Entregables - Prácticas de Programación Orientada a Objetos Java

1 Instrucciones Generales

1.1 Requisitos de Originalidad del Código

- Los códigos entregados deben ser ÚNICOS por estudiante
- NO se permiten códigos iguales o muy similares entre alumnos
- Cada práctica incluye elementos personalizados que deben reflejar información específica del estudiante
- Se verificará la originalidad mediante herramientas de detección de similitud
- Cualquier código duplicado resultará en calificación de 0 para todos los involucrados

1.2 Elementos de Personalización Requeridos

- Usar las iniciales del nombre completo del estudiante en nombres de clases principales
- Incluir los últimos 4 dígitos del número de matrícula en constantes numéricas
- Incorporar el día y mes de nacimiento o nombre en valores por defecto.

2 Criterios de Evaluación

2.1 Para cada práctica individual (1-11):

- **Funcionalidad completa (50%):** Todos los requisitos implementados correctamente

- **Calidad del código (30%):** Buenas prácticas, legibilidad, estructura adecuada
- **Manejo de errores (20%):** Validaciones apropiadas y manejo de excepciones

2.2 Para el proyecto final (Práctica 12):

- **Arquitectura y diseño (25%):** Implementación correcta del patrón MVC y diseño modular
- **Funcionalidad integral (25%):** Todos los módulos funcionando correctamente e integrados
- **Interfaz de usuario (25%):** Usabilidad, diseño intuitivo y experiencia de usuario
- **Calidad técnica (25%):** Código limpio, eficiente y bien estructurado

2.3 Entrega:

- **Código fuente completo** en estructura de proyecto organizada por carpetas, donde en cada carpeta con nombre Practica## y en cada carpeta todos los carchivos necesarios. Esto dentro de un repositorio de git.
- **Video demostración** para el proyecto final (5-10 minutos)

3 Practicas

3.1 Práctica 1: Introducción a Java y Configuración del Entorno

3.1.1 Entregables

1. Captura de pantalla del JDK instalado mostrando la versión
2. Captura de pantalla del IDE configurado con el workspace
3. Proyecto Java funcional con estructura básica
4. Programa "HolaMundo[InicialNombre][InicialApellido].java" que muestre:
    - Nombre completo del estudiante

- Número de matrícula
- Fecha actual del sistema

5. Archivo README.md documentando el proceso de instalación
6. Video de 2-3 minutos mostrando la ejecución y depuración del programa

3.2 Práctica 2: Clases y Objetos en Java

3.2.1 Entregables

1. Clase "Estudiante[InicialNombre][InicialApellido].java" con:
   - Atributos: nombre, matricula, edad, carrera, semestreActual
   - Mínimo 3 constructores diferentes
   - 5 métodos de instancia relacionados con operaciones estudiantiles
2. Clase "Universidad[UltimosDigitosMatricula].java" que contenga:
   - Array de estudiantes
   - Métodos para agregar, buscar y mostrar estudiantes
3. Clase principal que demuestre la creación y uso de al menos 5 objetos diferentes

3.3 Práctica 3: Encapsulamiento y Modificadores de Acceso

3.3.1 Entregables

1. Clase "CuentaBancaria[DiaNacimiento][MesNacimiento].java" con:
   - Todos los atributos privados
   - Métodos getter y setter apropiados
   - Validaciones en los setters usando los últimos dígitos de matrícula
   - Método toString() personalizado
2. Clase "Cliente[ApellidoPaterno].java" que demuestre:
   - Uso correcto de modificadores protected
   - Relación de composición con CuentaBancaria
3. Pruebas unitarias que valide el encapsulamiento correctamente
4. Pruebas unitarias para validar la funcionalidad

3.4 Práctica 4: Herencia

3.4.1 Entregables

1. Jerarquía de clases "Vehiculo[InicialNombre]":
   - Clase padre: VehiculoBase[InicialNombre]
   - Clases hijas: Auto[ApellidoPaterno], Motocicleta[ApellidoPaterno], Camion[ApellidoPaterno]
2. Implementación de al menos 3 métodos sobrescritos por clase hija
3. Uso correcto de super() en constructores y métodos
4. Clase "Concesionaria[UltimosDigitosMatricula]" que maneje la colección de vehículos
5. Programa principal que demuestre polimorfismo con la herencia
6. Diagrama UML completo de la jerarquía de herencia
7. Pruebas unitarias para cada clase de la jerarquía

3.5 Práctica 5: Polimorfismo

3.5.1 Entregables

1. Interfaz "Calculable[InicialNombre]" con métodos para cálculos específicos
2. Jerarquía de figuras geométricas:
   - Clase abstracta "Figura[DiaNacimiento]"
   - Clases concretas: Circulo[ApellidoPaterno], Rectangulo[ApellidoPaterno], Triangulo[ApellidoPaterno]
3. Implementación de sobrecarga de métodos (mínimo 3 sobrecargas por clase)
4. Clase "CalculadoraGeometrica[UltimosDigitosMatricula]" que use polimorfismo
5. Demostración de casting y instanceof en clase principal
6. Programa que procese un array polimórfico de figuras

3.6 Práctica 6: Clases Abstractas e Interfaces

3.6.1 Entregables

1. Clase abstracta "Empleado[InicialNombre][InicialApellido]" con:
   - Métodos abstractos y concretos
   - Atributos protegidos con información personalizada
2. Interfaces:
   - "Bonificable[DiaNacimiento]"
   - "Evaluable[MesNacimiento]"
   - "Promovible[UltimosDigitosMatricula]"
3. Clases concretas que implementen múltiples interfaces:
   - "Gerente[ApellidoPaterno]"
   - "Desarrollador[ApellidoPaterno]"
   - "Vendedor[ApellidoPaterno]"
4. Clase "EmpresaTI[InicialNombre][UltimosDigitosMatricula]" para gestionar empleados
5. Programa principal demostrando el uso conjunto de clases abstractas e interfaces

3.7 Práctica 7: Manejo de Excepciones

3.7.1 Entregables

1. Jerarquía de excepciones personalizadas:
   - "Exception[ApellidoPaterno]Base" (clase padre)
   - "Matricula[DiaNacimiento]InvalidaException"
   - "Saldo[MesNacimiento]InsuficienteException"
   - "Usuario[UltimosDigitosMatricula]NoEncontradoException"
2. Clase "SistemaBanco[InicialNombre][InicialApellido]" con:
   - Métodos que lancen excepciones personalizadas
   - Manejo de múltiples tipos de excepciones

- Logging de errores con información del estudiante

3. Implementación de try-with-resources
4. Prueba unitaria que demuestre el manejo correcto de excepciones
5. Archivo de log generado durante las pruebas

3.8 Práctica 8: Colecciones en Java

3.8.1 Entregables

1. Clase "Biblioteca[InicialNombre][UltimosDigitosMatricula]" que utilice:
   - ArrayList para libros disponibles
   - LinkedList para cola de reservas
   - HashMap para usuarios registrados (key: matrícula personalizada)
   - HashSet para categorías únicas
2. Clase "Libro[DiaNacimiento][MesNacimiento]" con implementación de Comparable
3. Implementación de Comparator personalizado para diferentes criterios de ordenamiento
4. Métodos que demuestren:
   - Búsqueda y filtrado usando streams
   - Operaciones CRUD completas
   - Manejo de iteradores
5. Programa principal con menú interactivo
6. Datos de prueba que incluyan información personalizada del estudiante
7. Programa que analice la complejidad temporal de las operaciones implementadas o que muestre el tiempo de cada operacion.

3.9 Práctica 9: Entrada/Salida (I/O) en Java

3.9.1 Entregables

1. Sistema de gestión de archivos "GestorArchivos[ApellidoPaterno][UltimosDigitosMatricula]":
   - Lectura/escritura de archivos de texto
   - Manejo de archivos binarios
   - Operaciones con directorios
2. Serialización de objetos:
   - Clase "Persona[InicialNombre]Serializable"
   - Métodos para guardar/cargar objetos serializados
3. Procesamiento de archivos CSV con datos estudiantiles personalizados
4. Implementación de backup automático con timestamp
5. Archivos de prueba generados:
   - "datos_[matricula].txt"
   - "backup_[diaNacimiento][mesNacimiento].dat"
   - "log_[apellidoPaterno].csv"
6. Programa principal con interfaz de línea de comandos

3.10 Práctica 10: Programación Concurrente

3.10.1 Entregables

1. Simulador de banco con hilos "BancoConcurrente[InicialNombre][InicialApellido]":
   - Clase "CajeroThread[DiaNacimiento]" que extienda Thread
   - Clase "ClienteRunnable[MesNacimiento]" que implemente Runnable
   - Sincronización de acceso a cuentas bancarias
2. Implementación de patrón Productor-Consumidor:
   - Buffer compartido con tamaño basado en últimos dígitos de matrícula
   - Múltiples productores y consumidores
3. Pool de hilos personalizado "ThreadPool[ApellidoPaterno]"

4. Demostración de:
    - Uso de synchronized, wait(), notify()
    - Manejo de deadlocks y race conditions
    - Uso de ExecutorService
5. Programa principal que lance múltiples hilos concurrentes
6. Análisis de rendimiento y logs de ejecución
7. Documentación sobre estrategias de sincronización utilizadas

3.11 Práctica 11: JavaFX para Interfaces Gráficas

3.11.1 Entregables

1. Aplicación GUI "SistemaEstudiantil[InicialNombre][UltimosDigitosMatricula]":
    - Ventana principal con menú personalizado
    - Formularios CRUD para gestión de estudiantes
    - Tabla con datos dinámicos
2. Componentes personalizados:
    - "Campo[ApellidoPaterno]Validado" (TextField con validaciones)
    - "Boton[DiaNacimiento]Estilizado" (Button con estilos CSS personalizados)
3. Implementación de:
    - Manejo de eventos de mouse y teclado
    - Validación de formularios en tiempo real
    - Diálogos modales personalizados
4. Archivo CSS "estilos_[matricula].css" con temas personalizados
5. Imágenes y recursos gráficos en carpeta "recursos_[apellidoPaterno]"
6. Ejecutable JAR funcional de la aplicación
7. Manual de usuario con capturas de pantalla

4 Producto Integrador: Sistema de Punto de Venta de Suscripciones de Gimnasio

4.1 Entregables

1. Aplicación completa "GymPOS[InicialNombre][InicialApellido][UltimosDigitosMatricula]" que integre:
   - Interfaz gráfica JavaFX profesional
   - Base de datos simulada con archivos
   - Sistema de autenticación de empleados
   - Manejo de excepciones robusto
   - Programación concurrente para procesos en segundo plano
2. Módulos principales:
   - **GestionClientes[ApellidoPaterno]**: CRUD completo de clientes con validaciones
   - **SistemaMembresias[DiaNacimiento][MesNacimiento]**: Diferentes tipos de suscripciones
   - **ProcesadorPagos[UltimosDigitosMatricula]**: Simulación de procesamiento de pagos
   - **GeneradorReportes[InicialNombre]**: Reportes en PDF/TXT con datos estadísticos
   - **ControlAcceso[ApellidoPaterno]**: Sistema de entrada/salida del gimnasio
3. Funcionalidades específicas:
   - Registro de nuevas suscripciones con descuentos personalizados
   - Renovación automática de membresías
   - Sistema de puntos/recompensas
   - Calendario de clases grupales
   - Control de inventario de equipos
   - Notificaciones automáticas por vencimiento
4. Características técnicas obligatorias:
   - Patrón MVC implementado correctamente

- Mínimo 15 clases organizadas en packages
- Serialización para persistencia de datos
- Multithreading para tareas pesadas (reportes, backups)
- Manejo completo de excepciones personalizadas
- Interfaz responsive y profesional

5. Archivos de entrega:
   - Código fuente completo en estructura de packages
   - Ejecutable JAR con todas las dependencias
   - Base de datos inicial con 20+ registros de prueba personalizados
   - Manual técnico (10-15 páginas) con diagramas UML
   - Manual de usuario con casos de uso
   - Video demostración (5-8 minutos) mostrando todas las funcionalidades
   - Archivo de configuración con datos del estudiante
6. Datos personalizados requeridos:
   - Nombre del gimnasio debe incluir apellido del estudiante
   - Precios base calculados usando últimos dígitos de matrícula en los centavos
   - Empleado administrador con datos del estudiante
   - Logotipo personalizado con iniciales

4.2 Criterios de evaluación específicos:

- Originalidad y creatividad en la implementación (20%)
- Correcta aplicación de conceptos POO (25%)
- Funcionalidad completa y manejo de errores (25%)
- Interfaz de usuario y experiencia (15%)
- Calidad del código y documentación (15%)