PROGRAMACIÓN ORIENTADA A OBJETOS

Taller 03

II Semestre – 2025

ITI - ICCI

Docentes: **Alejandro Paolini Godoy**, **Cristhian Rabi Reyes**

Ayudantes: **Daniel Durán García**, **Nicolás Rojas Bustos**

Treyo

Nos situamos en el año 2035. La empresa **TaskForge Ltda.** ha solicitado el desarrollo de una herramienta de gestión de proyectos para su equipo de desarrollo de software.

La herramienta debe permitir organizar proyectos y sus tareas, clasificarlas según su tipo, establecer prioridades, registrar responsables y generar reportes.

El sistema contará con dos roles principales:

- **Administrador**: Puede crear proyectos, gestionar tareas, asignar responsables y generar reportes completos.
- **Colaborador**: Puede consultar proyectos, ver tareas asignadas y actualizar su estado.

Objetivos:

- Diseñar un **modelo de dominio y diagrama UML** consistente con la implementación en Java.
- Implementar **herencia, polimorfismo e interfaces** en un sistema orientado a objetos
- Aplicar **patrones de diseño** (Singleton, Factory, Strategy, Visitor).
- Implementar **persistencia en archivos de texto** (lectura/escritura de proyectos, tareas y usuarios).

Archivos:

usuarios.txt: Contiene la información de los usuarios del sistema.

```
admin1|admin123|Administrador
colab1|1234|Colaborador
colab2|abcd|Colaborador
```

El txt se divide de la siguiente manera:

- Username: Indica el nombre de usuario.
- Contraseña: Indica la contraseña del usuario.
- Rol: Indica el rol que cumple el usuario.

proyectos.txt: Contiene la información de los proyectos.

```
PR001|Plataforma de E-commerce|admin1
PR002|Sistema de Inventario|admin1
PR003|Aplicación Móvil de Reservas|admin1
```

El txt se divide de la siguiente manera:

- ID: Identificador único del proyecto.
- Nombre: Indica el nombre del proyecto.
- Responsable: Indica el responsable del proyecto.

tareas.txt: Contiene la información de las tareas.

```
PR001|T001|Bug|Error en el login de usuarios|Pendiente|colab1|Alta|2025-08-01
PR001|T002|Feature|Agregar carrito de compras|Pendiente|colab2|Media|2025-08-01
PR001|T003|Documentacion|Redactar manual de usuario|En progreso|colab1|Baja|2025-08-02
PR002|T004|Feature|Implementar módulo de reportes|Pendiente|colab2|Alta|2025-08-02
PR002|T005|Bug|Fallo en exportación de datos|Pendiente|colab1|Alta|2025-08-03
PR003|T006|Feature|Diseñar pantalla de reservas|Pendiente|colab1|Media|2025-08-04
PR003|T007|Documentacion|Escribir casos de uso|Pendiente|colab2|Baja|2025-08-03
```

El txt se divide de la siguiente manera:

- Proyecto: Señala el id del proyecto al que pertenece.
- ID: Identificador único de la tarea.
- Tipo: Indica el tipo de tarea que es.
- Descripción: Indica la descripción de la tarea.
- Estado: Indica el estado actual en el que se encuentra la tarea.
- Responsable: Indica la persona responsable de la tarea.
- Complejidad: Indica la complejidad de la tarea.
- Fecha: Indica la fecha de creación de la tarea.

Requerimientos:

Deberán crear dos menús accesibles mediante un logueo capaz de hacer lo siguiente:

- **Menú Administrador (82 pts):**
  - **Ver lista completa de proyectos y tareas:** Mostrar todos los proyectos y sus tareas asociadas. (10 pts)
  - **Agregar o eliminar un proyecto**: Al crear, se debe ingresar nombre y responsable. Al eliminar, también se eliminan sus tareas asociadas. (14 pts)
  - **Agregar o eliminar una tarea en un proyecto**: Preguntar tipo de tarea (Bug, Feature, Documentación), descripción, estado inicial y responsable. (18 pts)
  - **Asignar prioridades con Strategy**: Permitir elegir estrategia de priorización (por fecha de creación, por tipo, por complejidad). (25 pts)
  - **Generar reporte de proyectos:** Crear un archivo reporte.txt con información detallada de proyectos, sus tareas y estado. (15 pts)

- **Menú Usuario (59 pts):**
  - **Ver proyectos disponibles:** Mostrar lista de proyectos con información básica. (6 pts)
  - **Ver tareas asignadas:** Filtrar tareas según el usuario logueado. (10 pts)
  - **Actualizar estado de una tarea**: Cambiar de "Pendiente" a "En progreso" o "Completada". (12 pts)
  - **Aplicar Visitor sobre tareas**: Diferentes acciones según tipo de tarea (20 pts)

Aclaraciones:

- A la hora de asignar una persona a una tarea, se deberá verificar que esta no esté asignada a ninguna otra tarea en la misma fecha.
- Para asignar prioridades, revisar lo siguiente:
    1. **Por fecha de creación** → Las tareas se ordenan según la fecha en que fueron creadas (más antiguas primero).
    2. **Por impacto** → Las tareas se ordenan según la criticidad del tipo de tarea:
        - **Bug** → Alta prioridad.
        - **Feature** → Media prioridad.
        - **Documentación** → Baja prioridad.
    3. **Por complejidad** → Las tareas se ordenan según un valor asignado de complejidad (ejemplo: Baja, Media, Alta)
- Para aplicar acciones según el tipo de tarea, revisar lo siguiente.
    1. Bug → Afecta criticidad del proyecto.
    2. Feature → Impacta en la estimación de tiempo.
    3. Documentación → Mejora la calidad del proyecto.

Consideraciones:

- Cualquier situación especial relacionada con el desarrollo o la entrega del taller debe ser comunicada con anticipación. Las dudas serán respondidas hasta las 17:00 del último día hábil antes de la fecha máxima de entrega (pueden existir excepciones según la situación).
- Cualquier consulta o duda sobre el taller debe realizarse mediante correo institucional o el grupo oficial de WhatsApp.
- Los talleres serán en pareja (2 personas por grupo).
- Este taller **DEBE** usar conceptos de Programación Orientada a Objetos, Arquitectura y Patrones de Diseño. Se pueden utilizar librerías externas previa autorización
- Pongan sus nombres completos, RUT y carrera en las primeras 5 líneas de código. Si no lo hacen, serán evaluados con nota **1.0** (**uno coma cero**).
- El taller deberá ser enviado en un archivo **.txt** que contenga el **link del repositorio GitHub** donde se encuentra alojado el proyecto (se revisarán los aportes de los colaboradores).
- El directorio debe tener el nombre de los integrantes. De lo contrario, serán evaluados con nota **1.0** (**uno coma cero**).
- Formato: **nombreApellidoIntegrante1_nombreApellidoIntegrante2_Taller3POO.txt**
- El repositorio debe tener una estructura clara: **el proyecto Eclipse debe estar en la raíz del repositorio (no dentro de carpetas adicionales)**.
- **No se aceptarán entregas atrasadas y se revisará hasta el último commit realizado antes de la hora límite de entrega.**
- El taller debe ser realizado como **proyecto de Eclipse**.
- Si al ejecutar la clase principal el programa se detiene inmediatamente por un error no manejado, se considerará que **no compila**.
- **Solo una persona del grupo debe subir el archivo de entrega.**
- El código debe estar documentado con **JavaDoc**, y la documentación general del proyecto debe incluirse en el **README** del repositorio, en el readme también deben agregar los datos de los participantes junto a su usuario de gitbub.
- Los **diagramas** (de clases, dominio, etc.) deben entregarse en un **archivo PDF** generado mediante una herramienta digital (no se aceptan diagramas a mano).
- Ambos integrantes deben tener **commits visibles y verificables** en el repositorio; si uno no realiza aportes, podrá recibir una **nota distinta**.
- Se penalizará con **nota 1.0 (uno coma cero)** cualquier copia total o parcial de otro grupo o fuente externa.

Entregables:

- Modelo de Dominio y Diagrama de clases.
- Código fuente del programa.
- Archivos txt que se ocuparon.

Criterios de evaluación:

Criterio	Puntaje	Descripción detallada
Orden en nombres de variables	Del 0 al 15	- **12-15 pts.:** Nombres claros y consistentes. - **8-11 pts.:** Mayormente claros, con fallos menores. - **4-7 pts.:** Ambiguos o genéricos.

		- **0-3:** Confusos e inconsistentes.
Estructura de código	Del 0 al 20	- **16-20 pts.:** Excelente organización y separación lógica. - **11-15 pts.:** Buena organización, con mejoras posibles. - **6-10 pts.:** Desorganizado, "código espagueti". - **0-5 pts.:** Sin estructura clara.
Modelo de Dominio	Del 0 al 10	- **9-10 pts.:** Representación fiel y eficiente del problema. - **6-8 pts.:** Funcional, pero con imprecisiones. - **3-5 pts.:** Incompleto, omite elementos clave. - **0-2 pts.:** Incorrecto o inexistente.
Diagrama de Clases	Del 0 al 15	- **12-15 pts.:** Preciso, completo y consistente con el código. - **8-11 pts.:** Representa la estructura con fallos menores. - **4-7 pts.:** Esquemático, omite detalles importantes. - **0-3 pts.:** Inexistente o no corresponde al código.
Menú Administrador	Del 0 al 82	Funcionalidades del Sistema Administrador.
Menú Usuario	Del 0 al 59	Funcionalidades del Sistema Usuario.
Repositorio GitHub	Del 0 al 20	**- 16-20 pts.:** Uso excelente (commits, ramas, README). - **11-15 pts.:** Uso bueno, con fallos en documentación o ramas. - **6-10 pts.:** Uso básico (pocos commits, mensajes pobres). - **0-5 pts.:** Mínima actividad o desorganizado.

Puntaje total: 221
Puntaje nota mínima: 131
Fecha de Inicio: lunes 20 de octubre.
Fecha máxima de entrega: viernes 21 de noviembre.

Correos: daniel.duran02@alumnos.ucn.cl, nicolas.rojas11@alumnos.ucn.cl
Grupo de WhatsApp: https://chat.whatsapp.com/GvnkfE5RUViJ0AgVtyusbs?mode=ac_t