Parcial 2 – Programación Orientada a Objetos

Facultad de Ingeniería – Universidad Nacional de Colombia
Nombre del estudiante: _____
Curso: _____
Fecha de entrega: _____

📋 Planteamiento del problema

Simulador de Gestión de Consultas Médicas

Diseñe e implemente un sistema para gestionar las consultas médicas en una clínica. El sistema debe permitir registrar pacientes, asignar médicos a consultas, almacenar el historial de consultas, y permitir búsquedas por paciente o médico.

☑ Requerimientos funcionales

El sistema debe permitir:

1. **Login de usuario médico o administrativo.**
2. **Registrar nuevos pacientes y médicos.**
3. **Asignar una consulta a un paciente con un médico y registrar síntomas, diagnóstico y tratamiento.**
4. **Consultar el historial médico de un paciente.**
5. **Listar todas las consultas realizadas por un médico.**

🔧 Requisitos técnicos (se evaluarán como parte de la rúbrica)

- Implementación del patrón **MVVM**.
- Uso de al menos **tres relaciones entre clases** (herencia, composición, asociación).
- Implementación de **una clase abstracta** e **interfaces**.
- Uso de **colecciones genéricas** para almacenar pacientes, médicos y consultas.
- Persistencia de datos en **archivos planos** o base de datos (usando Realtime o serialización).

- Manejo de **excepciones personalizadas** (por ejemplo, usuario no encontrado, campos vacíos, etc.).
- **Interfaz gráfica en JSwing**, con menús funcionales.
- Documentación del desarrollo en PDF con capturas y explicación paso a paso.
- Subir código a un repositorio en **GitHub**.

💡 Sugerencias para la implementación

- Puedes tener una clase abstracta `Persona` con métodos comunes a `Paciente` y `Medico`.
- Usa una interfaz `Agendable` para clases que gestionan horarios de consulta.
- Implementa una clase `Clinica` que maneje colecciones de pacientes, médicos y consultas.
- La vista puede ser una ventana principal con botones que abren diálogos para cada funcionalidad.

📌 Entregable

- PDF con explicación paso a paso del desarrollo.
- Capturas de pantalla que evidencien el funcionamiento de cada caso de uso.
- Lista de requerimientos con casillas de verificación.
- Link a repositorio GitHub con el código.

Estructura del proyecto sugerida

Clinica/

│

├── model/

```
│   ├── Persona.java            // Clase abstracta
│   ├── Paciente.java
│   ├── Medico.java
│   ├── Consulta.java
│   ├── Clinica.java            // Contiene colecciones
│   └── IPersistencia.java      // Interfaz
│
├── view/
│   ├── VentanaPrincipal.java   // JFrame con menú principal
│   ├── PanelRegistro.java      // Panel para registrar personas
│   ├── PanelConsulta.java      // Panel para asignar y registrar consultas
│   └── PanelHistorial.java     // Panel para consultar datos
│
├── viewmodel/
│   ├── ClinicaViewModel.java   // Conecta modelo y vista
│
├── persistencia/
│   └── PersistenciaArchivo.java  // Implementación de IPersistencia
│
├── excepciones/
│   ├── UsuarioNoEncontradoException.java
│   └── CampoVacioException.java
│
└── Main.java
```

Rúbrica

- Código (2,5)

Clases Abstractas

Organización

Persistencia DB, asociaciones

Casos de uso

Repositorio

*(Puntos adicionales para GUI Agradables e innovadoras, 0,5)

- Preguntas (2,5)

1

2

3