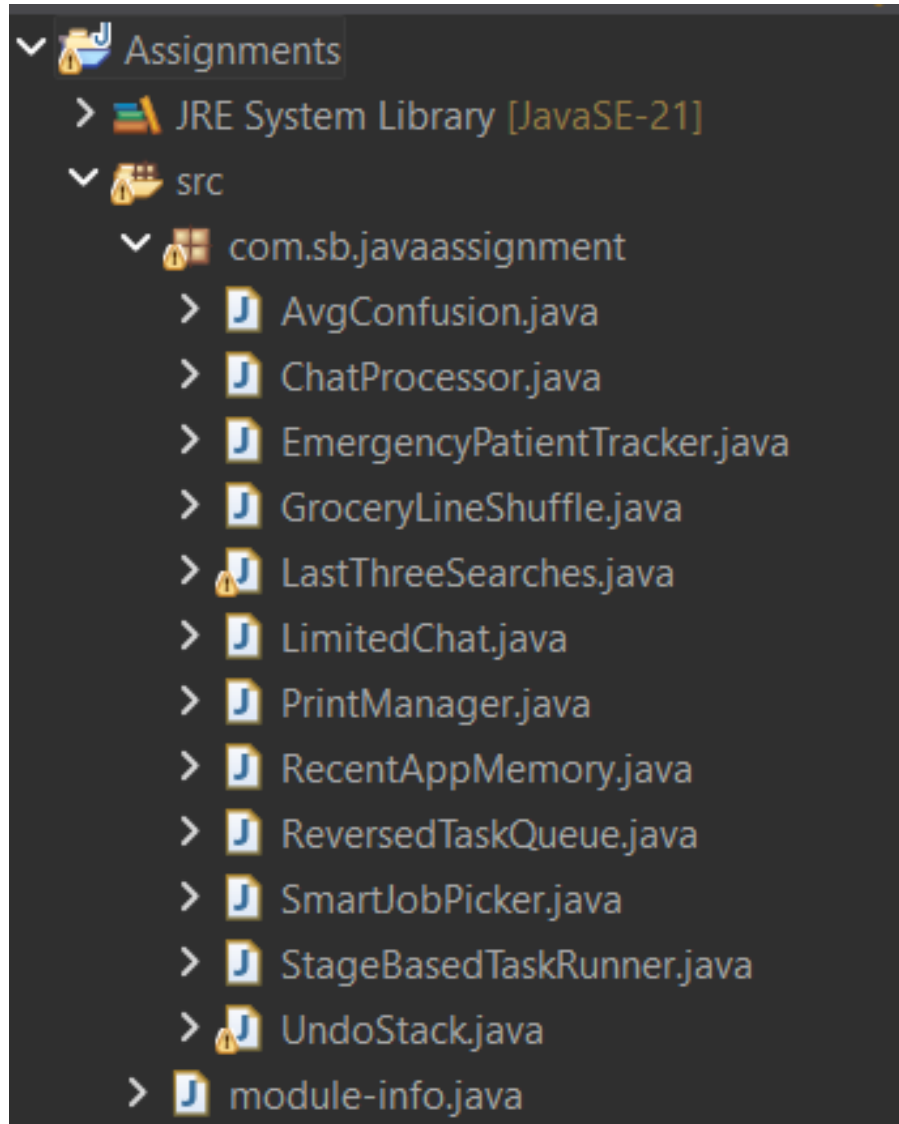
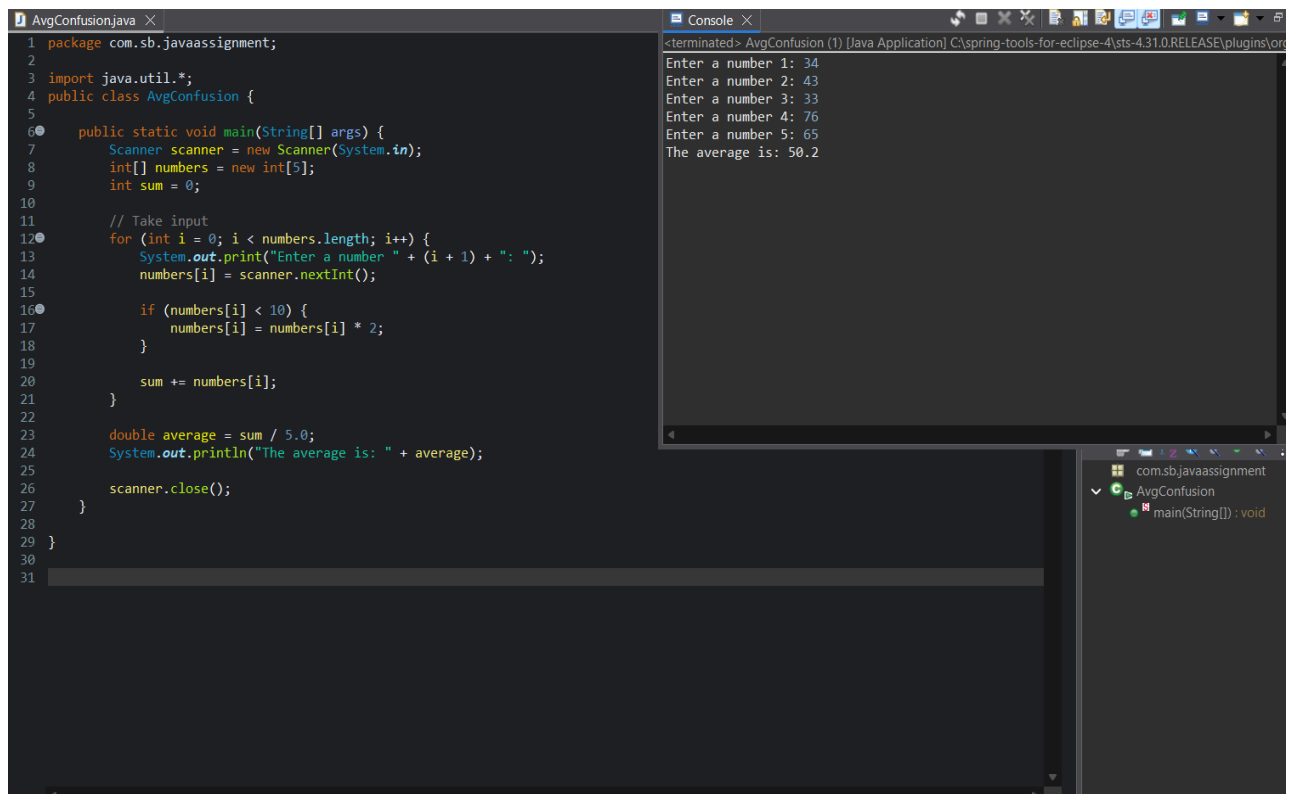


Core Java & Collections

Project Structure:



Code and Their Outputs:



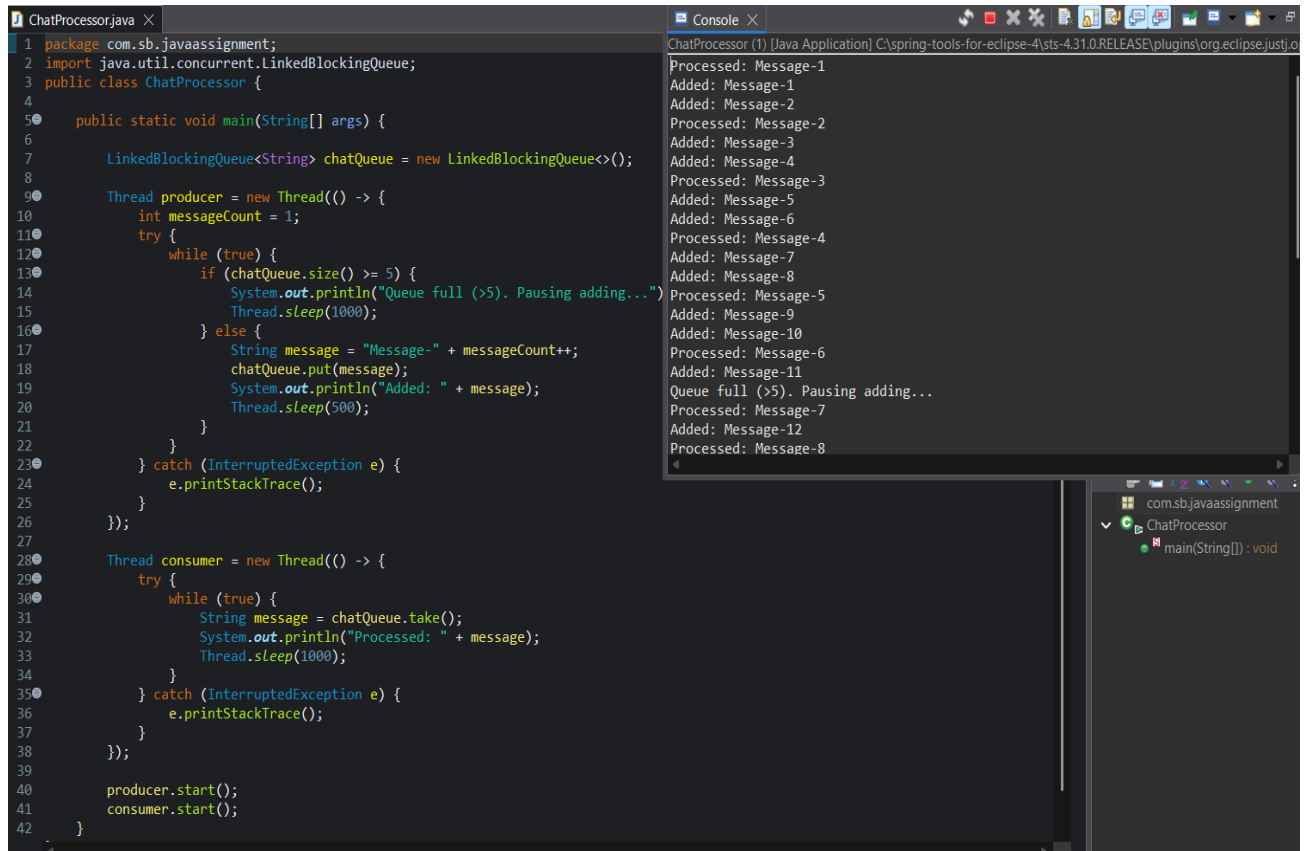
The screenshot shows the Eclipse IDE with the file `AvgConfusion.java` open. The code is as follows:

```
1 package com.sb.javaassignment;
2
3 import java.util.*;
4 public class AvgConfusion {
5
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8         int[] numbers = new int[5];
9         int sum = 0;
10
11         // Take input
12         for (int i = 0; i < numbers.length; i++) {
13             System.out.print("Enter a number " + (i + 1) + ": ");
14             numbers[i] = scanner.nextInt();
15
16             if (numbers[i] < 10) {
17                 numbers[i] = numbers[i] * 2;
18             }
19
20             sum += numbers[i];
21         }
22
23         double average = sum / 5.0;
24         System.out.println("The average is: " + average);
25
26         scanner.close();
27     }
28 }
29
30
31
```

The console output shows the execution of the program:

```
<terminated> AvgConfusion (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org
Enter a number 1: 34
Enter a number 2: 43
Enter a number 3: 33
Enter a number 4: 76
Enter a number 5: 65
The average is: 50.2
```

The project explorer on the right shows the package `com.sb.javaassignment` containing the class `AvgConfusion` with the `main(String[]) : void` method.



The screenshot shows the Eclipse IDE with the file `ChatProcessor.java` open. The code is as follows:

```
1 package com.sb.javaassignment;
2 import java.util.concurrent.LinkedBlockingQueue;
3 public class ChatProcessor {
4
5     public static void main(String[] args) {
6
7         LinkedBlockingQueue<String> chatQueue = new LinkedBlockingQueue<>();
8
9         Thread producer = new Thread(() -> {
10             int messageCount = 1;
11             try {
12                 while (true) {
13                     if (chatQueue.size() >= 5) {
14                         System.out.println("Queue full (>5). Pausing adding...");
15                         Thread.sleep(1000);
16                     } else {
17                         String message = "Message-" + messageCount++;
18                         chatQueue.put(message);
19                         System.out.println("Added: " + message);
20                         Thread.sleep(500);
21                     }
22                 }
23             } catch (InterruptedException e) {
24                 e.printStackTrace();
25             }
26         });
27
28         Thread consumer = new Thread(() -> {
29             try {
30                 while (true) {
31                     String message = chatQueue.take();
32                     System.out.println("Processed: " + message);
33                     Thread.sleep(1000);
34                 }
35             } catch (InterruptedException e) {
36                 e.printStackTrace();
37             }
38         });
39
40         producer.start();
41         consumer.start();
42     }
43 }
44
```

The console output shows the execution of the program:

```
ChatProcessor (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.eclipse.justi.e
Processed: Message-1
Added: Message-1
Added: Message-2
Processed: Message-2
Added: Message-3
Added: Message-4
Processed: Message-3
Added: Message-5
Added: Message-6
Processed: Message-4
Added: Message-7
Added: Message-8
Processed: Message-5
Added: Message-9
Added: Message-10
Processed: Message-6
Added: Message-11
Queue full (>5). Pausing adding...
Processed: Message-7
Added: Message-12
Processed: Message-8
```

The project explorer on the right shows the package `com.sb.javaassignment` containing the class `ChatProcessor` with the `main(String[]) : void` method.

```
EmergencyPatientTracker.java X
1 package com.sb.javaassignment;
2 import java.util.PriorityQueue;
3 import java.util.Comparator;
4
5 class Patient {
6     int severity;
7     long arrivalTime;
8
9     String name;
10
11     public Patient(String name, int severity, long arrivalTime) {
12         this.name = name;
13         this.severity = severity;
14         this.arrivalTime = arrivalTime;
15     }
16
17     @Override
18     public String toString() {
19         return name + " (Severity: " + severity + ")";
20     }
21 }
22
23 public class EmergencyPatientTracker {
24
25     public static void main(String[] args) throws InterruptedException {
26         // TODO Auto-generated method stub
27         PriorityQueue<Patient> patientQueue = new PriorityQueue<>(5, new Comparator<Patient>() {
28             @Override
29             public int compare(Patient p1, Patient p2) {
30                 if (p1.severity != p2.severity) {
31                     return Integer.compare(p1.severity, p2.severity); // lower severity = higher priority
32                 } else {
33                     return Long.compare(p1.arrivalTime, p2.arrivalTime); // earlier = higher priority
34                 }
35             }
36         });
37
38         // Add patients
39         addPatient(patientQueue, new Patient("Alice", 3, System.currentTimeMillis()));
40         Thread.sleep(100); // simulate time gap
41         addPatient(patientQueue, new Patient("Bob", 2, System.currentTimeMillis()));
42         Thread.sleep(100);
43     }
44
45     private static void addPatient(PriorityQueue<Patient> queue, Patient patient) {
46         if (queue.size() < 5) {
47             queue.add(patient);
48         } else {
49             System.out.println("Queue full. Cannot add: " + patient);
50         }
51     }
52 }

```

```
Console X
<terminated> EmergencyPatientTracker (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins
Added: Alice (Severity: 3)
Added: Bob (Severity: 2)
Added: Charlie (Severity: 2)
Added: Diana (Severity: 1)
Added: Ethan (Severity: 4)
Queue full. Cannot add: Frank (Severity: 1)

--- Treating patients ---
Treated: Diana (Severity: 1)
Treated: Bob (Severity: 2)
Treated: Charlie (Severity: 2)
Treated: Alice (Severity: 3)
Treated: Ethan (Severity: 4)

```

com.sb.javaassignment

- ▼ Patient
 - severity: int
 - arrivalTime: long
 - name: String
 - Patient(String, int, long)
 - toString(): String
- ▼ EmergencyPatientTracker
 - main(String[]): void
 - new Comparator() [...]
 - addPatient(PriorityQueue)
 - Patient

```
GroceryLineShuffle.java X
1 package com.sb.javaassignment;
2 import java.util.ArrayDeque;
3
4
5 public class GroceryLineShuffle {
6     public static void main(String[] args) {
7         ArrayDeque<String> line = new ArrayDeque<>();
8         addCustomer(line, "Ravi");
9         addCustomer(line, "Meena");
10        addCustomer(line, "John");
11        addCustomer(line, "Alex");
12
13        System.out.println("Grocery Queue: " + line);
14    }
15
16    static void addCustomer(ArrayDeque<String> line, String name) {
17        if (name.length() % 2 == 0)
18            line.addFirst(name);
19        else
20            line.addLast(name);
21    }
22 }

```

```
Console X
<terminated> GroceryLineShuffle (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins
Grocery Queue: [Alex, John, Ravi, Meena]

```

com.sb.javaassignment

- ▼ GroceryLineShuffle
 - main(String[]): void
 - addCustomer(ArrayDeque)

```
LastThreeSearches.java
1 package com.sb.javaassignment;
2 import java.util.*;
3 import java.util.ArrayDeque;
4
5 public class LastThreeSearches {
6
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         ArrayDeque<String> searchBox = new ArrayDeque<>();
10
11         for (int i = 0; i < 4; i++) {
12             System.out.print("Enter a search term: ");
13             String term = sc.nextLine();
14
15             if (searchBox.size() == 3) {
16                 searchBox.removeFirst();
17             }
18
19             searchBox.addLast(term);
20         }
21
22         System.out.println("Last 3 searches:");
23         for (String term : searchBox) {
24             System.out.println(term);
25         }
26     }
27 }
28
```

```
Console
<terminated> LastThreeSearches (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org
Enter a search term: sanket
Enter a search term: sai
Enter a search term: sai
Last 3 searches:
sanket
sai
sai
```

com.sb.javaassignment
LastThreeSearches
main(String[]): void

```
LimitedChat.java
1 package com.sb.javaassignment.LimitedChat;
2
3 import java.util.*;
4 import java.util.ArrayDeque;
5 import java.util.Deque;
6
7 public class LimitedChat {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         Scanner scanner = new Scanner(System.in);
12         Deque<String> chatHistory = new ArrayDeque<>();
13
14         for (int i = 0; i < 6; i++) {
15             System.out.print("Enter message " + (i + 1) + ": ");
16             String message = scanner.nextLine();
17
18             // If already 4 messages, remove oldest
19             if (chatHistory.size() == 4) {
20                 chatHistory.removeFirst();
21             }
22
23             chatHistory.addLast(message);
24
25             System.out.println("Current chat history: " + chatHistory);
26         }
27
28         scanner.close();
29     }
30 }
31
32
33
34
```

```
Console
<terminated> LimitedChat (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org
Enter message 1: hello
Current chat history: [hello]
Enter message 2: How are you!
Current chat history: [hello, How are you!]
Enter message 3: what are doing?
Current chat history: [hello, How are you!, what are doing?]
Enter message 4: everthing is fine?
Current chat history: [hello, How are you!, what are doing?, everthing is fine?]
Enter message 5: okay?
Current chat history: [How are you!, what are doing?, everthing is fine?, okay?]
Enter message 6: no woories
Current chat history: [what are doing?, everthing is fine?, okay?, no woories]
```

com.sb.javaassignment
LimitedChat
main(String[]): void

```
PrintManager.java X Console X
1 package com.sb.javaassignment;
2 import java.util.concurrent.ArrayBlockingQueue;
3
4 public class PrintManager {
5     public static void main(String[] args) {
6
7         ArrayBlockingQueue<String> printQueue = new ArrayBlockingQueue<>(3);
8
9         addJob(printQueue, "Job1");
10        addJob(printQueue, "Job2");
11        addJob(printQueue, "Job3");
12
13        addJob(printQueue, "Job4");
14
15        while (!printQueue.isEmpty()) {
16            String job = printQueue.poll();
17            System.out.println("Printing: " + job);
18        }
19    }
20
21    static void addJob(ArrayBlockingQueue<String> queue, String job) {
22        if (!queue.offer(job)) {
23            System.out.println("Queue full. Skipping: " + job);
24        } else {
25            System.out.println("Added: " + job);
26        }
27    }
28 }
29
```

```
<terminated> PrintManager (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.ec
Added: Job1
Added: Job2
Added: Job3
Queue full. Skipping: Job4
Printing: Job1
Printing: Job2
Printing: Job3
```

com.sb.javaassignment
PrintManager
main(String[]): void
addJob(ArrayBlockingQueue<String>, String): void

```
RecentAppMemory.java X Console X
1 package com.sb.javaassignment;
2 import java.util.*;
3 import java.util.LinkedList;
4
5 public class RecentAppMemory {
6
7     public static void main(String[] args) {
8         Scanner scanner = new Scanner(System.in);
9         LinkedList<String> recentApps = new LinkedList<>();
10
11        for (int i = 0; i < 5; i++) {
12            System.out.print("Open app " + (i + 1) + ": ");
13            String app = scanner.nextLine();
14
15            if (recentApps.contains(app)) {
16                recentApps.remove(app);
17            }
18
19            recentApps.addFirst(app);
20
21            System.out.println("Current Recent Apps: " + recentApps);
22        }
23
24        System.out.println("\nFinal Recent Apps: " + recentApps);
25
26        scanner.close();
27    }
28 }
29
30 }
31
```

```
<terminated> RecentAppMemory (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\
Open app 1: Netflix
Current Recent Apps: [Netflix]
Open app 2: Swiggy
Current Recent Apps: [Swiggy, Netflix]
Open app 3: Amazon
Current Recent Apps: [Amazon, Swiggy, Netflix]
Open app 4: Flipkart
Current Recent Apps: [Flipkart, Amazon, Swiggy, Netflix]
Open app 5: Tinder
Current Recent Apps: [Tinder, Flipkart, Amazon, Swiggy, Netflix]
Final Recent Apps: [Tinder, Flipkart, Amazon, Swiggy, Netflix]
```

com.sb.javaassignment
RecentAppMemory
main(String[]): void

```
ReversedTaskQueue.java
1 package com.sb.javaassignment;
2 import java.util.*;
3 import java.util.LinkedList;
4 import java.util.ListIterator;
5
6 public class ReversedTaskQueue {
7
8     public static void main(String[] args) {
9         Scanner scanner = new Scanner(System.in);
10        LinkedList<String> tasks = new LinkedList<>();
11
12        // Take 4 task inputs
13        for (int i = 0; i < 4; i++) {
14            System.out.print("Enter task " + (i + 1) + ": ");
15            String task = scanner.nextLine();
16
17            if (task.endsWith("!")) {
18                tasks.addFirst(task);
19            } else {
20                tasks.addLast(task);
21            }
22        }
23
24        // Print tasks in reverse
25        System.out.println("\nTasks in reverse order:");
26        ListIterator<String> iterator = tasks.listIterator(tasks.size());
27        while (iterator.hasPrevious()) {
28            System.out.println(iterator.previous());
29        }
30
31        scanner.close();
32    }
33 }
34
35 }
36
```

```
Console
<terminated> ReversedTaskQueue (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\
Enter task 1: Apple
Enter task 2: Mango
Enter task 3: Grapes
Enter task 4: Banana

Tasks in reverse order:
Banana
Grapes
Mango
Apple
```

com.sb.javaassignment
ReversedTaskQueue
main(String[]): void

```
SmartJobPicker.java
1 package com.sb.javaassignment;
2 import java.util.*;
3 import java.util.PriorityQueue;
4
5 class Job {
6     String name;
7     int urgency;
8
9     public Job(String name, int urgency) {
10        this.name = name;
11        this.urgency = urgency;
12    }
13
14    @Override
15    public String toString() {
16        return name + " (Urgency: " + urgency + ")";
17    }
18 }
19
20 public class SmartJobPicker {
21
22     public static void main(String[] args) {
23         PriorityQueue<Job> jobQueue = new PriorityQueue<>(
24             Comparator.comparingInt((Job j) -> j.urgency)
25                 .thenComparing(j -> j.name.length())
26         );
27
28         jobQueue.add(new Job("Fix", 2));
29         jobQueue.add(new Job("Deploy", 1));
30         jobQueue.add(new Job("Audit", 1));
31         jobQueue.add(new Job("Build", 3));
32
33         while (!jobQueue.isEmpty()) {
34             System.out.println("Picked: " + jobQueue.poll());
35         }
36     }
37 }
38
```

```
Console
<terminated> SmartJobPicker (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org
Picked: Audit (Urgency: 1)
Picked: Deploy (Urgency: 1)
Picked: Fix (Urgency: 2)
Picked: Build (Urgency: 3)
```

com.sb.javaassignment
Job
name: String
urgency: int
Job(String, int)
toString(): String
SmartJobPicker
main(String[]): void

```
StageBasedTaskRunner.java
1 package com.sb.javaassignment;
2
3 import java.util.concurrent.*;
4
5 class Task {
6     int id;
7     String desc;
8
9     Task(int id, String desc) {
10         this.id = id;
11         this.desc = desc;
12     }
13
14     public String toString() {
15         return "Task#" + id + " - " + desc;
16     }
17 }
18
19 public class StageBasedTaskRunner {
20     public static void main(String[] args) {
21         LinkedBlockingQueue<Task> stage1 = new LinkedBlockingQueue<>();
22         LinkedBlockingQueue<Task> stage2 = new LinkedBlockingQueue<>();
23
24         stage1.add(new Task(1, "Load"));
25         stage1.add(new Task(2, "Parse"));
26         stage1.add(new Task(3, "Validate"));
27         stage1.add(new Task(4, "Save"));
28
29         while (!stage1.isEmpty()) {
30             Task task = stage1.poll();
31             System.out.println("Stage 1: " + task);
32             if (task.id % 2 == 0) {
33                 stage2.add(task);
34             }
35         }
36
37         System.out.println("\nStage 2 Tasks:");
38         for (Task t : stage2) {
39             System.out.println("Stage 2: " + t);
40         }
41     }
42 }
```

```
Console
<terminated> StageBasedTaskRunner [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\
Stage 1: Task#1 - Load
Stage 1: Task#2 - Parse
Stage 1: Task#3 - Validate
Stage 1: Task#4 - Save

Stage 2 Tasks:
Stage 2: Task#2 - Parse
Stage 2: Task#4 - Save
```

com.sb.javaassignment

- Task
 - id : int
 - desc : String
 - Task(int, String)
 - toString() : String
- StageBasedTaskRunner
 - main(String[]) : void

```
UndoStack.java
1 package com.sb.javaassignment;
2
3 import java.util.Scanner;
4 import java.util.Stack;
5
6 public class UndoStack {
7
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10         Stack<String> commandStack = new Stack<>();
11
12         System.out.println("Enter 3 commands:");
13         for (int i = 0; i < 3; i++) {
14             commandStack.push(sc.nextLine());
15         }
16
17         // Undo (pop last command)
18         String undoneCommand = commandStack.pop();
19         System.out.println("Undone command: " + undoneCommand);
20
21         // Redo (push it back)
22         commandStack.push(undoneCommand);
23
24         System.out.println("Commands in stack now:");
25         for (int i = commandStack.size() - 1; i >= 0; i--) {
26             System.out.println(commandStack.get(i));
27         }
28     }
29 }
30 }
```

```
Console
<terminated> UndoStack (1) [Java Application] C:\spring-tools-for-eclipse-4\sts-4.31.0.RELEASE\plugins\org.eclip
Enter 3 commands:
Create
Update
Delete
Undone command: Delete
Commands in stack now:
Delete
Update
Create
```

com.sb.javaassignment

- UndoStack
 - main(String[]) : void