



UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

TAREA

QUEUE DE CANCIONES

MILDRED HANANI PINEDA PINEDA
0905-22-5811

PROGRAMACION 3

FEBRERO 2026

Parte A — Librería de Cola Propia

PASO 1

Verificar que versión de **Maven** y que versión de **Java** esta instalado con los siguientes comandos

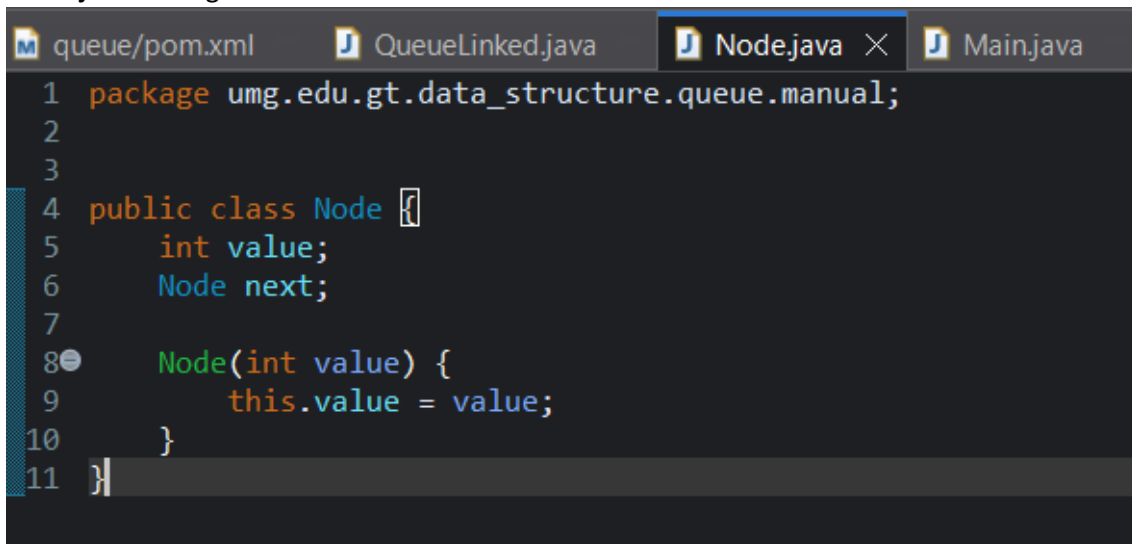
java -version

mvn -version

```
PS C:\Users\COMPUFIRE> java -version
openjdk version "11.0.30" 2026-01-20
OpenJDK Runtime Environment Temurin-11.0.30+7 (build 11.0.30+7)
OpenJDK 64-Bit Server VM Temurin-11.0.30+7 (build 11.0.30+7, mixed mode)
PS C:\Users\COMPUFIRE> mvn -version
Apache Maven 3.9.12 (848fbb4bf2d427b72bdb2471c22fced7ebd9a7a1)
Maven home: C:\Maven\apache-maven-3.9.12
Java version: 11.0.30, vendor: Eclipse Adoptium, runtime: C:\Program Files\Eclipse Adoptium\jdk-11.0.30.7-hotspot
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\COMPUFIRE>
```

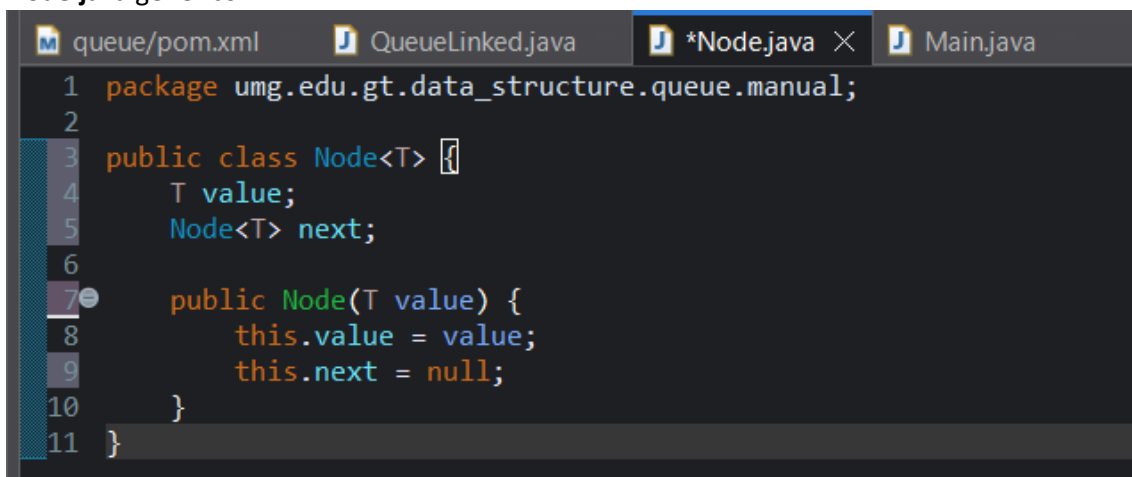
PASO 2

Node.java no es genérico



```
queue/pom.xml QueueLinked.java Node.java X Main.java
1 package umg.edu.gt.data_structure.queue.manual;
2
3
4 public class Node {
5     int value;
6     Node next;
7
8     Node(int value) {
9         this.value = value;
10    }
11 }
```


Node.java genérico



```
queue/pom.xml QueueLinked.java *Node.java X Main.java
1 package umg.edu.gt.data_structure.queue.manual;
2
3 public class Node<T> {
4     T value;
5     Node<T> next;
6
7     public Node(T value) {
8         this.value = value;
9         this.next = null;
10    }
11 }
```

PASO 3

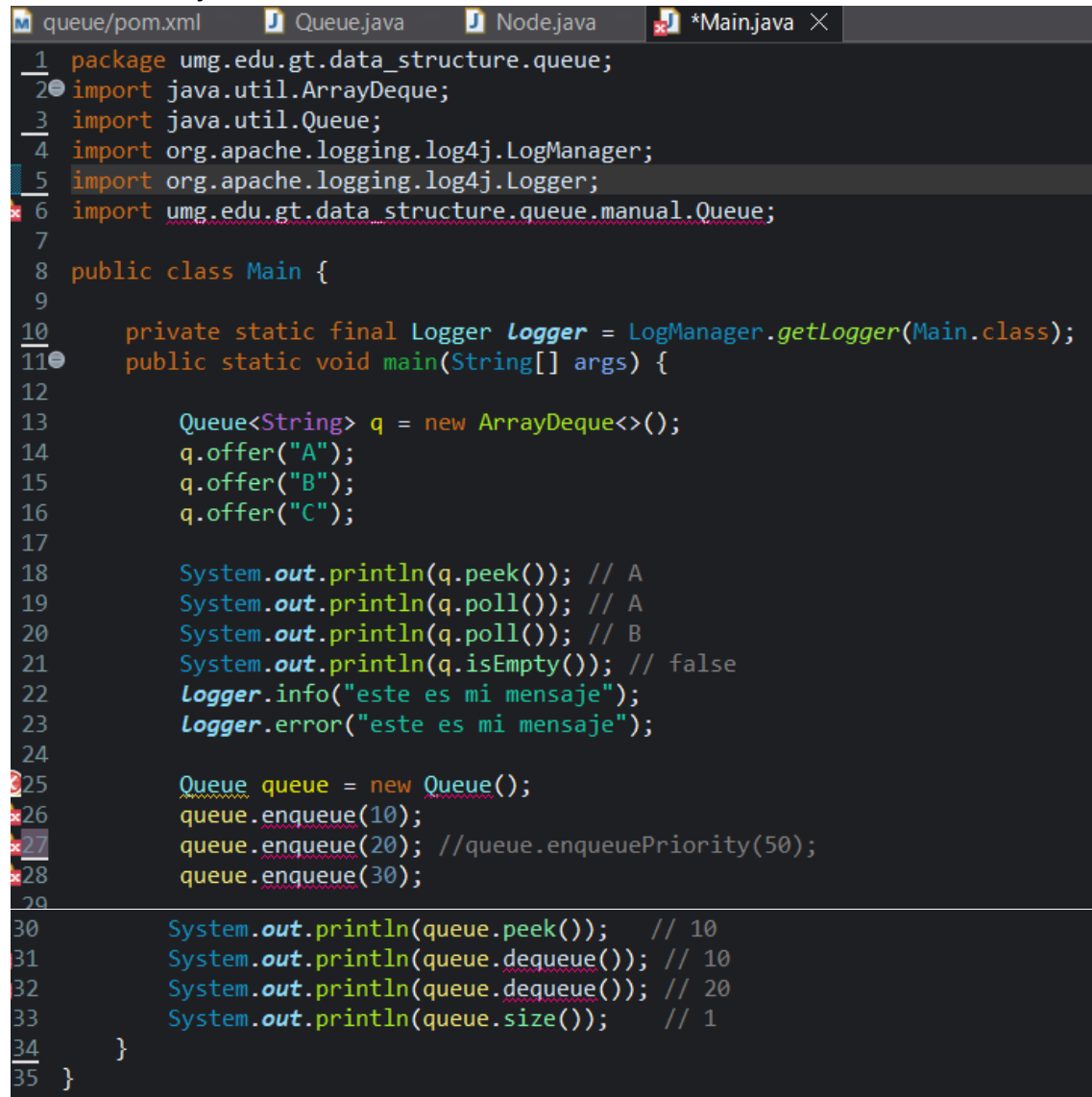
QueueLikened.java no genérico



```
1 package umg.edu.gt.data_structure.queue.manual;
2
3 public class QueueLinked {
4
5     private Node head; // primero en salir
6     private Node tail; // ultimo en entrar
7     private int size;
8
9     public boolean isEmpty() {
10         return size == 0;
11     }
12
13     public int size() {
14         return size;
15     }
16
17     // add
18     public void enqueue(int value) {
19         Node node = new Node(value);
20         if (isEmpty()) {
21             head = tail = node;
22         } else {
23             tail.next = node;
24             tail = node;
25         }
26         size++;
27     }
28
29     // delete
30     public int dequeue() {
31         if (isEmpty())
32             throw new IllegalStateException("Queue underflow");
33
34         int value = head.value;
35         head = head.next;
36         if (head == null) {
37             tail = null;
38         }
39         size--;
40         return value;
41     }
42
43     public int peek() {
44         if (isEmpty()) {
45             throw new IllegalStateException("Queue underflow");
46         }
47         return head.value;
48     }
49 }
50 }
```

QueueLikend.java genérico

```
queue/pom.xml  *QueueLinked.java  *Node.java  Main.java
1 package umg.edu.gt.data_structure.queue.manual;
2
3 public class QueueLinked<T> {
4
5     private Node<T> head; // primero en salir
6     private Node<T> tail; // ultimo en entrar
7     private int size;
8
9     public boolean isEmpty() {
10         return size == 0;
11     }
12
13     public int size() {
14         return size;
15     }
16
17     // Agregar elemento - O(1)
18     public void enqueue(T item) {
19         Node<T> node = new Node<>(item);
20         if (isEmpty()) {
21             head = tail = node;
22         } else {
23             tail.next = node;
24             tail = node;
25         }
26         size++;
27     }
28
29     // Eliminar y retornar el primero - O(1)
30     public T dequeue() {
31         if (isEmpty()) {
32             throw new IllegalStateException("Queue underflow: cannot dequeue from empty queue");
33         }
34
35         T value = head.value;
36         head = head.next;
37         if (head == null) {
38             tail = null;
39         }
40         size--;
41         return value;
42     }
43
44     // Ver el primero sin eliminar
45     public T peek() {
46         if (isEmpty()) {
47             throw new IllegalStateException("Queue underflow: cannot peek from empty queue");
48         }
49         return head.value;
50     }
51 }
```

PASO 4Actualizar **Main.java**

```
1 package umg.edu.gt.data_structure.queue;
2 import java.util.ArrayDeque;
3 import java.util.Queue;
4 import org.apache.logging.log4j.LogManager;
5 import org.apache.logging.log4j.Logger;
6 import umg.edu.gt.data_structure.queue.manual.Queue;
7
8 public class Main {
9
10     private static final Logger logger = LogManager.getLogger(Main.class);
11     public static void main(String[] args) {
12
13         Queue<String> q = new ArrayDeque<>();
14         q.offer("A");
15         q.offer("B");
16         q.offer("C");
17
18         System.out.println(q.peek()); // A
19         System.out.println(q.poll()); // A
20         System.out.println(q.poll()); // B
21         System.out.println(q.isEmpty()); // false
22         logger.info("este es mi mensaje");
23         logger.error("este es mi mensaje");
24
25         Queue queue = new Queue();
26         queue.enqueue(10);
27         queue.enqueue(20); //queue.enqueuePriority(50);
28         queue.enqueue(30);
29
30         System.out.println(queue.peek()); // 10
31         System.out.println(queue.dequeue()); // 10
32         System.out.println(queue.dequeue()); // 20
33         System.out.println(queue.size()); // 1
34     }
35 }
```

Actualizado Main.java

```

queue/pom.xml Queue.java Node.java *Main.java x
1 package umg.edu.gt.data_structure.queue;
2 import umg.edu.gt.data_structure.queue.manual.Queue;
3 public class Main {
4
5     public static void main(String[] args) {
6
7         // Prueba con Strings
8         Queue<String> queueStrings = new Queue<>();
9         queueStrings.enqueue("A");
10        queueStrings.enqueue("B");
11        queueStrings.enqueue("C");
12
13        System.out.println("=== Prueba con Strings ===");
14        System.out.println("Peek: " + queueStrings.peek()); // A
15        System.out.println("Dequeue: " + queueStrings.dequeue()); // A
16        System.out.println("Dequeue: " + queueStrings.dequeue()); // B
17        System.out.println("Size: " + queueStrings.size()); // 1
18        System.out.println("Dequeue: " + queueStrings.dequeue()); // C
19        System.out.println("IsEmpty: " + queueStrings.isEmpty()); // true
20
21        // Prueba con Integers
22        System.out.println("\n=== Prueba con Integers ===");
23        Queue<Integer> queueIntegers = new Queue<>();
24        queueIntegers.enqueue(10);
25        queueIntegers.enqueue(20);
26        queueIntegers.enqueue(30);
27
28        System.out.println("Peek: " + queueIntegers.peek()); // 10
29        System.out.println("Dequeue: " + queueIntegers.dequeue()); // 10
30        System.out.println("Size: " + queueIntegers.size()); // 2
31
32        System.out.println("\n=== Todas las pruebas completadas ===");
33    }
34 }

```

PASO 5

Instalar la librería localmente en **Maven**

Con el siguiente comando en la terminal **mvn clean install**

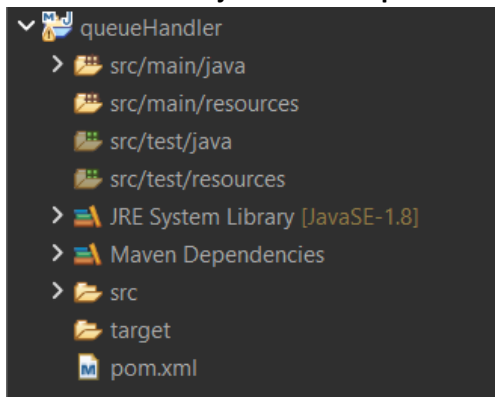
```

[INFO] --- surefire:3.2.5:test (default-test) @ queue ---
[INFO] --- jar:3.4.1:jar (default-jar) @ queue ---
[INFO] Building jar: C:\Users\COMPUFIRE\Downloads\queue\queue\target\queue-0.0.1-SNAPSHOT.jar
[INFO] --- install:3.1.2:install (default-install) @ queue ---
[INFO] Installing C:\Users\COMPUFIRE\Downloads\queue\queue\pom.xml to C:\Users\COMPUFIRE\.m2\repository\umg\edu\gt\data-structure\queue\queue\0.0.1-SNAPSHOT\queue-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\COMPUFIRE\Downloads\queue\queue\target\queue-0.0.1-SNAPSHOT.jar to C:\Users\COMPUFIRE\.m2\repository\umg\edu\gt\data-structure\queue\queue\0.0.1-SNAPSHOT\queue-0.0.1-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.047 s
[INFO] Finished at: 2026-02-24T11:56:16-06:00
[INFO] -----
PS C:\Users\COMPUFIRE\Downloads\queue\queue>

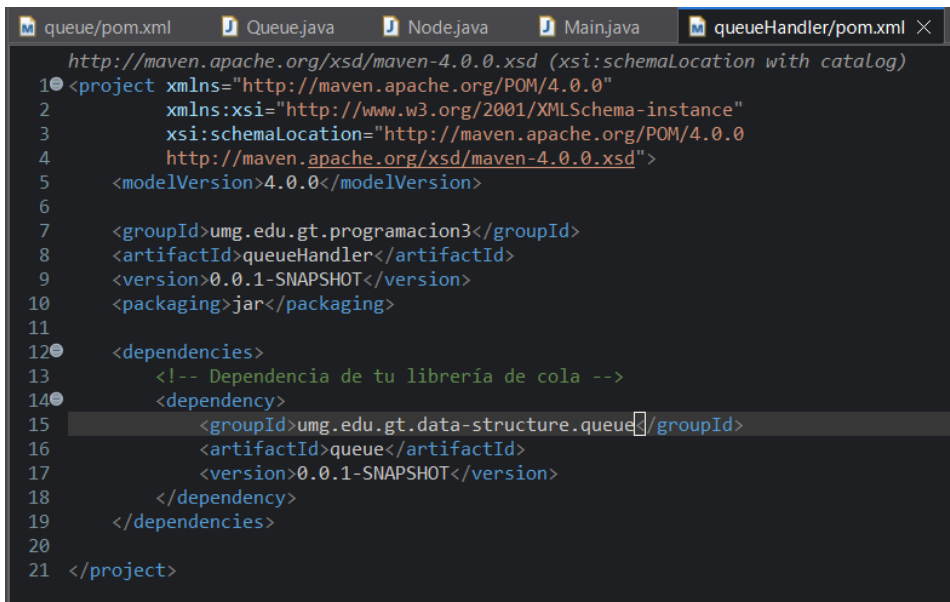
```

PASO 6

Crear un **New Project** llamado **queueHandler**

**PASO 7**

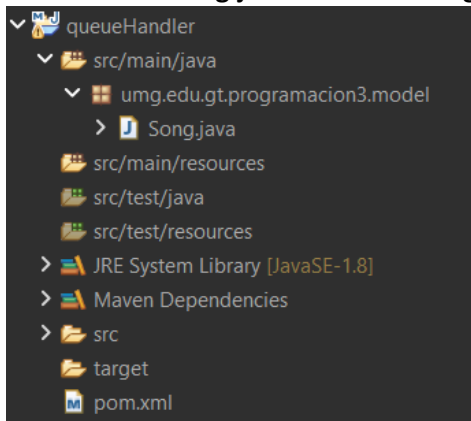
Agregar este código a **pom.xml**

**PASO 8**

En el Package Explorer, haz clic derecho sobre **src/main/java** (dentro del proyecto **queueHandler**) Selecciona **New** → **Package** En "Name", escribe:

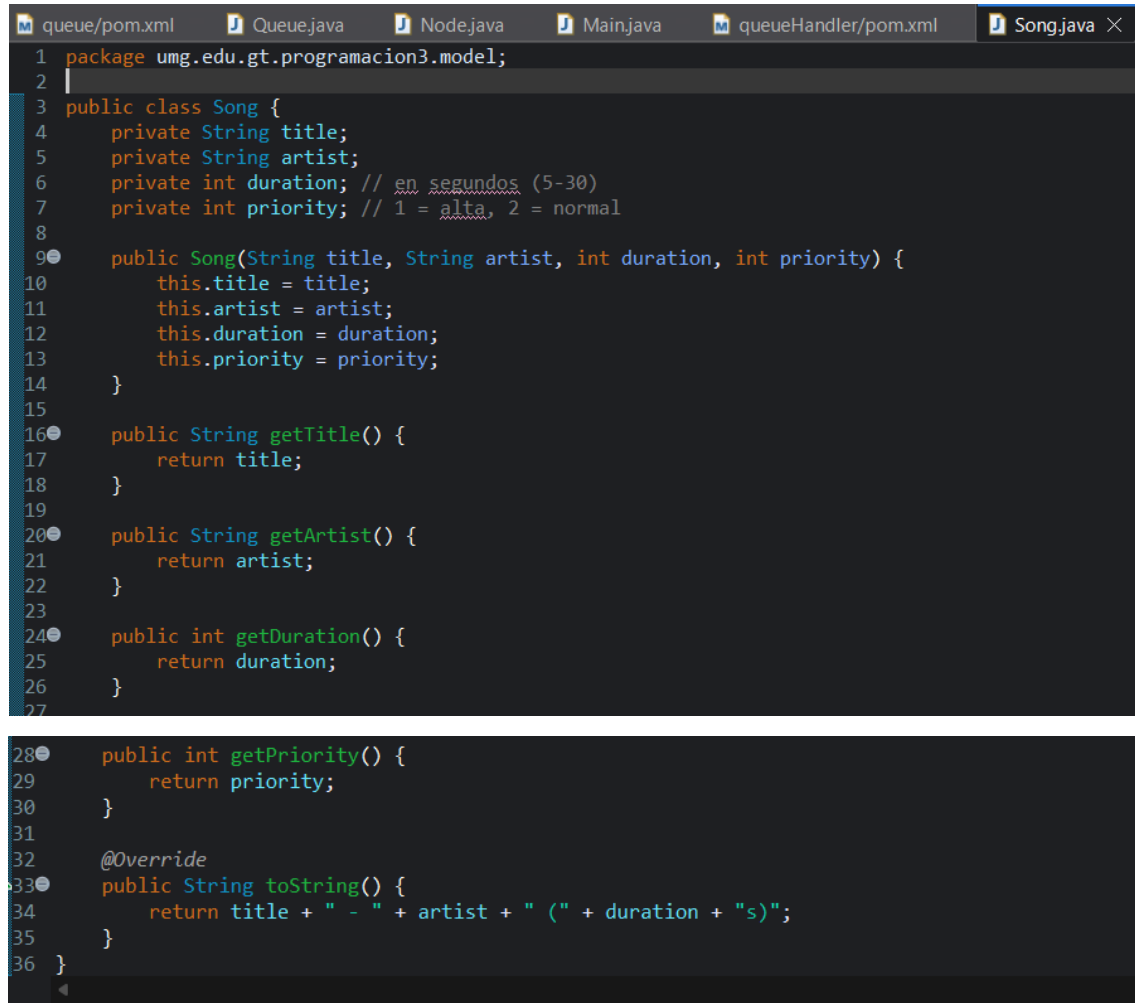
umg.edu.gt.programacion3.model

Crear la clase **Song.java** dentro de **umg.edu.gt.programacion3.model**



PASO 9

Colocar este código en la clase **Song.java**



```

1 package umg.edu.gt.programacion3.model;
2
3 public class Song {
4     private String title;
5     private String artist;
6     private int duration; // en segundos (5-30)
7     private int priority; // 1 = alta, 2 = normal
8
9     public Song(String title, String artist, int duration, int priority) {
10         this.title = title;
11         this.artist = artist;
12         this.duration = duration;
13         this.priority = priority;
14     }
15
16     public String getTitle() {
17         return title;
18     }
19
20     public String getArtist() {
21         return artist;
22     }
23
24     public int getDuration() {
25         return duration;
26     }
27
28     public int getPriority() {
29         return priority;
30     }
31
32     @Override
33     public String toString() {
34         return title + " - " + artist + " (" + duration + "s)";
35     }
36 }

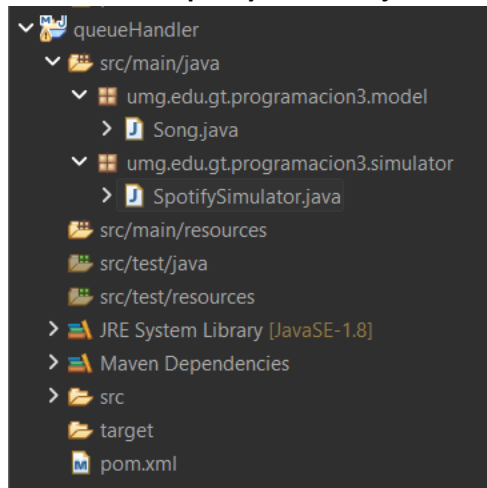
```

PASO 10

En el Package Explorer, haz clic derecho sobre **src/main/java** (dentro del proyecto queueHandler) Selecciona New → Package En "Name", escribe:

umg.edu.gt.programacion3.simulator

Crear la clase **SpotifySimulator.java** dentro de **umg.edu.gt.programacion3.simulator**



PASO 11

Colocar este código en la clase **SpotifySimulator.java**

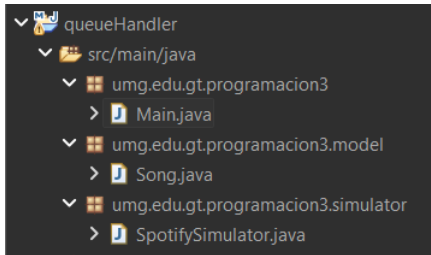
```

1 package umg.edu.gt.programacion3.simulator;
2
3 import umg.edu.gt.data_structure.queue.manual.Queue;
4 import umg.edu.gt.programacion3.model.Song;
5
6 public class SpotifySimulator {
7
8     private Queue<Song> highPriorityQueue; // Prioridad 1
9     private Queue<Song> normalPriorityQueue; // Prioridad 2
10
11     public SpotifySimulator() {
12         highPriorityQueue = new Queue<>();
13         normalPriorityQueue = new Queue<>();
14     }
15
16     // Agregar canción según su prioridad
17     public void addSong(Song song) {
18         if (song.getPriority() == 1) {
19             highPriorityQueue.enqueue(song);
20         } else {
21             normalPriorityQueue.enqueue(song);
22         }
23     }
24
25     // Obtener siguiente canción (prioridad primero)
26     private Song getNextSong() {
27         if (!highPriorityQueue.isEmpty()) {
28             return highPriorityQueue.dequeue();
29         } else if (!normalPriorityQueue.isEmpty()) {
30             return normalPriorityQueue.dequeue();
31         }
32         return null;
33     }
34
35     // Verificar si hay canciones
36     public boolean hasSongs() {
37         return !highPriorityQueue.isEmpty() || !normalPriorityQueue.isEmpty();
38     }
39
40     // Reproducir canción con simulación segundo a segundo
41     public void playSong(Song song) {
42         System.out.println("[LOG] Now playing: " + song.getTitle() + " - " +
43             song.getArtist() + " (" + song.getDuration() + "s)");
44
45         // Simular reproducción segundo a segundo
46         for (int i = 1; i <= song.getDuration(); i++) {
47             System.out.println("[LOG] Playing: " + song.getTitle() + " | " + i + "s / " +
48                 song.getDuration() + "s");
49             try {
50                 Thread.sleep(1000); // 1 segundo real = 1 segundo de simulación
51             } catch (InterruptedException e) {
52                 e.printStackTrace();
53             }
54         }
55         System.out.println("[LOG] Finished: " + song.getTitle());
56     }
57
58     // Iniciar playlist
59     public void startPlaylist() {
60         System.out.println("[LOG] Starting playlist...");
61
62         while (hasSongs()) {
63             Song song = getNextSong();
64             if (song != null) {
65                 playSong(song);
66             }
67         }
68
69         System.out.println("[LOG] Playlist finished.");
70     }
71 }
72
73 public int getPriority() {
74     return priority;
75 }
76
77 @Override
78 public String toString() {
79     return title + " - " + artist + " (" + duration + "s)";
80 }
81
82 }

```

PASO 12

En el Package Explorer, haz clic derecho sobre **src/main/java** (dentro del proyecto queueHandler) Selecciona New → Package En "Name", escribe: **umg.edu.gt.programacion3**
 Crear la clase **Main.java** dentro de **umg.edu.gt.programacion3**



PASO 11

Colocar este código en la clase **Main.java**

```

1 package umg.edu.gt.programacion3;
2
3 import umg.edu.gt.programacion3.model.Song;
4 import umg.edu.gt.programacion3.simulator.SpotifySimulator;
5
6 public class Main {
7
8     public static void main(String[] args) {
9
10         // Crear el simulador
11         SpotifySimulator simulator = new SpotifySimulator();
12
13         // Agregar canciones con diferentes prioridades
14         // Prioridad 1 = Alta, Prioridad 2 = Normal
15
16         // Canciones de prioridad alta (1)
17         simulator.addSong(new Song("Bohemian Rhapsody", "Queen", 8, 1));
18         simulator.addSong(new Song("Hotel California", "Eagles", 10, 1));
19
20         // Canciones de prioridad normal (2)
21         simulator.addSong(new Song("Imagine", "John Lennon", 7, 2));
22         simulator.addSong(new Song("Hey Jude", "The Beatles", 12, 2));
23         simulator.addSong(new Song("Like a Rolling Stone", "Bob Dylan", 9, 2));
24
25         // Iniciar la simulación
26         simulator.startPlaylist();
27     }
28 }

```

PASO 12

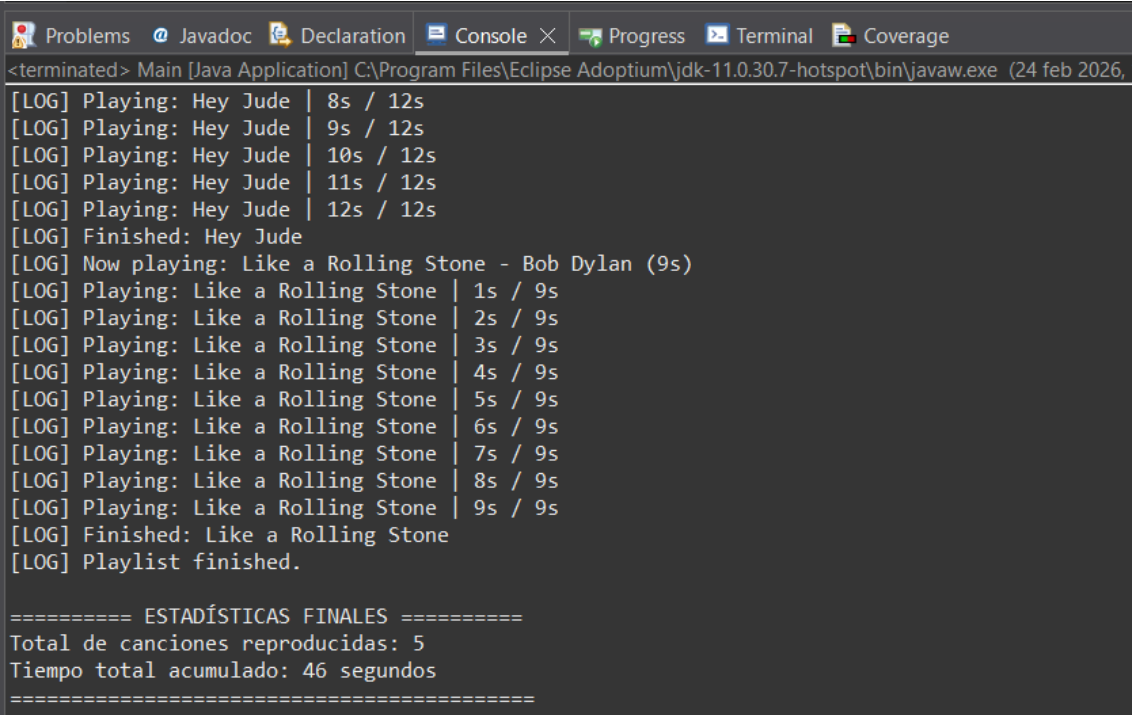
Simulación de Playlist en Consola

```

<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-11.0.30.7-hotspot\bin\javaw.exe (24 f
[LOG] Playing: Hey Jude | 1s / 12s
[LOG] Playing: Hey Jude | 2s / 12s
[LOG] Playing: Hey Jude | 3s / 12s
[LOG] Playing: Hey Jude | 4s / 12s
[LOG] Playing: Hey Jude | 5s / 12s
[LOG] Playing: Hey Jude | 6s / 12s
[LOG] Playing: Hey Jude | 7s / 12s
[LOG] Playing: Hey Jude | 8s / 12s
[LOG] Playing: Hey Jude | 9s / 12s
[LOG] Playing: Hey Jude | 10s / 12s
[LOG] Playing: Hey Jude | 11s / 12s
[LOG] Playing: Hey Jude | 12s / 12s
[LOG] Finished: Hey Jude
[LOG] Now playing: Like a Rolling Stone - Bob Dylan (9s)
[LOG] Playing: Like a Rolling Stone | 1s / 9s
[LOG] Playing: Like a Rolling Stone | 2s / 9s
[LOG] Playing: Like a Rolling Stone | 3s / 9s
[LOG] Playing: Like a Rolling Stone | 4s / 9s
[LOG] Playing: Like a Rolling Stone | 5s / 9s
[LOG] Playing: Like a Rolling Stone | 6s / 9s
[LOG] Playing: Like a Rolling Stone | 7s / 9s
[LOG] Playing: Like a Rolling Stone | 8s / 9s
[LOG] Playing: Like a Rolling Stone | 9s / 9s
[LOG] Finished: Like a Rolling Stone
[LOG] Playlist finished.

```

evidencia_extensiones_parteD



```
<terminated> Main [Java Application] C:\Program Files\Eclipse Adoptium\jdk-11.0.30.7-hotspot\bin\javaw.exe (24 feb 2026, 1
[LOG] Playing: Hey Jude | 8s / 12s
[LOG] Playing: Hey Jude | 9s / 12s
[LOG] Playing: Hey Jude | 10s / 12s
[LOG] Playing: Hey Jude | 11s / 12s
[LOG] Playing: Hey Jude | 12s / 12s
[LOG] Finished: Hey Jude
[LOG] Now playing: Like a Rolling Stone - Bob Dylan (9s)
[LOG] Playing: Like a Rolling Stone | 1s / 9s
[LOG] Playing: Like a Rolling Stone | 2s / 9s
[LOG] Playing: Like a Rolling Stone | 3s / 9s
[LOG] Playing: Like a Rolling Stone | 4s / 9s
[LOG] Playing: Like a Rolling Stone | 5s / 9s
[LOG] Playing: Like a Rolling Stone | 6s / 9s
[LOG] Playing: Like a Rolling Stone | 7s / 9s
[LOG] Playing: Like a Rolling Stone | 8s / 9s
[LOG] Playing: Like a Rolling Stone | 9s / 9s
[LOG] Finished: Like a Rolling Stone
[LOG] Playlist finished.

===== ESTADÍSTICAS FINALES =====
Total de canciones reproducidas: 5
Tiempo total acumulado: 46 segundos
=====
```