PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 3
4ta práctica (tipo b)
(Primer Semestre 2025)

Indicaciones Generales:
Duración: **110 minutos.**

- Se les recuerda que, de acuerdo al reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otro estudiante o cometer plagio para el desarrollo de esta práctica.
- Se permite el uso de apuntes de clase, diapositivas, ejercicios prácticos y código fuente. Sin embargo, todo este material debe descargarse antes de comenzar a resolver el enunciado.
- Se permite el uso de Internet exclusivamente para consultar páginas oficiales de Microsoft y Oracle. Sin embargo, cualquier forma de comunicación con otros estudiantes o terceros está estrictamente prohibida.

Puntaje total: 20 puntos

- Cada propuesta de solución a cada pregunta deberá subirse a la plataforma `Paideia` en un archivo comprimido en formato `ZIP`. No se aceptarán los trabajos compactados con otros programas como `RAR`, `WinRAR`, `7zip` o similares. Es importante asegurarse de incluir únicamente el código fuente, excluyendo el código objeto y las librerías.
- Cada archivo deberá tener el siguiente formato de nombre "Lab04_2025_1_CO_PA_PN_PR" donde: **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno, **PN** indica: primer nombre y **PR** indica: número de la pregunta pudiendo ser PR igual a P1, P2 o P3 únicamente. De no colocar este requerimiento se le descontará 3 puntos de la nota final. Un ejemplo de formato podría ser el siguiente: "Lab04_2025_1_19941146_Melgar_Héctor_P2".
- En la plataforma `Paideia` se encuentran 2 archivos que puede utilizar en su propuesta de solución: `Principal_P1.java` y `Principal_P2.java` que contienen el código del método `main` de cada pregunta. El archivo `Principal_P1.java` contiene el `main` de la pregunta 1 y el archivo `Principal_P2.java` contiene el `Main` de la pregunta 2.

Cuestionario

- Los objetivos de este laboratorio son:
  - Reforzar los conceptos del curso *Programación 2* relacionados con el Paradigma Orientado a Objetos, con énfasis en los mecanismos de **encapsulamiento**, **herencia** y **polimorfismo**.
  - Implementar propuestas de solución aplicando los mecanismos del Paradigma Orientado a Objetos y librerías, utilizando el lenguaje de programación `Java`.
  - Aplicar las mejores prácticas de programación adquiridas a lo largo de la carrera, incluyendo los principios de DRY (*Don't Repeat Yourself*), prevención del hardcoding, nomenclatura semántica y coherente para clases y métodos, así como otros principios de diseño y mantenibilidad del software.

Pregunta 1 (10 puntos)
En el contexto de las bases de datos, una **entidad** es cualquier objeto, concepto o cosa del mundo real

que tiene importancia para un sistema y sobre el cual se desea almacenar información. Cada entidad suele tener un **nombre** y un conjunto de **atributos** que describen la entidad. Así, por ejemplo la entidad `Persona` podría tener como atributos `id`, `nombres`, `apellidos paternos` y `apellidos maternos`.

Se desea crear un aplicativo en Java, sin utilizar IDE, que permita gestionar entidades en memoria para lo cual se solicita que:

- (1 punto) Cree el paquete `pe.edu.pucp.progra03.lab04.entidad`. Este paquete debe contener las clases `Entidad`, `Columna` y `Fila`, así como el enumerado `TipoDeDato`. La descripción de las clases y el enumerado se realizan al finalizar la pregunta.
- (1 punto) Implementar el enumerado `TipoDeDato`.
- (1 punto) Implementar la clase `Columna`.
- (2 punto) Implementar la clase `Fila`.
- (3 punto) Implementar la clase `Entidad`.
- (2 punto) Generar el `jar` denominado `pregunta1.jar` que contenga la clases compiladas de paquete `pe.edu.pucp.progra03.lab04.entidad`. Los comandos para compilar y generar librerías se detallan al finalizar esta pregunta.

Para probar su librería, se presenta la clase principal `Principal_P1.java`, la cual genera la salida que se presenta en la `Consola_P1`.

```java
package pe.edu.pucp.progra03.lab04;

import java.util.ArrayList;
import pe.edu.pucp.progra03.lab04.entidad.Columna;
import pe.edu.pucp.progra03.lab04.entidad.Entidad;
import pe.edu.pucp.progra03.lab04.entidad.TipoDeDato;

public class Principal_P1 {

    public static void main(String[] args) {
        ArrayList<Columna> listaColumnas = new ArrayList<>();
        listaColumnas.add(new Columna("id", TipoDeDato.NUMERO));
        listaColumnas.add(new Columna("nombre", TipoDeDato.CADENA));
        listaColumnas.add(new Columna("paterno", TipoDeDato.CADENA));
        listaColumnas.add(new Columna("materno", TipoDeDato.CADENA));

        Entidad entidad = new Entidad("Alumno", listaColumnas);
        entidad.insertarFila();
        entidad.agregarEnteroEnFila(19941146);
        entidad.agregarCadenaEnFila("Héctor Andrés");
        entidad.agregarCadenaEnFila("Melgar");
        entidad.agregarCadenaEnFila("Sasieta");
        entidad.insertarFila();
        entidad.agregarEnteroEnFila(20112728);
        entidad.agregarCadenaEnFila("Freddy Alberto");
        entidad.agregarCadenaEnFila("Paz");
        entidad.agregarCadenaEnFila("Espinoza");
        System.out.println(entidad);
    }
}
```

Programa 1: clase Principal_P1.java

Consola_P1

```
id,nombre,paterno,materno
19941146,Héctor Andrés,Melgar,Sasieta
20112728,Freddy Alberto,Paz,Espinoza
```

El enumerado `TipoDeDato` pertenece al paquete `pe.edu.pucp.progra03.lab04.entidad` y define dos posibles tipos de datos: `NUMERO` y `CADENA`.

La clase `Columna`, ubicada en el paquete `pe.edu.pucp.progra03.lab04.entidad`, representa una estructura de datos que modela una entidad. Esta clase contiene dos atributos privados:

- `nombre` (de tipo `String`): representa el nombre de la columna.
- `tipoDeDato` (de tipo `TipoDeDato`): indica el tipo de dato que almacena la columna, utilizando el enumerado previamente definido (`NUMERO` o `CADENA`).

La clase `Fila`, ubicada en el paquete `pe.edu.pucp.progra03.lab04.entidad`, representa una estructura de datos que modela una fila de valores, asociada a una entidad, en donde cada posición almacena un dato correspondiente a una columna.

La clase `Fila` posee un único atributo: `listaDatos`. `listaDatos` es una lista (`ArrayList`) de tipo `Object` que almacena los diferentes valores de la fila. Al usar `Object` como tipo genérico, la clase puede almacenar cualquier tipo de dato, aunque en este caso específico solo se insertan enteros (`Integer`) y cadenas (`String`), como se observa en los métodos definidos.

Esta clase contiene los siguientes métodos:

- `public void insertarEntero(Integer dato)`: agrega un valor entero al final de la lista de datos.
- `public void insertarCadena(String dato)`: agrega una cadena de texto al final de la lista de datos.
- `public Object obtenerDato(Integer indice)`: devuelve el objeto ubicado en la posición especificada de la lista.

La clase `Entidad`, que forma parte del paquete `pe.edu.pucp.progra03.lab04.entidad`, modela la estructura de la entidad, en donde se agrupan columnas (`Columna`) y filas (`Fila`) que almacenan datos estructurados bajo un mismo contexto o conjunto.Esta clase contiene los siguientes atributos privados:

- `nombre (String)`: representa el nombre de la entidad.
- `listaColumnas (ArrayList<Columna>)`: lista de columnas que definen la estructura de la entidad. Es `protected`, lo cual permite su uso directo en clases hijas.
- `listaFilas (ArrayList<Fila>)`: lista de filas, donde cada `Fila` contiene los datos correspondientes a una instancia de la entidad.

Esta clase contiene los siguientes métodos:

- `public void insertarFila()`: agrega una nueva fila vacía (`Fila`) al final de listaFilas.
- `public void agregarEnteroEnFila(Integer entero)`: inserta un valor entero en la última fila creada.
- `public void agregarCadenaEnFila(String cadena)`: inserta una cadena de texto en la última fila creada.

Consideraciones para compilación por línea de comandos

Si el paquete con código fuente `pe.edu.pucp.progra03.lab04.entidad` se encontrase dentro de la carpeta `scr` y se desease tener el paquete compilado en la carpeta `build`, se puede ejecutar el siguiente comando:

```
javac src/pe/edu/pucp/progra03/lab04/entidad/*.java -d build
```

Consideraciones para la generación del jar por línea de comandos

Si el paquete compilado `pe.edu.pucp.progra03.lab04.entidad` se encontrase dentro de la carpeta `build` y se desease generar el `jar` en la carpeta `lib`, se puede ejecutar el siguiente comando:

```
cd build

jar cvf ../lib/pregunta1.jar pe/edu/pucp/progra03/lab04/entidad/*.class
```

Consideraciones para compilar la clase principal usando el jar generado

Si el paquete compilado `pe.edu.pucp.progra03.lab04.entidad` se encontrase empaquetado en un `jar` dentro de la carpeta `lib`. Se puede compilar la clase principal de la siguiente manera de forma tal que se quede compilado en la carpeta `build_principal`

```
javac -classpath lib/pregunta1.jar src/pe/edu/pucp/progra03/lab04/Principal_P1.java
 -d build_principal
```

Si se desea ejecutar ahora la clase principal se podría hacer lo siguiente:

```
cd build_principal

java -classpath ../lib/pregunta1.jar; pe/edu/pucp/progra03/lab04/Principal_P1
```

Pregunta 2 (10 puntos)

Se desea hacer ahora que la entidad sea persistente. En el contexto de una base de datos, la persistencia se refiere a la capacidad de los datos de mantenerse almacenados de forma permanente, incluso después de que se apague el sistema o finalice un programa. Dicho de otro modo, persistencia significa que los datos no se pierden con el tiempo o al cerrar la aplicación.

Se desea crear un aplicativo en Java, sin utilizar IDE, que permita gestionar entidades persistentes en archivo para lo cual se solicita que:

- (1 punto) Cree el paquete `pe.edu.pucp.progra03.lab04.entidad.persistencia`. Este paquete debe contener la clases `EntidadPersistente`. La descripción de la clase se realizan al finaliza la pregunta.
- (1 punto) Implementar los atributos de la clase `EntidadPersistente`.
- (1 punto) Implementar el constructor de la clase `EntidadPersistente`.
- (3 punto) Implementar el método `salvarEnArchivo` de la clase `EntidadPersistente`.
- (3 punto) Implementar el método `cargarDesdeArchivo` de la clase `EntidadPersistente`.
- (2 punto) Generar el `jar` denominado `pregunta2.jar` que contenga la clases compiladas de paquete `pe.edu.pucp.progra03.lab04.entidad.persistencia`. Los comandos para compilar y generar librerías se detallan al finalizar la pregunta 1.

Para probar su librería, se presenta la clase principal `Principal_P2.java`, la cual genera la salida que se presenta en la `Consola_P2` la cual es la misma que se presenta en `Consola_P1`.

```
1 package pe.edu.pucp.progra03.lab04;
2
3 import java.util.ArrayList;
4 import pe.edu.pucp.progra03.lab04.entidad.Columna;
5 import pe.edu.pucp.progra03.lab04.entidad.TipoDeDato;
6 import pe.edu.pucp.progra03.lab04.entidad.persistencia.EntidadPersistente;
7
8 public class Principal_P2 {
```

```java

    public static void main(String[] args) {
        ArrayList<Columna> listaColumnas = new ArrayList<>();
        listaColumnas.add(new Columna("id", TipoDeDato.NUMERO));
        listaColumnas.add(new Columna("nombre", TipoDeDato.CADENA));
        listaColumnas.add(new Columna("paterno", TipoDeDato.CADENA));
        listaColumnas.add(new Columna("materno", TipoDeDato.CADENA));

        EntidadPersistente entidad = new EntidadPersistente("Alumno", listaColumnas);
        entidad.insertarFila();
        entidad.agregarEnteroEnFila(19941146);
        entidad.agregarCadenaEnFila("Héctor Andrés");
        entidad.agregarCadenaEnFila("Melgar");
        entidad.agregarCadenaEnFila("Sasieta");
        entidad.insertarFila();
        entidad.agregarEnteroEnFila(20112728);
        entidad.agregarCadenaEnFila("Freddy Alberto");
        entidad.agregarCadenaEnFila("Paz");
        entidad.agregarCadenaEnFila("Espinoza");
        entidad.salvarEnArchivo();

        EntidadPersistente entidad2 = new EntidadPersistente("Alumno", listaColumnas);
        entidad2.cargarDesdeArchivo();
        System.out.println(entidad2);
    }
}
```

Programa 2: clase Principal_P1.java

Consola_P2

```
id,nombre,paterno,materno
19941146,Héctor Andrés,Melgar,Sasieta
20112728,Freddy Alberto,Paz,Espinoza
```

La clase `EntidadPersistente`, ubicada en el paquete `pe.edu.pucp.progra03.lab04.entidad.persistencia`, extiende la funcionalidad de la clase `Entidad` añadiendo la capacidad de persistir los datos en un archivo `CSV` y de cargar datos desde dicho archivo.

Posee como atributo adicional `nombreArchivo (String)` que almacena el nombre del archivo en disco asociado a la entidad. Se construye automáticamente concatenando el nombre de la entidad con la extensión `.csv`.

Posee dos métodos: el método `salvarEnArchivo` se encarga de guardar la información contenida en la entidad en un archivo de texto con formato `CSV`. Por otro lado, el método `cargarDesdeArchivo` permite leer un archivo `CSV` previamente generado para restaurar el contenido de la entidad.

Consideraciones para generar archivos de texto

La clase `FileWriter` en `Java` es una clase de la biblioteca estándar que permite escribir caracteres en archivos de texto. Pertenece al paquete `java.io` (`import java.io.FileWriter`) y se utiliza comúnmente cuando se desea guardar información en un archivo en formato plano (texto).

Cuando se crea un objeto de tipo `FileWriter`, se inicializa indicando el nombre del archivo al que se desea escribir. Si el archivo no existe, se crea automáticamente; si ya existe, su contenido anterior será sobrescrito (por ejemplo: `FileWriter writer = new FileWriter("alumnos.csv");`).

El método `write` permite escribir en el archivo una secuencia de caracteres, que puede estar contenida en un String (por ejemplo: `writer.write(contenido);`).

El método `close` es fundamental, ya que cierra el flujo de salida al archivo. Al cerrarlo, se asegura que todos los datos almacenados temporalmente en memoria (buffer) se escriban efectivamente en el archivo físico. No llamar a close podría provocar pérdida de datos o que el archivo quede incompleto.

La clase `FileWriter`, al igual que otros elementos del paquete `java.io`, puede lanzar una excepción del tipo `IOException` (`import java.io.IOException`). Esta excepción es una clase que representa errores que pueden ocurrir durante operaciones de entrada/salida, como no poder acceder al archivo, falta de permisos, problemas con el sistema de archivos, o errores de escritura.

Consideraciones para leer archivos de texto

En `Java`, las clases `BufferedReader` y `FileReader` son parte del paquete java.io y se utilizan conjuntamente para leer texto desde archivos de manera eficiente (`import java.io.BufferedReader` e `import java.io.FileReader`).

`FileReader` es una clase diseñada para leer archivos de texto carácter por carácter. Es una forma sencilla de leer el contenido de un archivo, pero no es la más eficiente si se va a leer línea por línea o grandes volúmenes de datos, ya que accede al archivo directamente sin ningún tipo de almacenamiento intermedio (buffer).

`BufferedReader` mejora el rendimiento de lectura al envolver un `Reader` (como `FileReader`) y almacenar temporalmente bloques de texto en un buffer en memoria. Esto reduce el número de accesos directos al disco, que son operaciones costosas en términos de tiempo. Por eso, cuando se quiere leer texto línea por línea de forma eficiente, es habitual usar `BufferedReader` junto con `FileReader`.

Se usan juntas así:
`BufferedReader br = new BufferedReader(new FileReader("archivo.txt"));`
El método `readLine` es uno de los métodos más usados de `BufferedReader`. Permite leer una línea completa de texto del archivo (es decir, hasta encontrar un salto de línea). Cada vez que se llama, devuelve una cadena (`String`) con el contenido de la siguiente línea. Cuando ya no hay más líneas que leer, retorna `null`.

Un ejemplo común de uso sería:

```
String linea;
while ((linea = br.readLine()) != null) {
    // procesar la línea
}
```

Por otro lado, el método `split` es un método de la clase `String` en Java que se utiliza para dividir una cadena de texto en partes más pequeñas (subcadenas), usando como criterio un delimitador o separador definido por una expresión regular.

Ejemplo de uso:

```
String linea = "Juan,25,Lima";
String[] datos = linea.split(",");
```

Resultado:

```
datos[0] = "Juan"
datos[1] = "25"
datos[2] = "Lima"
```

Pregunta 3 (3 puntos)
Se le solicita crear una instancia del servicio MySQL en AWS RDS y, posteriormente, ejecutar el script para crear la tabla `ALUMNOS`.El script se encuentra disponible en la plataforma Paideia con el nombre

`alumno.sql`. Una vez creada la tabla en el ambiente de AWS, llame al Jefe de Práctica para que se le asigne el puntaje en esta pregunta.

Profesores del curso: Freddy Paz, Andrés Melgar, Heider Sánchez, Eric Huiza

Pando, 16 de abril de 2025