

# **Memoria del Proyecto**

## **Gestión y Consumo de Jamones**

### **RA 2**



Aitor Jury Rodríguez. 2º DAM.  
Programación de Servicios y Procesos - RA2.

## Índice

<b>Índice</b>	<b>2</b>
<b>1. Introducción</b>	<b>3</b>
<b>2. Objetivo del Proyecto</b>	<b>4</b>
<b>3. Herramientas</b>	<b>5</b>
<b>4. Especificación de clases</b>	<b>6</b>
4.1. Clase Jamon	6
4.2. Clase Almacen	6
4.3. Clase Productor (implementa Runnable)	6
4.4. Clase Consumidor (implementa Runnable)	6
4.5. Clase Programa	6
4.6. Clase Main	6
<b>5. Diseño</b>	<b>7</b>
5.1. Organización física del proyecto	7
5.2. UML	7
5.3. Casos de uso	8
<b>6. Funcionamiento</b>	<b>9</b>
<b>7. Mejoras implementadas</b>	<b>10</b>
<b>8. Conclusión</b>	<b>11</b>

## **1. Introducción**

En este proyecto se ha desarrollado una simulación de un sistema de producción y consumo de jamones utilizando Java. La finalidad es representar de manera sencilla cómo se produce, almacena y consume un producto dentro de una cadena de suministro, aplicando conceptos de concurrencia y sincronización de hilos.

El proyecto se inspira en el clásico problema del productor-consumidor, donde varios productores generan jamones y varios consumidores los retiran de un almacén con capacidad limitada. La simulación permite observar cómo se gestionan las condiciones de espera cuando el almacén se llena o se vacía, garantizando que los elementos se procesen correctamente sin pérdida de datos ni conflictos por accesos concurrentes.

## **2. Objetivo del Proyecto**

El objetivo principal de este proyecto es simular un sistema de producción y consumo de jamones mediante Java, utilizando hilos y sincronización para controlar la concurrencia. Se buscan alcanzar los siguientes objetivos:

- Representar productores (granjas) que generan jamones con identificador único.
- Representar consumidores (tiendas) que retiran y procesan los jamones.
- Implementar un almacén compartido (buffer) con capacidad limitada, donde los jamones se depositan temporalmente.
- Garantizar que la producción y consumo sean seguros, sin pérdida de datos ni consumo de jamones inexistentes.
- Mostrar en consola la evolución del sistema y registrar todas las operaciones en un archivo de texto.

### **3. Herramientas**

Para el desarrollo del proyecto se han utilizado las siguientes herramientas:

- Lenguaje de programación: Java SE 11.
- IDE recomendado: IntelliJ IDEA / Eclipse.
- Librerías: solo se han utilizado las de la API básica de Java (java.util.\* , java.io.\*).
- Hilos: se han implementado mediante Thread y Runnable.
- Estructuras de datos: LinkedList para implementar el buffer compartido.
- Control de concurrencia: synchronized, wait(), notifyAll().
- Archivo de registro: BufferedWriter para guardar la secuencia de producción y consumo de jamones.

## **4. Especificación de clases**

El programa se organiza en varias clases principales:

### **4.1. Clase Jamon**

- Representa un jamón con un ID único.
- Contiene constructor, método getId() y toString().
- Función: simular un jamón producido y consumido.

### **4.2. Clase Almacen**

- Implementa el buffer compartido entre productores y consumidores.
- Usa LinkedList<Jamon> como cola FIFO.
- Controla la capacidad máxima (max).
- Métodos sincronizados:
  - produceNextJamon(int idP): produce un jamón en orden numérico y lo añade al buffer.
  - consume(int idC): retira un jamón del buffer.
  - write(String message): imprime mensajes en consola y los guarda en archivo.
  - closeWrite(): cierra el archivo.
- Sincronización mediante synchronized, wait() y notifyAll().

### **4.3. Clase Productor (*implementa Runnable*)**

- Cada productor genera un número determinado de jamones.
- Cada iteración: crea un jamón, lo añade al almacén y espera un tiempo aleatorio simulando producción.
- Al finalizar, imprime un mensaje de finalización.

### **4.4. Clase Consumidor (*implementa Runnable*)**

- Cada consumidor retira un número determinado de jamones.
- Cada iteración: retira un jamón del almacén y espera un tiempo aleatorio simulando procesamiento.
- Al finalizar, imprime un mensaje de finalización.

### **4.5. Clase Programa**

- Controla la creación de productores y consumidores y la inicialización del almacén.
- Solicita parámetros al usuario: tamaño del buffer, número de jamones, productores y consumidores.
- Distribuye los jamones entre productores y consumidores, lanza los hilos y espera a que terminen.
- Calcula el tiempo total de ejecución y cierra el archivo de registro.

### **4.6. Clase Main**

- Contiene el método main() que inicia el programa.
- Llama a los métodos de Programa para lanzar la simulación.

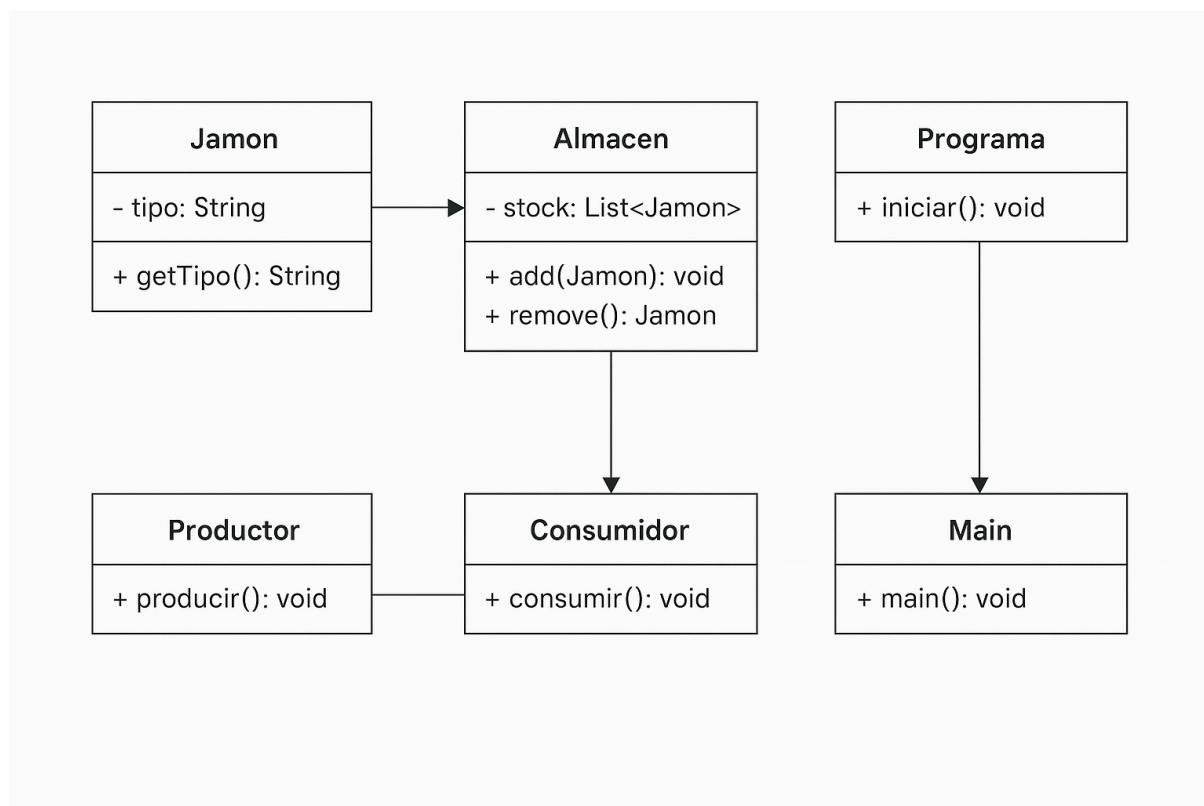
## 5. Diseño

### 5.1. Organización física del proyecto

El proyecto se organiza de la siguiente manera:

```
JamonSync/
|
|---src/
|   |---Jamon.java
|   |---Almacen.java
|   |---Productor.java
|   |---Consumidor.java
|   |---Programa.java
|   |---Main.java
|
|---registro_jamones.txt (Si está creado)
```

### 5.2. UML



Relaciones importantes:

- Productor y Consumidor acceden al Almacen para producir o consumir jamones.
- Programa coordina la creación de los hilos y la inicialización del Almacen.
- Main solo inicia el programa y solicita los parámetros al usuario.

### **5.3. Casos de uso**

1. Iniciar simulación:
  - Actor: Usuario.
  - Descripción: El usuario introduce los parámetros de simulación (tamaño del almacén, número de jamones, productores y consumidores).
  - Resultado: Se crea un Almacen y se lanzan los hilos de productores y consumidores.
2. Producir jamón:
  - Actor: Productor (hilo).
  - Descripción: Cada productor genera un jamón con un ID único y lo añade al almacén.
  - Condición: Si el almacén está lleno, espera.
  - Resultado: Jamón añadido al buffer y registro en archivo.
3. Consumir jamón:
  - Actor: Consumidor (hilo).
  - Descripción: Cada consumidor retira un jamón del almacén y lo procesa.
  - Condición: Si el almacén está vacío, espera.
  - Resultado: Jamón retirado del buffer y registro en archivo.
4. Finalizar simulación
  - Actor: Programa.
  - Descripción: Cuando todos los productores y consumidores han terminado, se imprime el tiempo total de ejecución y se cierra el archivo de registro.

## 6. Funcionamiento

Ejemplo de salida:

```
[Productor 1] Producido Jamon-1  
[Almacen] Se ha almacenado Jamon-1  
[Almacen] Contenido actual: [Jamon-1]  
[Productor 2] Producido Jamon-2  
[Almacen] Se ha almacenado Jamon-2  
[Almacen] Contenido actual: [Jamon-1, Jamon-2]  
[Almacen] Se ha retirado Jamon-1  
[Consumidor 1] Consumido Jamon-1  
[Almacen] Contenido actual: [Jamon-2]  
[Almacen] Se ha retirado Jamon-2  
[Consumidor 2] Consumido Jamon-2  
[Almacen] Contenido actual: []  
...
```

*Tiempo total de ejecucion: 8.817 segundos*

*Programa finalizado: todos los jamones producidos y consumidos...*

1. El usuario indica los parámetros de la simulación (tamaño del almacén, número de jamones, productores y consumidores).
2. Se crea el almacén con capacidad limitada.
3. Se calculan los jamones que le corresponden a cada productor y consumidor.
4. Se lanzan los hilos de productores y consumidores.
5. Cada productor genera un jamón en orden y lo añade al buffer. Si el buffer está lleno, espera.
6. Cada consumidor retira un jamón del buffer. Si el buffer está vacío, espera.
7. Todos los jamones se producen y consumen en **orden numérico**, respetando FIFO.
8. Al finalizar todos los hilos, se imprime el tiempo total de ejecución y se cierra el archivo de registro.

Observaciones:

- Se respeta FIFO.
- Los jamones se producen y consumen en orden numérico.
- Se imprimen mensajes de espera si el buffer se llena o se vacía.

## **7. Mejoras implementadas**

Durante el desarrollo del proyecto, además de cumplir con los requisitos mínimos, se implementaron mejoras para hacer la simulación más completa y controlada. Estas mejoras son:

1. Producción y consumo en orden numérico de jamones:  
Inicialmente, la creación de jamones podía generar IDs en orden correcto, pero su almacenamiento y consumo podían aparecer desordenados debido a la concurrencia de varios productores.  
Se modificó la clase Almacen para que la creación de los jamones y su inserción en la cola se realicen dentro de un método sincronizado (produceNextJamon()), asegurando que los jamones se produzcan y consuman siempre en orden numérico, respetando FIFO.
2. Múltiples productores y consumidores configurables:  
La simulación permite al usuario especificar el número de productores y consumidores.  
El programa reparte automáticamente los jamones entre los productores y consumidores, incluyendo sobrantes, lo que permite simular escenarios más complejos y cercanos a la realidad de una cadena de producción.
3. Registro en archivo de texto (registro\_jamones.txt):  
Todas las acciones de producción y consumo se registran en un archivo, además de mostrarse por consola.  
Esto permite revisar posteriormente la secuencia exacta de jamones producidos y consumidos, útil para pruebas y análisis del sistema.
4. Medición del tiempo total de ejecución:  
Se calcula el tiempo total desde el inicio hasta el fin de la simulación.  
Esto permite al usuario observar cómo el tamaño del buffer, el número de productores y consumidores afectan al rendimiento del sistema.
5. Control de capacidad del almacén:  
El tamaño del buffer es configurable al inicio.  
Si el buffer se llena, los productores esperan, y si se vacía, los consumidores esperan.  
Esto asegura que la simulación funcione correctamente bajo diferentes condiciones de carga, evitando pérdida de jamones o errores de concurrencia.

## **8. Conclusión**

El proyecto cumple los objetivos planteados:

- Simula un sistema de producción y consumo seguro.
- Los jamones se producen y consumen en orden numérico, respetando FIFO.
- La sincronización con wait()/notifyAll() evita errores de concurrencia.
- Permite variar el tamaño del buffer, número de productores y consumidores, haciendo la simulación flexible.
- Registra todas las acciones en un archivo y calcula el tiempo total de ejecución, aportando trazabilidad.

El proyecto demuestra comprensión de los conceptos de hilos, concurrencia, sincronización y gestión de un buffer compartido, aplicados en un entorno simulado realista de producción y consumo de jamones.