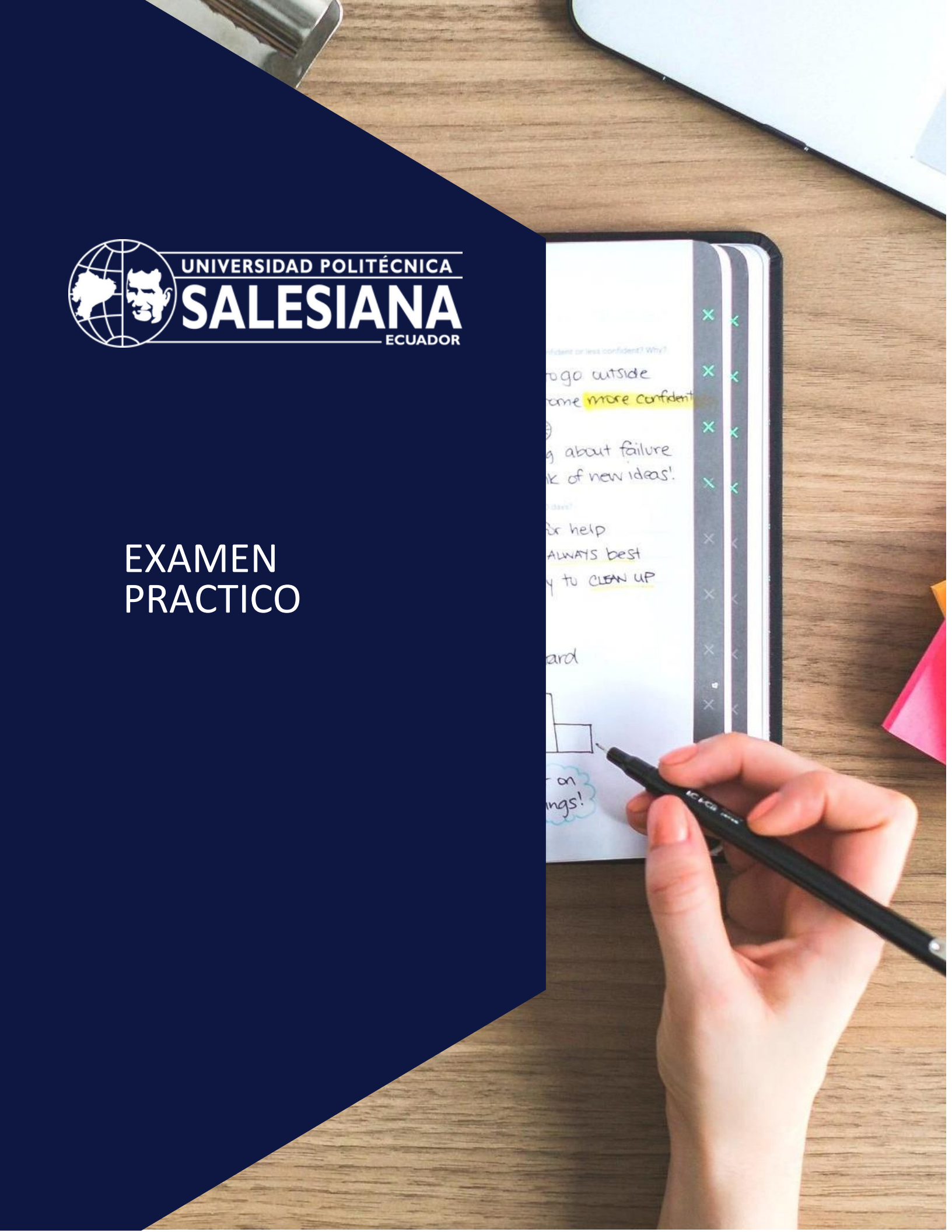
UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

EXAMEN PRACTICO



Informe Final del Proyecto: Juego de Buscaminas en Consola

Enlace al repositorio de GitHub del proyecto: https://github.com/lios31/EXAMEN-POO.git

Introducción

Este proyecto consiste en una implementación del juego clásico Buscaminas en Java, desarrollado con principios de programación orientada a objetos (POO) y pruebas unitarias con JUnit para garantizar la calidad del código. El diseño del sistema asegura modularidad, extensibilidad y un desarrollo estructurado.

Se han desarrollado las funcionalidades esenciales del juego, como inicialización del tablero, marcación y descubrimiento de casillas, manejo de minas, y validación de la lógica de victoria/derrota.

Resumen de las Etapas Implementadas

Etapa 1: Definición de Clases y Atributos

Se definieron las clases principales para representar los componentes esenciales del juego:

- **Clase Tablero:** Representa el tablero del juego.
  - **Atributos:**
    - filas: cantidad de filas en el tablero.
    - columnas: cantidad de columnas en el tablero.
    - numeroDeMinas: número total de minas en el tablero.
    - casillas: matriz que almacena el estado de cada casilla.
  - **Métodos principales:**
    - inicializarTablero(filas, columnas, numeroDeMinas): Inicializa el tablero con las dimensiones y minas especificadas.
    - descubrirCasilla(fila, columna): Descubre una casilla.
    - marcarCasilla(fila, columna): Marca una casilla como sospechosa.
- **Clase Casilla:** Representa cada casilla en el tablero.
  - **Atributos:**
    - tieneMina, estaDescubierta, estaMarcada, minasAdyacentes.
- **Excepciones:**
  - **CoordenadaInvalidaException:** Maneja errores al acceder a coordenadas fuera del rango.

Etapa 2: Implementación del Patrón MVC

El diseño se estructura en tres capas:

- **Modelo**: Clases `Tablero` y `Casilla`.
- **Vista**: Clase `Vista` para la interacción con el usuario.
- **Controlador**: Clase `ControladorJuego` que contiene la lógica del juego.

Esta estructura permite una separación clara entre la lógica del negocio, la interfaz de usuario y el control del flujo del juego.

Etapa 3: Manejo de Excepciones y Persistencia de Datos

Manejo de Excepciones:

- Se implementaron excepciones personalizadas como CasillaYaDescubiertaException y CoordenadaInvalidaException.

Persistencia de Datos:

- Utilización de archivos de texto para guardar y cargar el estado del juego.
- Serialización de objetos para permitir la pausa y reanudación del juego.

Etapa 4: Aplicación de Código Limpio y TDD

Principios de Código Limpio:

- **DRY (Don't Repeat Yourself)**: Reducción de duplicación mediante métodos reutilizables.
- **KISS (Keep It Simple, Stupid)**: Mantenimiento de soluciones simples y claras.

TDD (Desarrollo Guiado por Pruebas):

- Se escribieron pruebas unitarias antes de desarrollar la funcionalidad principal.
- Uso de JUnit para validar métodos clave como descubrirCasilla() y verificarVictoria()

Etapa 5: Validación de Reglas del Juego

Se implementaron las reglas principales del Buscaminas:

- Descubrir una casilla con una mina termina el juego.
- Se gana el juego al descubrir todas las casillas que no contienen minas.
- El número de minas adyacentes se calcula y se muestra al descubrir una casilla.

Etapa 6: Desarrollo de Pruebas Unitarias

Se implementaron pruebas unitarias utilizando **JUnit** para asegurar la calidad del código.

Clase `TableroTest`: Pruebas para la lógica del tablero.

- **Pruebas implementadas:**
  - Inicialización correcta del tablero (filas, columnas, minas).

- Comportamiento al descubrir casillas con/sin mina.
- Marcación y desmarcación de casillas.
- Validación de excepciones para coordenadas inválidas.

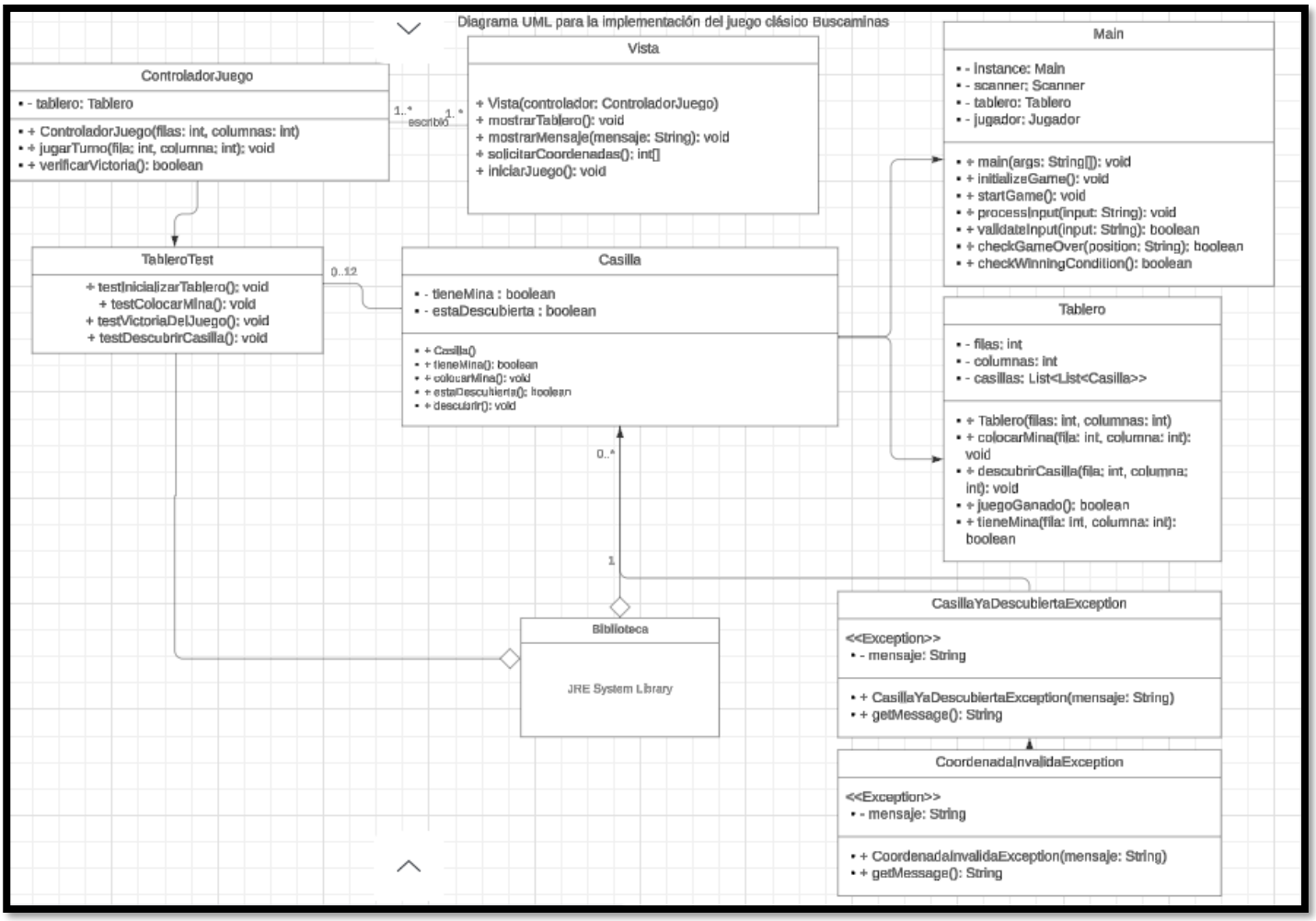
- **Estructura del Proyecto:** Buscaminas en Consola (Eclipse)

```
Buscaminas/
├── src/
│   ├── modelo/                 # Paquete para las clases del modelo (lógica del juego)
│   │   ├── Casilla.java
│   │
│   │   ├── Tablero.java
│   ├── vista/                  # Paquete para la interfaz de usuario
│   │   └── Vista.java
│   ├── controlador/            # Paquete para la lógica del juego (controlador)
│   │   └── Juego.java
│   ├── excepciones/            # Paquete para excepciones personalizadas
│   │   ├── CasillaYaDescubiertaException.java
│   │   └── CoordenadaInvalidaException.java
├── test/                       # Paquete para pruebas unitarias (JUnit)
│   └── pruebas/
│   ├── MainTest.java
├── main.java
├── README.md                   # Documentación del proyecto
└── .gitignore                  # Archivos a ignorar por Git
```

6. Diagrama de Clases UML Actualizado

El diagrama de clases UML refleja la estructura final del sistema, destacando las relaciones entre las clases, atributos y métodos.

- El diagrama incluye las clases principales, el Controlador y la Vista.



Conclusión

El desarrollo del **Juego de Buscaminas en Consola** permitió aplicar de manera práctica los conceptos fundamentales de POO, así como fortalecer habilidades en diseño, desarrollo y colaboración. La implementación del patrón MVC, el manejo de excepciones personalizadas, la persistencia de datos y el uso de GitHub consolidaron una experiencia integral de desarrollo de software.