**Práctica de Examen Integral Final Práctico**

**Java (NetBeans, Maven o Ant)**

***Caso empresarial:*** PuraVida MiPyME Suite

**Universidad: *Universidad Técnica Nacional***

**Carrera: *Ingeniería en Tecnologías de la información***

**Profesor: *Andrés Joseph Jiménez Leandro***

**Alumno 1: *Emesis Mayerli Mairena Sevilla***

**Alumno 2: *Jairo Steven Herrera Romero.***

**Fecha Completa: 19-08-2025.**

**Modalidad:** En parejas

# Instrucciones Generales

* El examen es totalmente práctico y se realiza en parejas únicamente.
* **Lenguaje y entorno:** Java 16+ en NetBeans 22+; puede usar Maven o Ant de Framework.
* Se deben realizar diagramas de clases UML de POO/OOP (directa en Word o imagen o PDF).
* No se permiten entornos colaborativos en la nube *(la red universitaria los restringe)*.
* Cero dependencias externas *(sin librerías de terceros, ni API’s, ni ORM’s, ni chart libs; aunque se les permite JSON/Gson y no tienen restricción en el caso de las PDF)*. Solo Java estándar (I/O, javax.crypto, java.security, Collections, Comparator, Streams, SwingWorker).
* Persistencia en archivos locales *(CSV, TXT, PDF y JSON simple implementado por el equipo)* y/o Base de Datos *(PostGreSQL 15+/pgAdmin 3+ o SQL Server Express Edition 2019+/SSMS 20+ o Oracle DB 23ai Free/21c XE con Oracle SQL Developer 23+)*.

o En caso del JSON, se debe crear el propio *serializer*/*deserializer*.

* **Entregables:** generar JAR ejecutable, ZIP del proyecto, Diagramas POO y push a GitHub *(URL)*.

# Reglas de examen

* **Tiempo**: Inicia a las 1:00 PM (13:00 horas) y termina sin ampliación a las 5:00 PM (17:00 horas) de forma continua e ininterrumpida.
* **Parejas**: trabajo conjunto, un solo repositorio, uso de ramas y pull requests.
* **IA**: se permite consulta puntual, **pero** no se admite *copiar/pegar* bloques de código, tampoco se permite uso de código auto-generado o generado por IA, sin adaptación personalizada. La defensa y el *git log* deben demostrar autoría (se verificará contra herramientas anti-IA).
* **Sin frameworks extra** ni librerías de terceros, ni API’s, ni OEM’s.
* **Sin entornos colaborativos en la nube** durante la prueba.
* **Datos iniciales**: Crear en Excel dos archivos CSV base (productos.csv, clientes.csv) de 15 filas cada uno (o al caso 2 tablas Productos y Clientes, con 15 inserciones ***– se debe subir en los entregables los SQL o los CSV, depende de la ruta que tomaron***).
* **Semilla de control**: el alumno entregará un número entero SEED para pruebas repetibles del SwingWorker.

# Resumen del Caso Empresarial

Un startup MiPyME costarricense contrata a su equipo para construir **PuraVida MiPyME Suite**, un **escritorio Swing** con arquitectura **POO** que resuelva cuatro necesidades críticas:

1. **Catálogo & Inventario**: alta/edición/baja de productos; importación desde CSV; búsquedas y ordenamientos por múltiples criterios; detección de duplicados.
2. **Clientes (con Datos Sensible)**: registro/edición; **almacenamiento cifrado** del número de identidad (cédula) en disco; visualización en claro sólo bajo rol “Administrador”.
3. **Órdenes & Facturación**: creación de órdenes con ítems, impuestos, descuentos y **estrategias de pago** (efectivo/tarjeta/transferencia). Generación de **comprobante .txt** con **código de control (HMAC-SHA256)**.
4. **Operaciones & Análisis**: dashboard textual (sin librerías gráficas externas) con **Top-5** productos por ingresos, **clientes frecuentes** y **existencias críticas**. Una tarea pesada (recalcular fidelización) debe ejecutarse en **segundo plano** con barra de progreso y cancelación.
   * Construir PuraVida MiPyME Suite: aplicación de escritorio (Swing) con arquitectura POO para un startup.
   * **Módulos mínimos:** Catálogo & Inventario, Clientes (con cédula cifrada), Órdenes & Facturación, Dashboard y tarea pesada en segundo plano (SwingWorker).
   * **Patrones exigidos:** *Strategy* (pagos y descuentos), *Factory Method* (creación de pagos), *Observer* (refresco UI ante cambios).
   * Reglas de negocio con excepciones personalizadas (stock insuficiente, duplicados, pagos inválidos).

# Entregables Obligatorios

1. Repositorio público con acceso al profesor, con estructura limpia (domain/, infrastructure/, application/, ui/, tests/, data/) *–> Estructura de Carpetas/Folders*.
2. Diagrama de clases UML (imagen o PDF) alineado al código final.
3. JAR ejecutable sin dependencias externas (Clean & Build).
4. README.md: cómo compilar/ejecutar, usuarios de prueba y flujo de validación.
5. ZIP del proyecto y registro Git con ≥10 commits significativos (mínimo 5 cada miembro de la pareja) y una etiqueta v1.0-examen.
6. Datos de prueba en data/ *(CSV iniciales y JSON exportados por la app o clase de conexión con base de datos y exportación de datos por la app)*.

# Requisitos funcionales y técnicos

## Módulo Catálogo & Inventario

* ***Entidades:*** Producto {codigo, nombre, categoria, precio, stockMin, stockActual}.
* ***Operaciones:*** alta/edición/baja, búsqueda por texto y filtros por categoría, **ordenamientos combinados** (precio desc, nombre asc, stock asc).
* ***Importación CSV:*** detecta duplicados por codigo y crea un **reporte .txt** con altas/actualizaciones/errores (línea y causa).
* ***Estructuras de datos:*** List<Producto> + Map<String, Producto> (índice por codigo).
* ***Validaciones con excepciones personalizadas:*** PrecioInvalidoException,

StockNegativoException, ProductoDuplicadoException.

**Módulo Clientes *(Datos Sensibles con cifrado)***

* ***Entidad:*** Cliente {idInterno, nombre, correo, telefono, cedulaCifrada}.
  + La **cédula se guarda cifrada** en disco con **AES/GCM/NoPadding** *(clave derivada de una contraseña conocida PuraVida-2025! con PBKDF2WithHmacSHA256).*
  + La UI muestra “\*\*\*” y un botón “Revelar” que **solicita contraseña admin** y, si es correcta, **descifra en memoria** y la muestra *temporalmente (se revierte al cambiar de pestaña)*.

## Órdenes & Facturación

* ***Entidades:*** Orden {id, cliente, items, totalBruto, impuestos, descuentos, totalNeto, fecha} y ItemOrden {producto, cantidad, precioUnitario}.
* ***Estrategias de pago (Strategy):*** PagoEfectivo, PagoTarjeta, PagoTransferencia, todas implementan Pago.
* ***Descuentos (otro de Strategy):*** p.ej., “cliente frecuente” 5% si > ₡200,000 en el mes; “promoción categoría” configurable.
* ***Comprobante .txt con:*** detalle, totales, hash de integridad **HMAC-SHA256** (clave fija CONTROL-CLAVE-2025).
* ***Reglas de negocio con excepciones:*** StockInsuficienteException,

PagoRechazadoException.

## Operaciones, Análisis y Concurrencia

* ***Dashboard textual (en un JTextArea) que imprima:***

o Top-5 productos por **ingresos** (usa Comparator + streams). o Top-5 clientes por **monto total**. o **Existencias críticas** (stockActual < stockMin).

* ***Tarea costosa:*** “Recalcular fidelización” simula 50k iteraciones (o procesa todas las órdenes) en un **SwingWorker** con barra de progreso y botón **Cancelar**.

## Persistencia y formato de datos

* ***Repositorio genérico Repositorio<T> con implementación en archivos:*** 
  + \*.csv legible (delimitador ;, comillas dobles para escapar).
  + \*.json “mínimo” hecho a mano (sin librerías): arreglos de objetos, primitivas y strings con escapado básico.
* ***Transacciones simples:*** escribir a un archivo temporal y renombrar atómico; backup .bak.

## Patrón Observer y modularidad UI

• EventBus simple (propio) o PropertyChangeSupport para notificar cambios a las vistas (tablas refrescan al agregar/editar/eliminar).

## Pruebas

* ***JUnit 5 con 6 pruebas mínimas:*** cifrado/descifrado, HMAC, cálculo de totales (con impuestos y descuentos), regla de stock, ordenamiento Top-5, importación CSV con duplicados.
* Si JUnit no está disponible sin internet, provea una clase MiniTestRunner que ejecute aserciones manuales *(assert activado con -ea)*.

## Git y control de versiones

* Mínimo **10 commits** con mensajes descriptivos; **1 etiqueta** v1.0-examen.
* .gitignore acorde (carpetas target/ o build/, archivos .bak, etc.). **Se evalúa el uso correcto de Git/GitHub** como parte del curso.

## 4) GUI mínima exigida (Swing, sin librerías externas)

* Ventana principal con **pestañas**: *Catálogo*, *Clientes*, *Órdenes*, *Dashboard*.
* ***Controles:*** tablas, formularios, botones, validación en línea y **diálogo de progreso** (modal) durante tareas largas. *(Estos controles y eventos están en los contenidos curriculares de GUI).*

**Sugerencia de Gestión Cronológica:**

1. Diseño rápido + UML
2. Dominio, repos y CSV/JSON
3. GUI base + Catálogo/Clientes
4. Órdenes, estrategias, comprobante HMAC
5. Cifrado cédula + rol Admin
6. Dashboard + SwingWorker
7. Pruebas / MiniTestRunner + pulido
8. Git, etiqueta, Clean & Build, ZIP, push

# Rúbrica de Evaluación (100pts)

|  |  |  |
| --- | --- | --- |
| **Criterio Detalle** | | **Puntos** |
| **POO + UML** | Modelo de dominio coherente; diagrama de clases consistente; encapsulamiento, herencia y polimorfismo donde aporta valor. | *10* |
| **Persistencia local** | Repositorios genéricos; CSV/JSON propio; escritura atómica y backups. | *12* |
| **Cifrado & Seguridad** | AES/GCM para cédula; derivación PBKDF2; revelado por rol Admin; manejo seguro en memoria. | *8* |
| **Órdenes & Facturación** | Totales con impuestos/desc.; Strategy de pagos y descuentos; comprobante con HMAC-SHA256. | *15* |
| **Estructuras y Algoritmos** | Índices Map; Comparator/streams; Top‑5; rendimiento aceptable. | *10* |
| **Excepciones** | Excepciones personalizadas para reglas; manejo y mensajes claros. | *8* |
| **Concurrencia (SwingWorker)** | Progreso y cancelación; UI no bloqueada. | *10* |
| **GUI/UX** | Pestañas, formularios, validación, refresco por Observer. | *10* |
| **Pruebas** | 6 pruebas (o MiniTestRunner) para cálculos críticos y utilidades criptográficas. | *5* |
| **Git/GitHub** | ≥10 commits claros, etiqueta v1.0-examen, push o ZIP + git log. | *5* |
| **Entrega y defensa** | Estructura del repo, README mínimo, y respuestas técnicas. | *7* |
| **Total** |  | ***100*** |

# Checklist de Corrección Rápida

[ ] Diagrama de clases UML actualizado y coherente con el código.

[ ] CRUD de productos y clientes funcionando; importación CSV con reporte de duplicados/errores.

[ ] Cédula cifrada en disco (AES/GCM); revelado condicionado por rol Admin.

[ ] Órdenes con impuestos y descuentos correctos; Strategy de pagos; comprobante .txt con HMAC-SHA256.

[ ] Dashboard con Top‑5 productos/Clientes y existencias críticas.

[ ] Tarea pesada en SwingWorker con barra de progreso y opción de cancelar (UI no bloqueada).

[ ] Excepciones personalizadas para validaciones/reglas y mensajes claros al usuario.

[ ] 6 pruebas unitarias (o MiniTestRunner) para cálculos e integridad criptográfica.

[ ] git log con ≥10 commits significativos y etiqueta v1.0-examen.

[ ] JAR ejecutable, ZIP final del proyecto, README mínimo.

***¡BUENA SUERTE!***