Preparcial 2 - 15 de octubre de 2025

DATOS GENERALES

- **Asignatura:** Programación II
- **Núcleo temático 2:** Preparcial 2
- **Tema a evaluar:** Parcial 1 (Builder, Singleton, Prototype, Factory Method) + Patrones estructurales (Adapter, Bridge, Composite, Decorator, Facade)

Desarrollar el siguiente contexto haciendo uso del pensamiento computacional.

1. Contexto: FitHub+ – Centro Deportivo y de Bienestar

Un centro deportivo llamado FitHub+ ofrece servicios (clases dirigidas, zonas de musculación, spa) y planes (mensual, trimestral, anual). Cada cliente puede reservar servicios, agregar extras (locker premium, toalla, bebidas isotónicas, entrenador personal), pagar con distintos medios y consultar su historial. La gerencia desea una aplicación de escritorio (JavaFX) para:

- Administrar el catálogo de servicios y planes.
- Tomar reservas con extras y calcular el costo total.
- Integrar fuentes de datos externas (CSV/JSON) con proveedores y horarios.
- Soportar distintos medios de pago conectados a pasarelas reales o simuladas.
- Ofrecer un “panel simplificado” para operaciones frecuentes.

La solución debe usar los patrones solicitados y proveer métodos lógicos: ordenar alfabéticamente, filtrar por rango de precio, buscar por palabra clave, calcular totales, generar reportes por fecha, y ranking de servicios más reservados.

Mapeo de Patrones (qué y cómo aplicarlos)

- Composite – Catálogo de Servicios
  - Idea: modelar servicios como un árbol.
  - ServiceComponent (interface) con operaciones comunes: getNombre(), getPrecioBase(), calcularPrecio(), listarHojas().
  - Hojas: ClaseDirigida (ej. Spinning, Yoga), Zona (Musculación), Spa.
  - Compuesto: PaqueteServicio (p. ej. “Combo Bienestar” que agrupa Spa + Clase Yoga).
  - Vista sugerida: TreeView para visualizar la jerarquía.
  - Lógica requerida:
    - Ordenar hijos alfabéticamente (ordenarPorNombreAsc()).
    - Filtrar servicios por precio máximo.
    - Buscar servicio por texto (nombre/etiquetas).
- Decorator – Extras en la Reserva
  - Idea: envolver un servicio reservado con extras que suman costo.
  - Reserva (componente base) con getDescripcion() y getCosto().
  - ReservaBase envuelve un ServiceComponent.
  - Decoradores: ConLockerPremium, ConToalla, ConBebida, ConEntrenadorPersonal.
  - Lógica requerida:

    - Calcular costo total incremental.
    - Listar extras aplicados en la descripción.

- Adapter – Integración de datos externos
    - Idea: el sistema interno espera ProveedorHorarioPort para obtener horarios y capacidad; se debe adaptar distintas fuentes (CSV).
    - ProveedorHorarioPort (target): List<HorarioDTO> cargarHorarios()
    - Adapters:
        - CsvHorarioAdapter lee horarios.csv.
        - JsonHorarioAdapter lee horarios.json.
    - Lógica requerida:
        - Validar y normalizar datos (fechas, cupos).
        - Reportar inconsistencias (capacidad negativa, solapamientos).

- Bridge – Medio de pago vs. Pasarela
    - Idea: separar la abstracción (tipo de pago) de su implementación (pasarela).
    - Abstracción: Pago con procesar(Monto); refinamientos: PagoTarjeta, PagoPSE, PagoEfectivo.
    - Implementor: PasarelaPago con autorizar()/capturar(). Implementaciones: MercadoPagoGateway, PayPalGateway, PasarelaDummy.
    - Lógica requerida:
        - Simular autorizaciones y rechazos.
        - Cambiar de pasarela en tiempo de ejecución sin tocar la abstracción.

- Facade – Flujo “Reservar rápido”
    - Idea: exponer una fachada para la operación frecuente “reservar servicio con extras y pagar”.
    - ReservaFacade con métodos:
        - ReservaDTO reservarRapido(Cliente c, ServiceComponent s, List<Extra> extras, MedioPago mp)
        - Maneja: verificación de cupos (Adapter), construcción de la reserva (Builder), aplicación de extras (Decorator), cobro (Bridge), y persistencia.
    - Lógica requerida:
        - Devolver un resultado con estado (éxito/fallo), mensajes y total.

- Singleton – Configuración / Sesión / Repositorio
    - Idea: garantizar una única instancia para configuración de la app o acceso central a repositorios.
    - AppConfig (Singleton) con rutas de archivos, pasarela por defecto, formato de fechas, etc.
    - (Opcional) DataStore como Singleton in-memory o con conexión única.

- Builder – Construcción de Reservas Complejas
    - Idea: crear reservas paso a paso, evitando constructores enormes.

- ReservaBuilder con pasos: seleccionar cliente, servicio base, fecha/hora, extras, promociones y medio de pago.
- Director opcional para recetas comunes (p. ej. “ReservaExpress” vs. “ReservaPersonalizada”).
- Lógica requerida:
  - Validaciones encadenadas (fechas válidas, cliente activo, cupos).

Requisitos Funcionales Clave (resumen)

1. Catálogo
   - CRUD de servicios y paquetes (Composite).
   - Ordenar alfabéticamente y por precio.
   - Filtrar por rango de precio y tipo.
2. Reservas
   - Crear reserva con extras (Decorator + Builder).
   - Calcular costos, impuestos y descuentos.
   - Ver disponibilidad (Adapter).
   - Confirmar y cobrar (Bridge).
3. Integraciones
   - Importar horarios desde CSV o JSON (Adapter).
   - Reportar filas inválidas y mostrar resumen.
4. Flujo rápido
   - ReservaFacade.reservarRapido(...) para agilizar la operación.
5. Reportes
   - Histórico por cliente y por servicio.
   - Top 5 servicios más reservados en un rango de fechas.
   - Ingresos totales por día/mes.

Requisitos de Interfaz (JavaFX)

- **Pantalla 1 – Dashboard:** KPIs (reservas hoy, ingresos, ocupación).
- **Pantalla 2 – Catálogo:**
  - TreeView (Composite), TableView detalle, TextField búsqueda, ComboBox ordenamiento (A-Z, precio).
- **Pantalla 3 – Nueva Reserva:**
  - Selector de servicio (árbol), DatePicker, ComboBox hora (desde Adapter), lista de CheckBox de extras (Decorator), botón “Reservar rápido” (Facade), sección de pago (Bridge).
- **Pantalla 4 – Importación:**
  - Botones “Importar CSV”, tabla de previsualización, área de errores.
- **Pantalla 5 – Reportes:**
  - Filtros por fecha, TableView resultados, exportación a CSV.