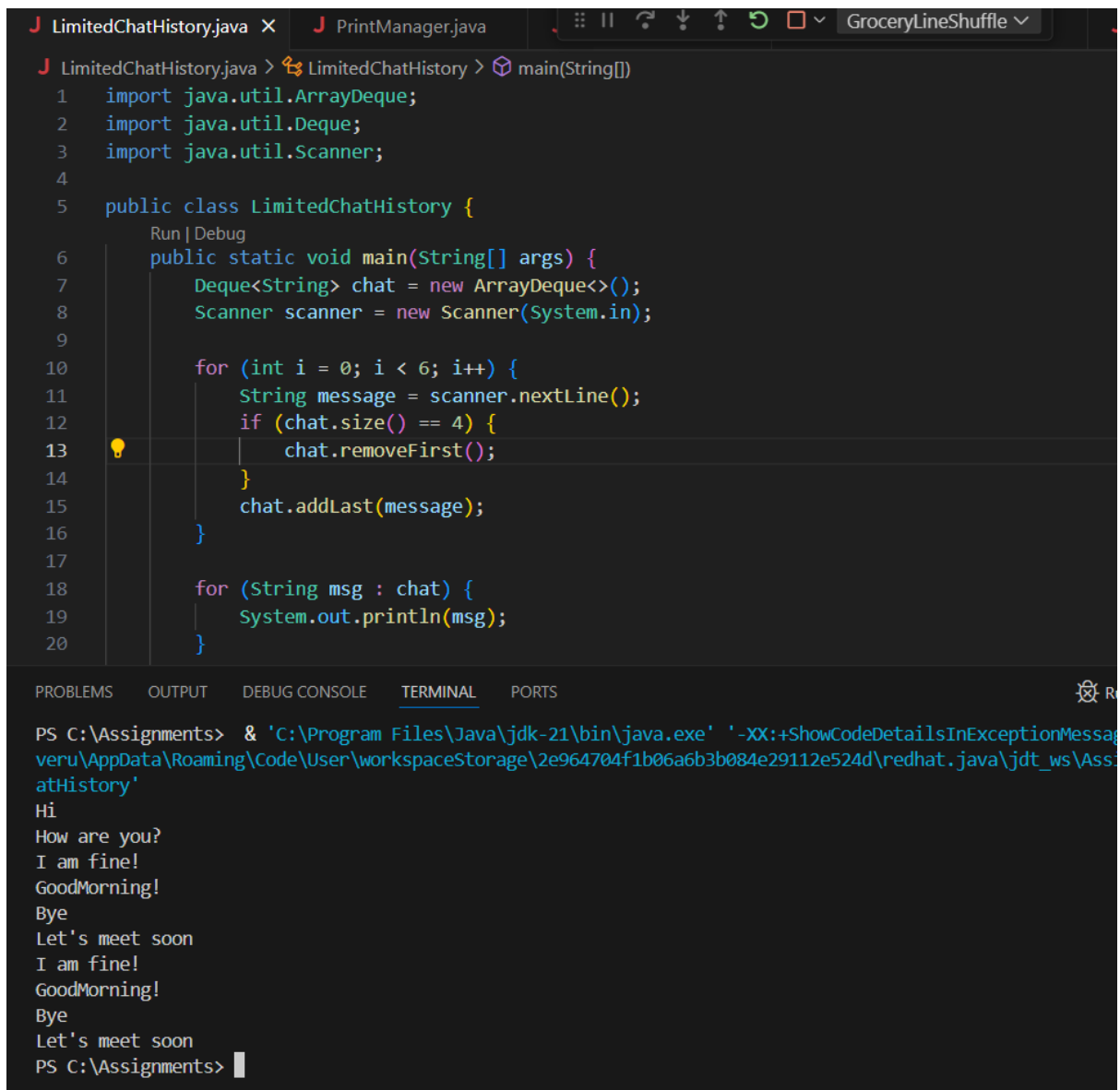
PORTS

```

PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
PS C:\Assignments> java
PS C:\Assignments> python
PS C:\Assignments> springboot
PS C:\Assignments> git
PS C:\Assignments> exit
PS C:\Assignments> python
PS C:\Assignments> springboot
PS C:\Assignments> git
PS C:\Assignments>

```

Figure 5LastThreeSearches



The screenshot displays an IDE with the file `LimitedChatHistory.java` open. The code defines a `main` method that uses a `Deque` to maintain a chat history of size 4. It reads input from `System.in` and prints the history. The terminal output shows the program's execution with the following sequence of messages: `Hi`, `How are you?`, `I am fine!`, `GoodMorning!`, `Bye`, `Let's meet soon`, `I am fine!`, `GoodMorning!`, `Bye`, and `Let's meet soon`. The terminal prompt is `PS C:\Assignments>`.

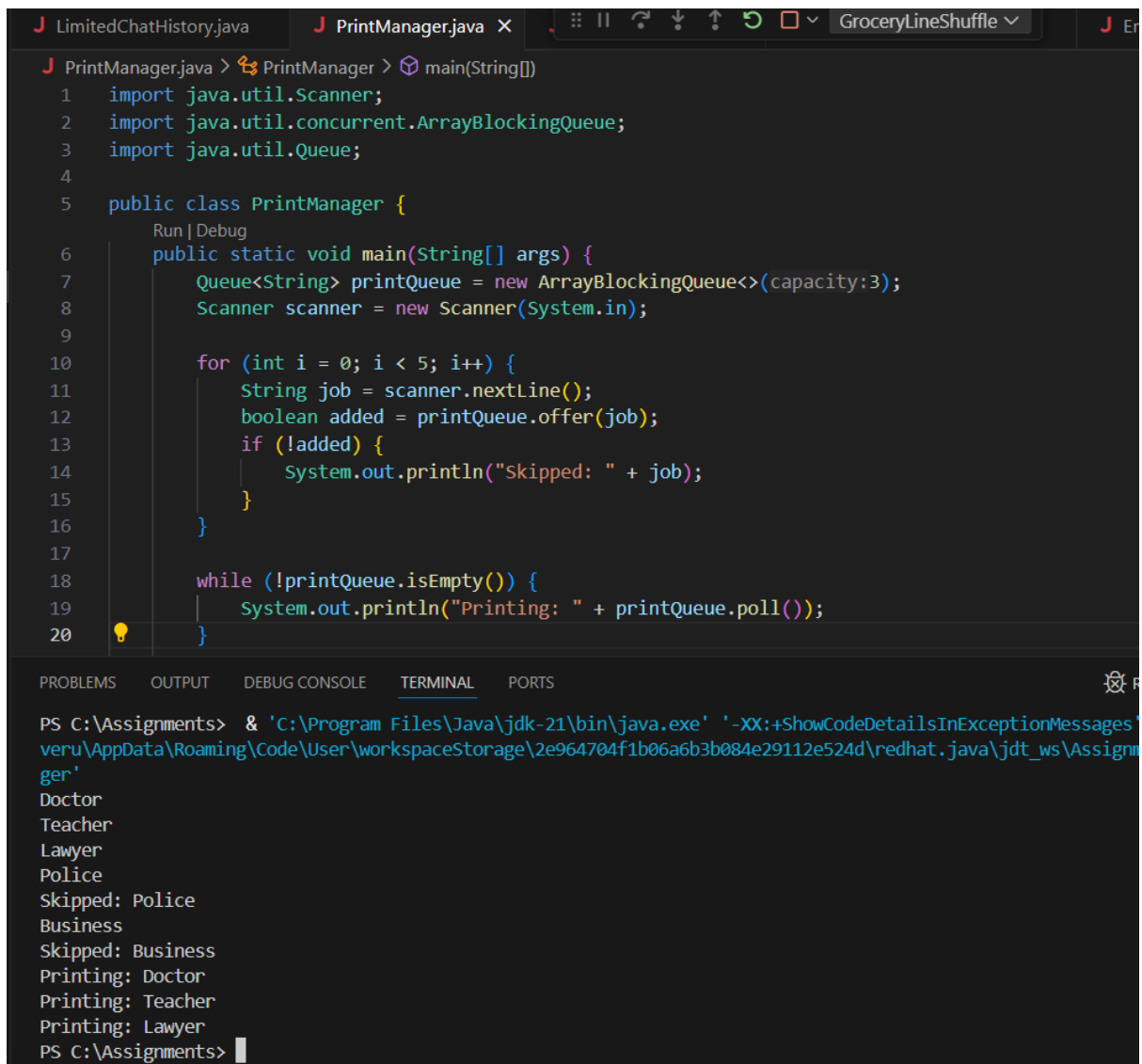
```
1 import java.util.ArrayDeque;
2 import java.util.Deque;
3 import java.util.Scanner;
4
5 public class LimitedChatHistory {
6     public static void main(String[] args) {
7         Deque<String> chat = new ArrayDeque<>();
8         Scanner scanner = new Scanner(System.in);
9
10        for (int i = 0; i < 6; i++) {
11            String message = scanner.nextLine();
12            if (chat.size() == 4) {
13                chat.removeFirst();
14            }
15            chat.addLast(message);
16        }
17
18        for (String msg : chat) {
19            System.out.println(msg);
20        }
21    }
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessage' -veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\Assignments\LimitedChatHistory'

Hi
How are you?
I am fine!
GoodMorning!
Bye
Let's meet soon
I am fine!
GoodMorning!
Bye
Let's meet soon
PS C:\Assignments>

Figure 6 LimitedChatHistory



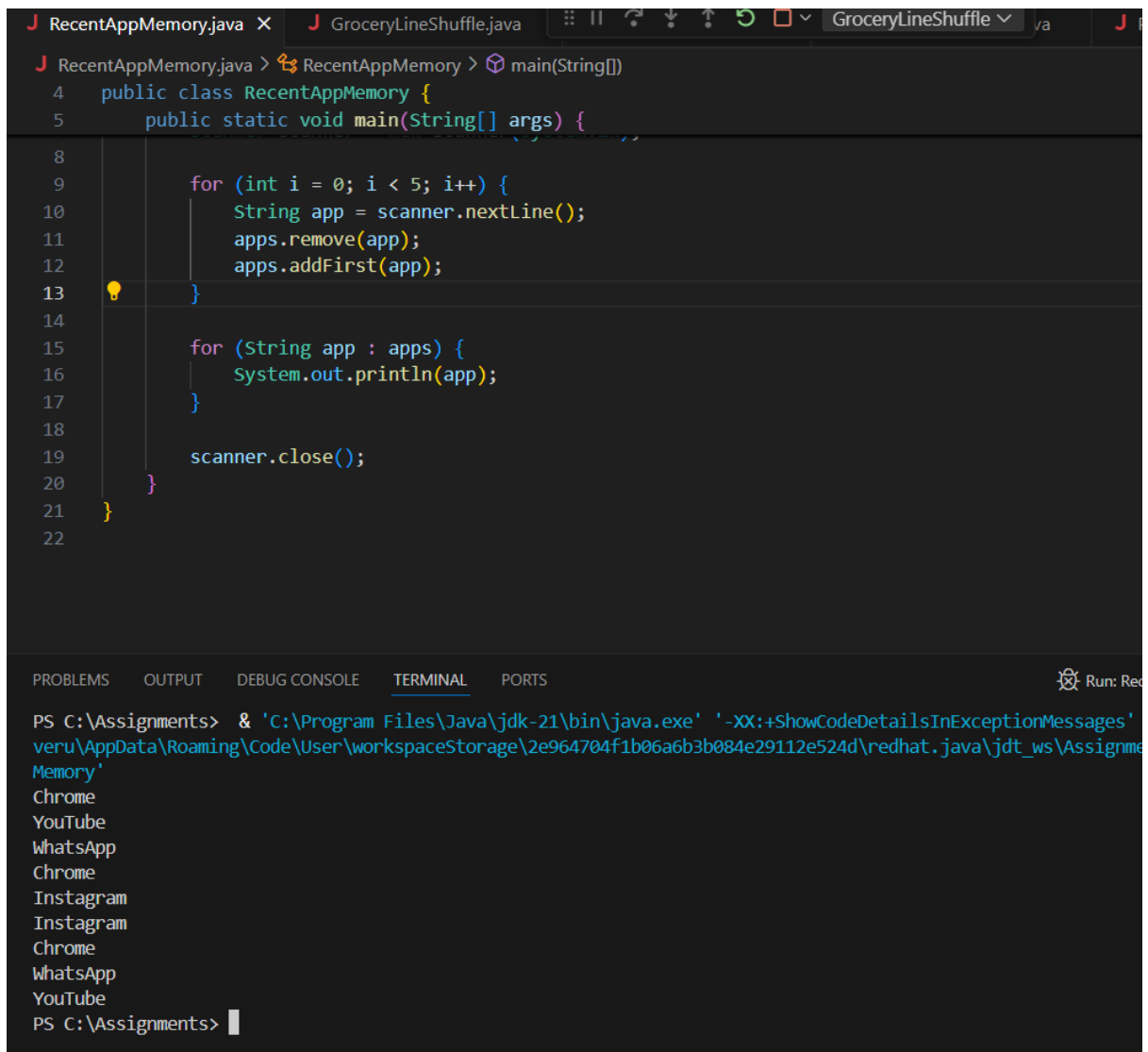
The screenshot displays an IDE window with the file `PrintManager.java` open. The code defines a `PrintManager` class with a `main` method. The `main` method uses a `Scanner` to read input, a `Queue` to store jobs, and a `while` loop to process them. The output in the terminal shows the sequence of jobs: Police, Business, Doctor, Teacher, and Lawyer, with some being skipped due to the queue's capacity.

```
PrintManager.java > PrintManager > main(String[])
1  import java.util.Scanner;
2  import java.util.concurrent.ArrayBlockingQueue;
3  import java.util.Queue;
4
5  public class PrintManager {
6      public static void main(String[] args) {
7          Queue<String> printQueue = new ArrayBlockingQueue<>(capacity:3);
8          Scanner scanner = new Scanner(System.in);
9
10         for (int i = 0; i < 5; i++) {
11             String job = scanner.nextLine();
12             boolean added = printQueue.offer(job);
13             if (!added) {
14                 System.out.println("Skipped: " + job);
15             }
16         }
17
18         while (!printQueue.isEmpty()) {
19             System.out.println("Printing: " + printQueue.poll());
20         }
21     }
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\Assignm
ger'
Doctor
Teacher
Lawyer
Police
Skipped: Police
Business
Skipped: Business
Printing: Doctor
Printing: Teacher
Printing: Lawyer
PS C:\Assignments>
```

Figure 7PrintManager

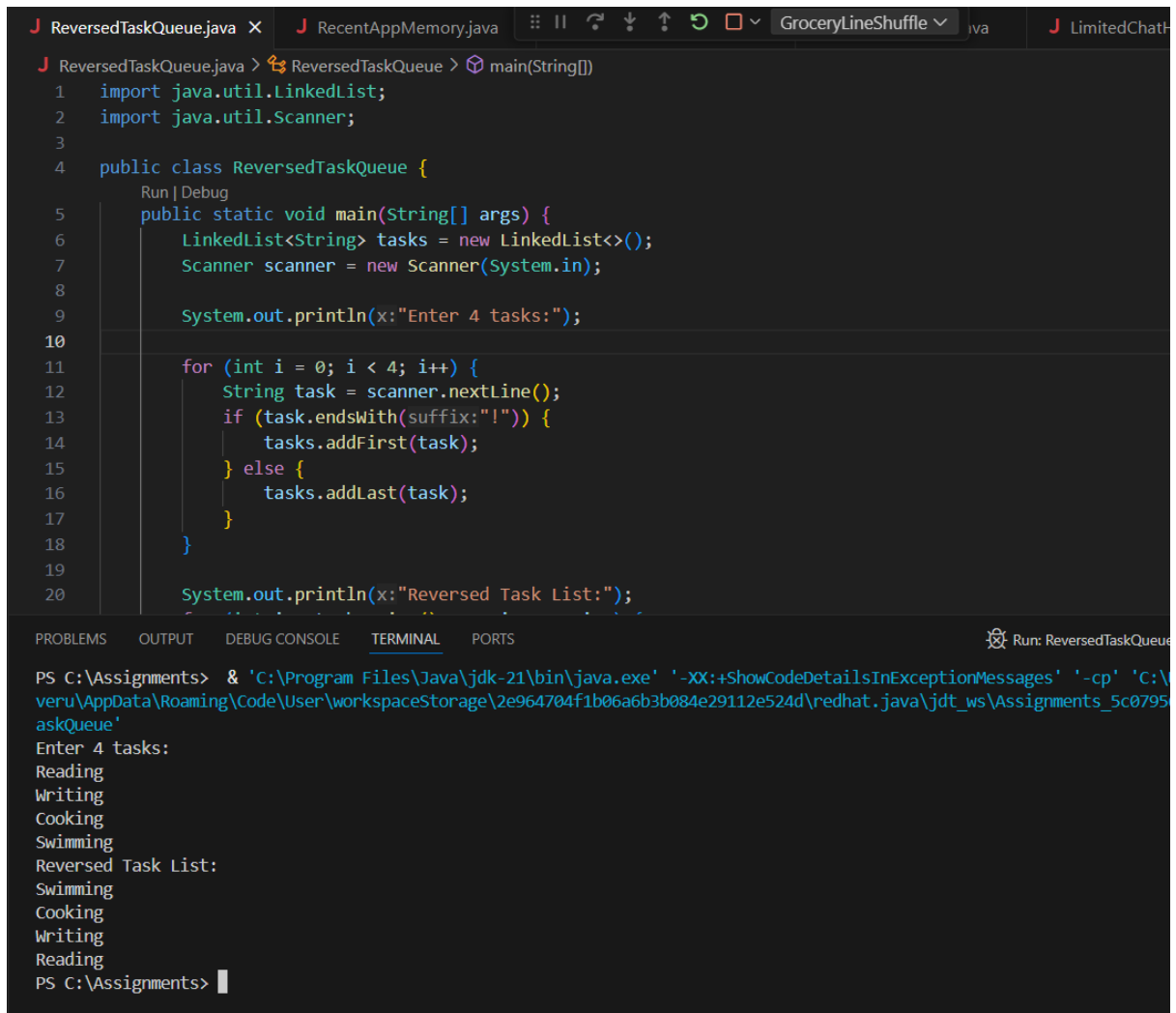


The screenshot shows an IDE with two tabs: 'RecentAppMemory.java' and 'GroceryLineShuffle.java'. The 'RecentAppMemory.java' tab is active, displaying the following code:

```
1 public class RecentAppMemory {
2     public static void main(String[] args) {
3
4         for (int i = 0; i < 5; i++) {
5             String app = scanner.nextLine();
6             apps.remove(app);
7             apps.addFirst(app);
8         }
9
10        for (String app : apps) {
11            System.out.println(app);
12        }
13
14        scanner.close();
15    }
16 }
```

The terminal output shows the execution of the program, displaying a list of applications: Chrome, YouTube, WhatsApp, Chrome, Instagram, Instagram, Chrome, WhatsApp, YouTube. The terminal prompt is 'PS C:\Assignments>'.

Figure 8RecentAppMemory



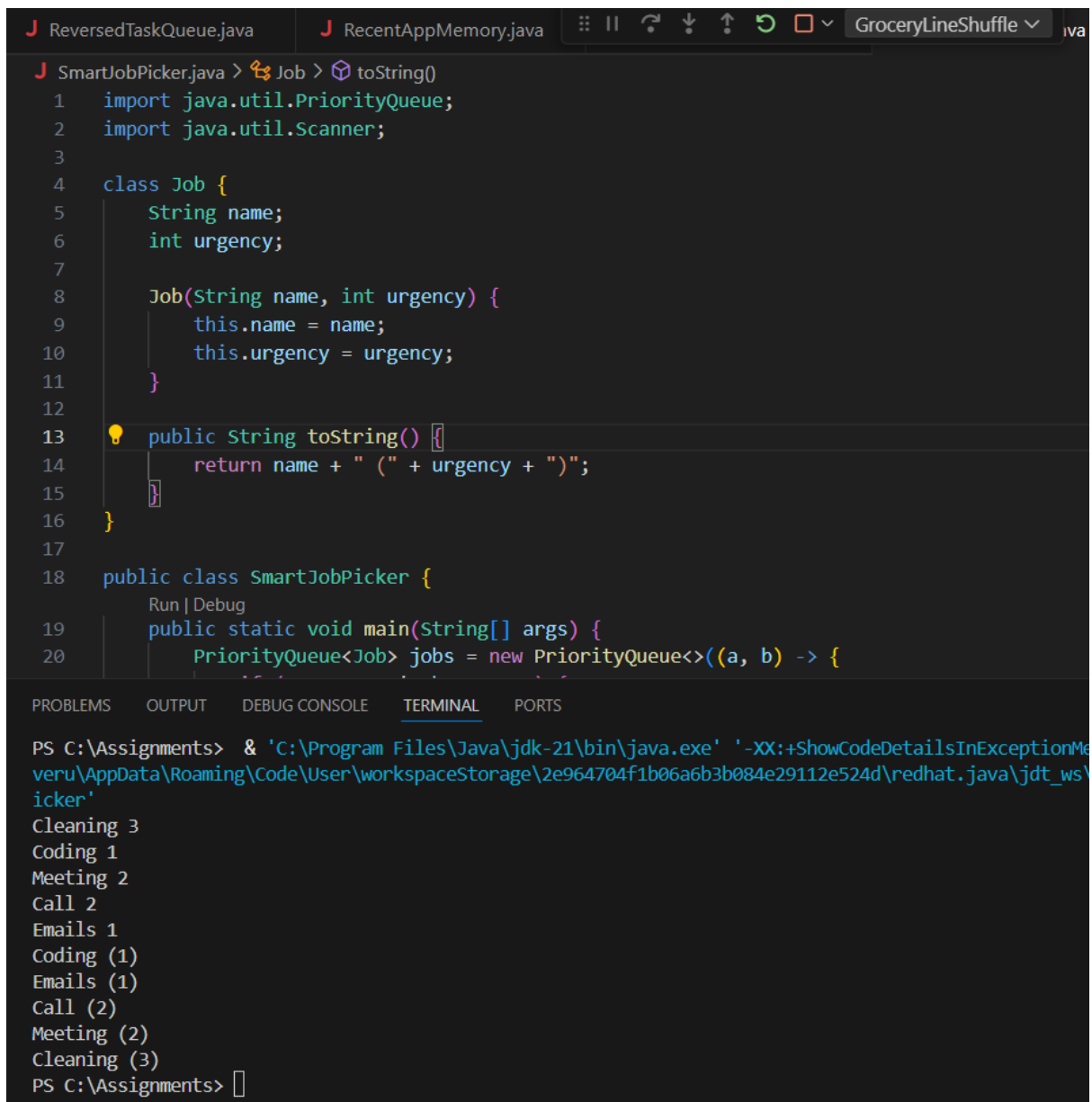
The screenshot displays an IDE window with the file `ReversedTaskQueue.java` open. The code defines a `ReversedTaskQueue` class with a `main` method that uses a `LinkedList` to store tasks. The tasks are entered via a `Scanner` and added to the list. The list is then printed in reverse order.

```
1 import java.util.LinkedList;
2 import java.util.Scanner;
3
4 public class ReversedTaskQueue {
5     public static void main(String[] args) {
6         LinkedList<String> tasks = new LinkedList<>();
7         Scanner scanner = new Scanner(System.in);
8
9         System.out.println(x:"Enter 4 tasks:");
10
11         for (int i = 0; i < 4; i++) {
12             String task = scanner.nextLine();
13             if (task.endsWith(suffix:"!")) {
14                 tasks.addFirst(task);
15             } else {
16                 tasks.addLast(task);
17             }
18         }
19
20         System.out.println(x:"Reversed Task List:");
```

The terminal output shows the execution of the program. It prompts the user to enter 4 tasks, which are entered as "Reading", "Writing", "Cooking", and "Swimming". The program then prints the reversed task list: "Swimming", "Cooking", "Writing", and "Reading".

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\Assignments_5c0795askQueue'
Enter 4 tasks:
Reading
Writing
Cooking
Swimming
Reversed Task List:
Swimming
Cooking
Writing
Reading
PS C:\Assignments>
```

Figure 9ReversedTaskQueue



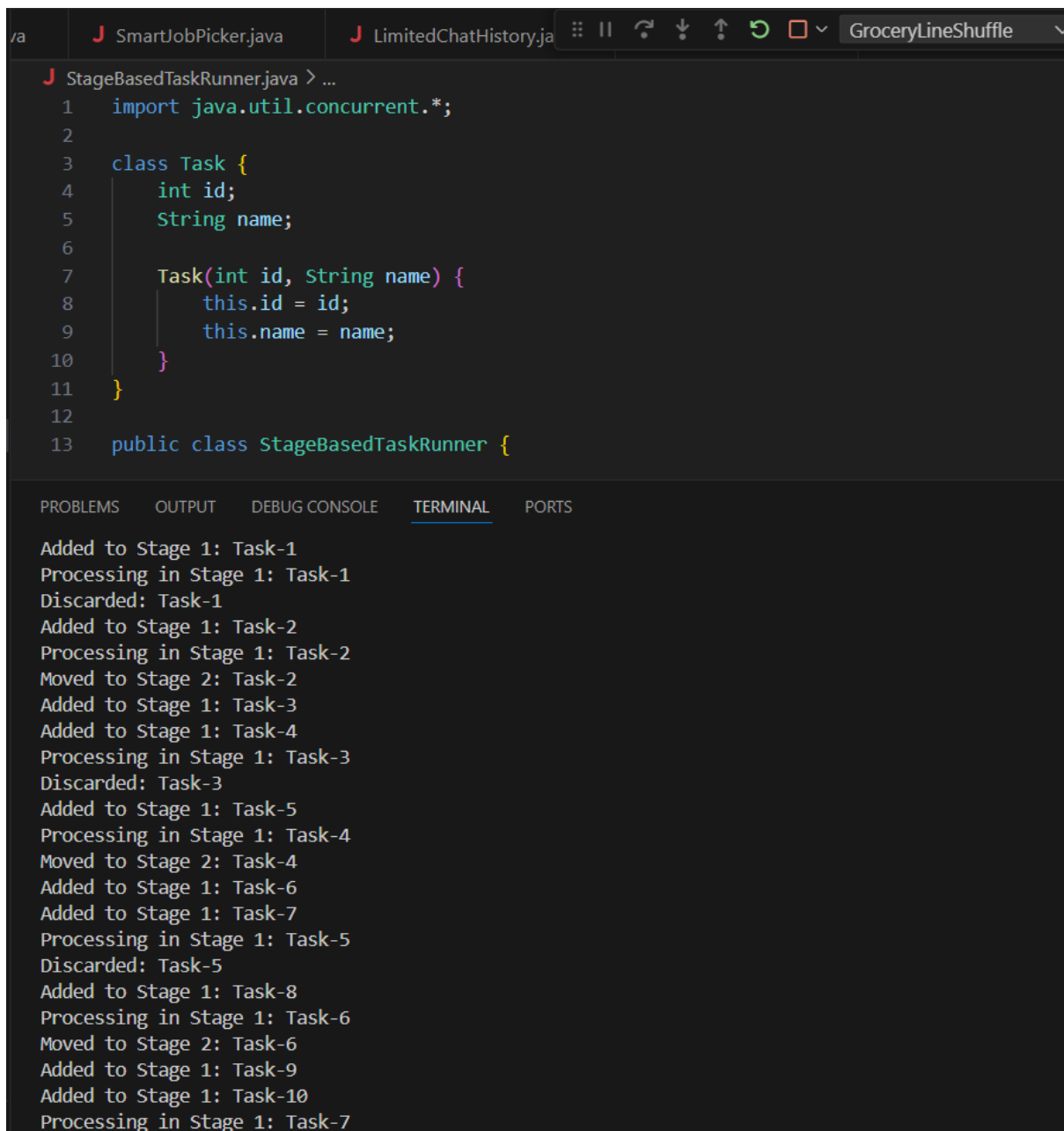
The screenshot shows an IDE with the following components:

- Top Bar:** Tabs for `ReversedTaskQueue.java`, `RecentAppMemory.java`, and `GroceryLineShuffle`. Icons for running, debugging, and other IDE functions are visible.
- Editor:** Displays `SmartJobPicker.java`. The code includes imports for `java.util.PriorityQueue` and `java.util.Scanner`. It defines a `Job` class with `name` and `urgency` attributes, a constructor, and a `toString()` method. The `SmartJobPicker` class contains a `main` method that initializes a `PriorityQueue` and starts adding jobs.
- Terminal:** Shows the command prompt output. It displays the full path to the Java executable, the command to run the program, and the resulting output: `Cleaning 3`, `Coding 1`, `Meeting 2`, `Call 2`, `Emails 1`, `Coding (1)`, `Emails (1)`, `Call (2)`, `Meeting (2)`, and `Cleaning (3)`.

```
1 import java.util.PriorityQueue;
2 import java.util.Scanner;
3
4 class Job {
5     String name;
6     int urgency;
7
8     Job(String name, int urgency) {
9         this.name = name;
10        this.urgency = urgency;
11    }
12
13    public String toString() {
14        return name + " (" + urgency + ")";
15    }
16 }
17
18 public class SmartJobPicker {
19     public static void main(String[] args) {
20         PriorityQueue<Job> jobs = new PriorityQueue<>((a, b) -> {
```

PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' -jar 'C:\Users\veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.java\jdt_ws\SmartJobPicker'
Cleaning 3
Coding 1
Meeting 2
Call 2
Emails 1
Coding (1)
Emails (1)
Call (2)
Meeting (2)
Cleaning (3)
PS C:\Assignments>

Figure 10 SmartJobPicker



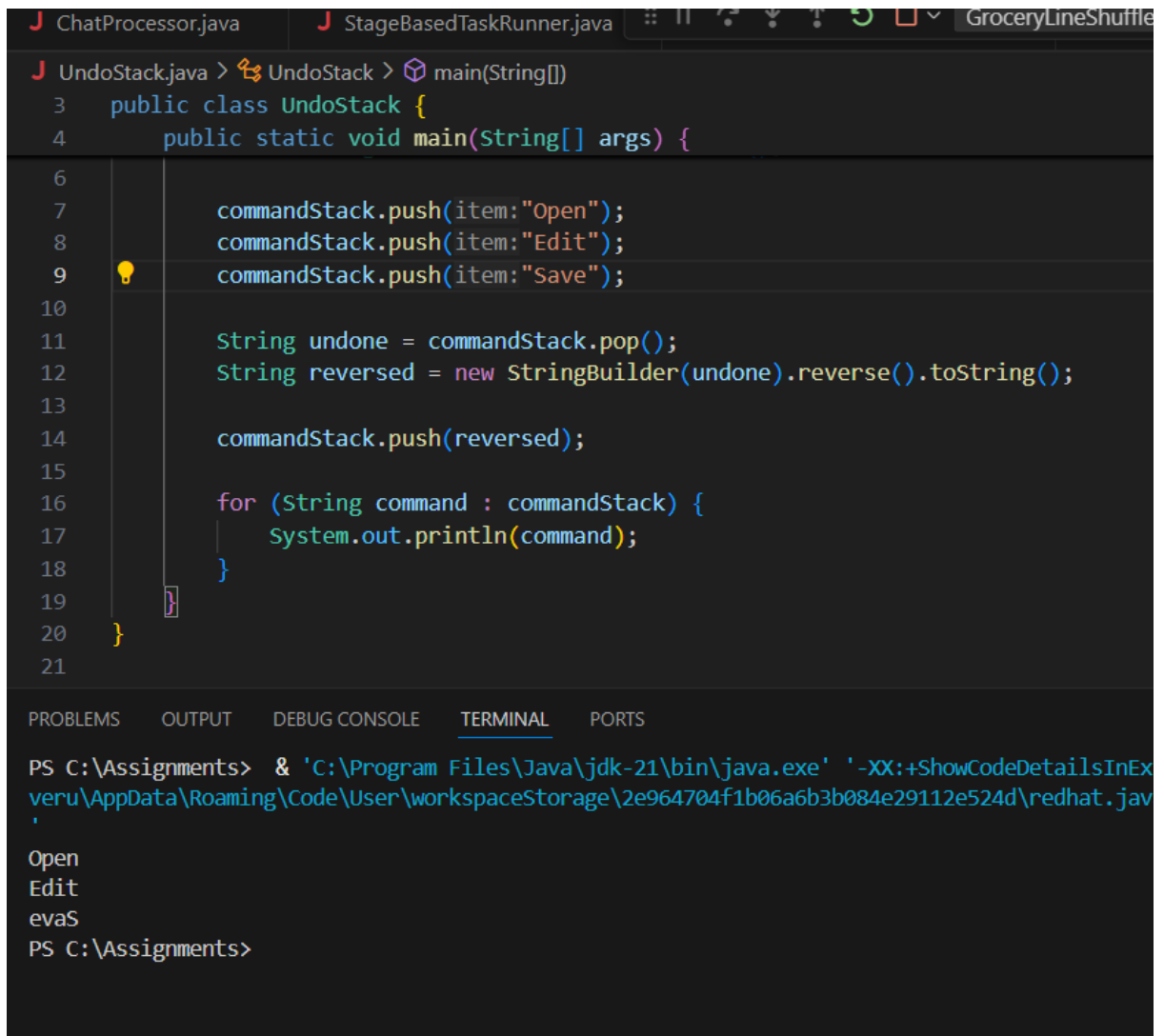
The screenshot shows an IDE with the file `StageBasedTaskRunner.java` open. The code defines a `Task` class with `id` and `name` attributes, and a `StageBasedTaskRunner` class. The terminal output shows the execution of the program, displaying the flow of tasks through different stages, including additions, processing, and discards.

```
StageBasedTaskRunner.java > ...
1  import java.util.concurrent.*;
2
3  class Task {
4      int id;
5      String name;
6
7      Task(int id, String name) {
8          this.id = id;
9          this.name = name;
10     }
11 }
12
13 public class StageBasedTaskRunner {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Added to Stage 1: Task-1
Processing in Stage 1: Task-1
Discarded: Task-1
Added to Stage 1: Task-2
Processing in Stage 1: Task-2
Moved to Stage 2: Task-2
Added to Stage 1: Task-3
Added to Stage 1: Task-4
Processing in Stage 1: Task-3
Discarded: Task-3
Added to Stage 1: Task-5
Processing in Stage 1: Task-4
Moved to Stage 2: Task-4
Added to Stage 1: Task-6
Added to Stage 1: Task-7
Processing in Stage 1: Task-5
Discarded: Task-5
Added to Stage 1: Task-8
Processing in Stage 1: Task-6
Moved to Stage 2: Task-6
Added to Stage 1: Task-9
Added to Stage 1: Task-10
Processing in Stage 1: Task-7
```

Figure 11 *StageBasedTaskRunner*



The screenshot shows an IDE with several tabs at the top: ChatProcessor.java, StageBasedTaskRunner.java, and GroceryLineShuffle. The active tab is UndoStack.java, showing the following code:

```
3 public class UndoStack {
4     public static void main(String[] args) {
5
6         commandStack.push(item:"Open");
7         commandStack.push(item:"Edit");
8         commandStack.push(item:"Save");
9
10        String undone = commandStack.pop();
11        String reversed = new StringBuilder(undone).reverse().toString();
12
13        commandStack.push(reversed);
14
15        for (String command : commandStack) {
16            System.out.println(command);
17        }
18    }
19 }
20
21
```

Below the code editor, the TERMINAL tab is active, showing the command prompt output:

```
PS C:\Assignments> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInEx
veru\AppData\Roaming\Code\User\workspaceStorage\2e964704f1b06a6b3b084e29112e524d\redhat.jav
,
Open
Edit
evaS
PS C:\Assignments>
```

Figure 12UndoStack