**GESTOR DE TAREAS CON PILAS Y COLAS**

**Guatemala, 30 de mayo de 2025**

**AUTOR:**

Irvin José González Mateo 0900-24-24648****

**ASESOR ACADÉMICO:**

Ing. César Juárez cjuarezl8@miumg.edu.gt

**ÍNDICE**

[**Información General 4**](#_w7abz8nzexww)

[**Requisitos del Sistema 4**](#_7ifvfy6mw2ej)

[**Funcionamiento principal del software 5**](#_ghu146744ezf)

[**Estructura del Proyecto 6**](#_at4cox4tqact)

[**Arquitectura del sistema 6**](#_an0x2if18o1f)

[**Diagrama de clases 7**](#_mnw5ailq17mq)

[**Descripción detallada de clases 8**](#_l123mpkeifqi)

[**Flujo de trabajo del sistema 10**](#_l0ocp5ub6z3)

[**Manejo de Errores 11**](#_7jaukxhd6nrn)

[**Configuraciones Especiales 12**](#_opbs8gqvllmk)

[**Consideraciones Técnicas 13**](#_cgbusn0sczz)

[**Glosario o Definiciones Técnicas 14**](#_eh147aui6yed)

[**Contacto y Soporte 14**](#_hc0mtiffcfp7)

**TABLA DE HISTORIAL DE CAMBIOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSIÓN** | **FECHA** | **DESCRIPCIÓN** | **AUTOR** |
| 1.0 | 08/05/2025 | versión inicial del sistema | Irvin González |
| 1.1 | 25/05/2025 | versión con correcciones | Irvin González |
| 1.2 | 30/05/2025 | versión de entrega final | Irvin González |

# Información General

* **Nombre del Proyecto:** Sistema de Gestión de Garantías de Computadoras
* **Lenguaje:** Java (JDK 17+)
* **Persistencia:** Archivos binarios (.dat) y archivos de texto (.txt)
* **Estructuras de Datos:** Cola (Queue) implementada con LinkedList
* **Patrón de Diseño:** MVC simplificado con separación de responsabilidades
* **Propósito:** Gestionar el flujo completo de reparación de computadoras en garantía, desde la recepción hasta la entrega, utilizando un sistema de colas FIFO para cada etapa del proceso.

# 

# Requisitos del Sistema

Software Requerido

* **JDK:** Java Development Kit 17 o superior
* **Editor recomendado:** IntelliJ IDEA, Eclipse, NetBeans o Visual Studio Code
* **Sistema operativo:** Multiplataforma (Windows, Linux, macOS)

Hardware Mínimo

* **Memoria mínima:** 512 MB de RAM
* **Espacio en disco:** 50 MB libres
* **Procesador:** Cualquier procesador compatible con Java

Dependencias Externas

* **Java Standard Library:** Para manejo de I/O, serialización y colecciones
* **java.time:** Para manejo de fechas
* **java.util.regex:** Para validación de patrones
* **java.io:** Para persistencia de datos

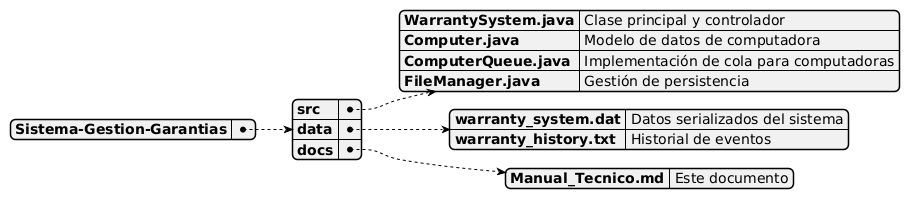
# Funcionamiento principal del software

Este sistema está diseñado para gestionar el flujo completo de reparación de computadoras en un taller de servicio técnico. El software maneja cinco etapas principales:

1. **Recepción:** Registro inicial de la computadora con datos del cliente y descripción del problema
2. **Inspección:** Diagnóstico técnico y evaluación de reparabilidad
3. **Reparación:** Proceso de reparación por parte de técnicos especializados
4. **Control de Calidad:** Verificación de la reparación realizada
5. **Entrega:** Devolución del equipo al cliente

El sistema utiliza colas FIFO (First In, First Out) para cada etapa, asegurando un procesamiento ordenado y trazable de todos los equipos.

# Estructura del Proyecto



# 

# Arquitectura del sistema

El sistema sigue una arquitectura modular con separación clara de responsabilidades:

**Capa de Presentación (UI)**

WarrantySystem.java - Interfaz de consola y navegación

**Capa de Lógica de Negocio**

WarrantySystem.java - Procesos de workflow y validaciones

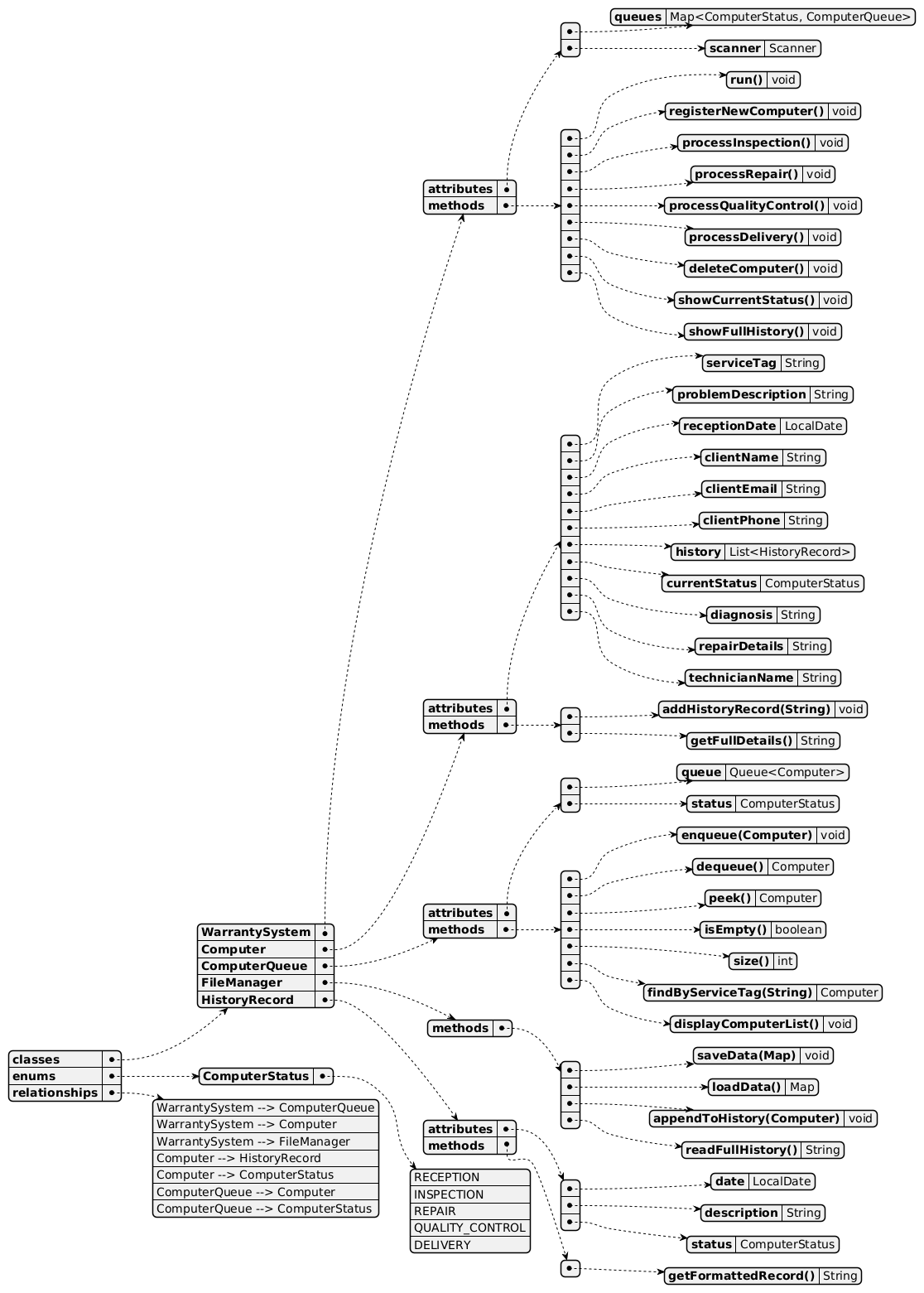
ComputerQueue.java - Gestión de colas y operaciones FIFO

**Capa de Datos**

Computer.java - Modelo de entidad principal

FileManager.java - Persistencia y serialización

# Diagrama de clases



# Descripción detallada de clases

Computer.java

**Responsabilidad:** Modelo de datos principal que representa una computadora en el sistema.

**Atributos principales:**

* serviceTag: Identificador único del equipo (String, convertido a mayúsculas)
* problemDescription: Descripción del problema reportado
* receptionDate: Fecha de recepción del equipo (LocalDate)
* clientName: Nombre del cliente propietario
* clientEmail: Correo electrónico del cliente
* clientPhone: Número telefónico (8 dígitos)
* history: Lista de registros históricos de la computadora
* currentStatus: Estado actual en el flujo de trabajo
* diagnosis: Diagnóstico técnico (opcional)
* repairDetails: Detalles de la reparación realizada (opcional)
* technicianName: Nombre del técnico asignado (opcional)

**Métodos principales:**

* addHistoryRecord(String): Agrega un nuevo registro al historial
* getFullDetails(): Retorna información completa formateada
* toString(): Información básica de la computadora

ComputerQueue.java

**Responsabilidad:** Implementa una cola FIFO específica para cada estado del workflow.

**Características:**

* Utiliza LinkedList<Computer> como estructura subyacente
* Mantiene referencia al estado que representa
* Proporciona métodos de búsqueda y selección
* Maneja la visualización de elementos en cola

**Métodos principales:**

* enqueue(Computer): Agrega computadora al final de la cola
* dequeue(): Remueve y retorna la primera computadora
* findByServiceTag(String): Busca computadora por service tag
* displayComputerList(): Muestra lista numerada de computadoras

FileManager.java

**Responsabilidad:** Gestiona toda la persistencia de datos del sistema.

**Funcionalidades:**

* **Serialización binaria:** Guarda/carga el estado completo del sistema
* **Historial de texto:** Mantiene log legible de todas las operaciones
* **Manejo de errores:** Gestiona archivos corruptos o faltantes
* **Codificación UTF-8:** Asegura compatibilidad con caracteres especiales

**Archivos generados:**

* warranty\_system.dat: Estado serializado del sistema
* warranty\_history.txt: Historial cronológico de eventos

WarrantySystem.java

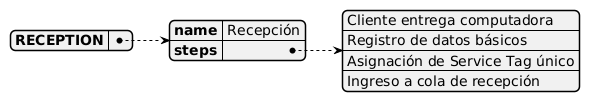
**Responsabilidad:** Controlador principal y interfaz de usuario.

**Funcionalidades principales:**

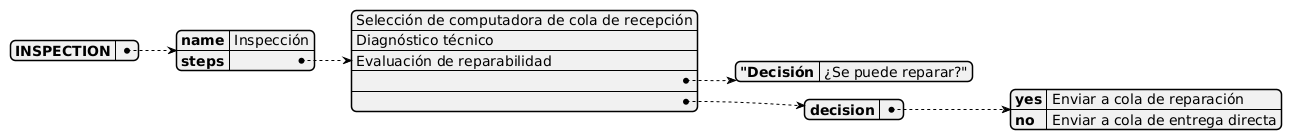
* **Navegación de menús:** Sistema de menús jerárquicos
* **Validación de entrada:** Validación robusta de todos los inputs
* **Gestión de workflow:** Coordina el flujo entre diferentes estados
* **Interfaz de consola:** Proporciona experiencia de usuario completa

# Flujo de trabajo del sistema

1. Recepción (RECEPTION)



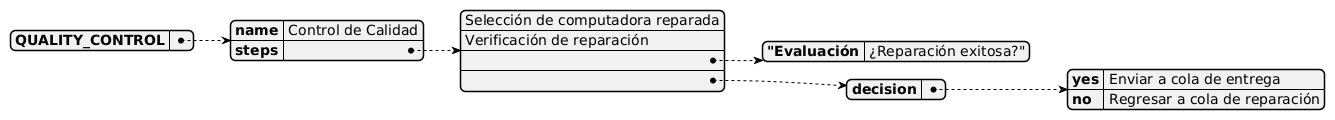
2. Inspección (INSPECTION)



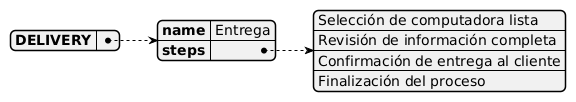
3. Reparación (REPAIR)



4. Control de Calidad (QUALITY\_CONTROL)



5. Entrega (DELIVERY)



# Manejo de Errores

Archivos Faltantes

* **Comportamiento:** El sistema crea automáticamente las estructuras iniciales
* **Implementación:** Método createInitialQueues() en FileManager
* **Recuperación:** Sin pérdida de funcionalidad

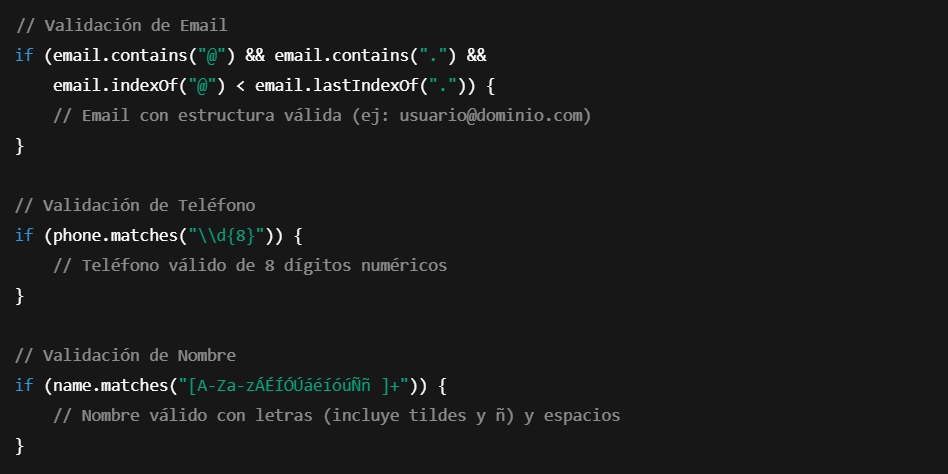
Archivos Corruptos

* **Detección:** Try-catch en métodos de carga
* **Fallback:** Reinicialización con estructuras vacías
* **Logging:** Mensaje de error al usuario

Excepciones Comunes

* **IOException:** Problemas de acceso a archivos
* **ClassNotFoundException:** Incompatibilidad de versiones serializadas
* **NumberFormatException:** Entrada inválida de usuario
* **NullPointerException:** Prevención mediante validaciones

Validaciones de Entrada



# Configuraciones Especiales

Archivos del Sistema

* **Ubicación:** Directorio de ejecución del programa
* **warranty\_system.dat:** Datos principales serializados
* **warranty\_history.txt:** Historial en texto plano

Formatos de Fecha

* **Patrón:** dd/MM/yyyy (ejemplo: 15/05/2025)
* **Clase utilizada:** LocalDate con DateTimeFormatter

Codificación de Caracteres

* **Estándar:** UTF-8 para todos los archivos de texto
* **Configuración:** StandardCharsets.UTF\_8
* **Compatibilidad:** Soporte completo para caracteres especiales y tildes

Comandos Especiales

* **'M':** Retorno al menú principal desde cualquier punto
* **'<':** Comando de retroceso (reservado para futuras versiones)

# Consideraciones Técnicas

Interfaz de Usuario

* **Tipo:** Aplicación de consola completamente textual
* **Navegación:** Sistema de menús numerados
* **Limpieza de pantalla:** Compatible con Windows (cls) y Unix (clear)
* **Internacionalización:** Todas las salidas en español y mayúsculas

Gestión de Memoria

* **Estructuras:** Uso eficiente de LinkedList para colas
* **Serialización:** Objetos completos guardados en memoria durante ejecución
* **Liberación:** Garbage collection automático de Java

Concurrencia

* **Diseño:** Sistema monohilo (single-threaded)
* **Seguridad:** No requiere sincronización
* **Escalabilidad:** Adecuado para uso individual o pequeños equipos

Extensibilidad

* **Estados:** Fácil adición de nuevos estados en enum ComputerStatus
* **Validaciones:** Métodos modulares para nuevas reglas de negocio
* **Persistencia:** Arquitectura preparada para bases de datos

# Glosario o Definiciones Técnicas

Términos de Estructuras de Datos

* **Cola (Queue):** Estructura FIFO donde el primer elemento en entrar es el primero en salir
* **LinkedList:** Implementación de lista enlazada que permite inserciones y eliminaciones eficientes
* **Serialización:** Proceso de convertir objetos Java en secuencia de bytes para almacenamiento

Términos del Dominio

* **Service Tag:** Identificador único alfanumérico asignado a cada computadora
* **Workflow:** Flujo de trabajo que define las etapas del proceso de reparación
* **Historial:** Registro cronológico de todos los eventos relacionados con una computadora
* **Estado (Status):** Etapa actual de una computadora en el proceso de reparación

Términos Técnicos

* **FIFO:** First In, First Out - Política de procesamiento por orden de llegada
* **UTF-8:** Codificación de caracteres estándar que soporta caracteres internacionales
* **Parseo:** Proceso de analizar y extraer información de texto estructurado
* **Thread-safe:** Código seguro para ejecución concurrente

# Contacto y Soporte

Para soporte técnico adicional o reporte de bugs, contactar al equipo de desarrollo del proyecto.

**Versión del Manual:** 1.0  
**Fecha de Actualización:** Mayo 2025  
**Compatibilidad:** Java 17+  
**Desarrollador:** github.com/irvinjosg