UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programación Orientada a Objetos

Sesión 15: Programación con clases persistentes y bases de datos.

Recordando:

¿Que vimos la clase pasada?

Universidad Tecnológica del Perú

Logro de aprendizaje

Al finalizar la sesión, el estudiante soluciona problemas aplicando Programación con clases persistentes y bases de datos usando Java y mysql en la resolución de ejercicios.

Universidad Tecnológica del Perú

Utilidad

La programación con clases persistentes y bases de datos ofrece múltiples beneficios para los estudiantes de sistemas, tanto en términos de conocimientos técnicos como de preparación para el entorno laboral.

Agenda

- Integración con Bases de Datos

Universidad Tecnológica del Perú

Saberes Previos

- Conceptos básicos de programación orientada a objetos.
- Familiaridad con el lenguaje de programación (preferiblemente Java).
- Comprender la importancia de la reutilización de código y la mantenibilidad del software.

Integración con Bases de Datos

La integración entre clases persistentes y bases de datos es crucial para el desarrollo de aplicaciones robustas y escalables. Este punto de la presentación aborda varios aspectos importantes:

Integración con Bases de Datos

- **Tipos de Bases de Datos:** Se discuten los diferentes tipos de bases de datos (relacionales, NoSQL, etc.) y cómo cada uno puede interactuar con clases persistentes. Se resaltan las características específicas de cada tipo y cómo se adaptan a las necesidades de almacenamiento de datos de las aplicaciones.

Integración con Bases de Datos

- **Compatibilidad y Adherencia a Estándares:** Se aborda la compatibilidad entre las clases persistentes y los estándares de las bases de datos. Esto implica considerar cómo se estructuran los datos en la base de datos y cómo se mapean esos datos a las clases persistentes para garantizar una integración fluida y coherente.

Integración con Bases de Datos

- **Operaciones CRUD (Crear, Leer, Actualizar, Eliminar):**

  Se detallan las operaciones básicas de manipulación de datos en bases de datos y cómo se traducen estas acciones al trabajar con clases persistentes.

  Se muestran ejemplos concretos de cómo se realizan operaciones de inserción, lectura, actualización y eliminación utilizando clases persistentes.

Integración con Bases de Datos

- **Optimización de Consultas y Rendimiento:**

  Se discuten estrategias para optimizar las consultas a la base de datos cuando se utilizan clases persistentes.

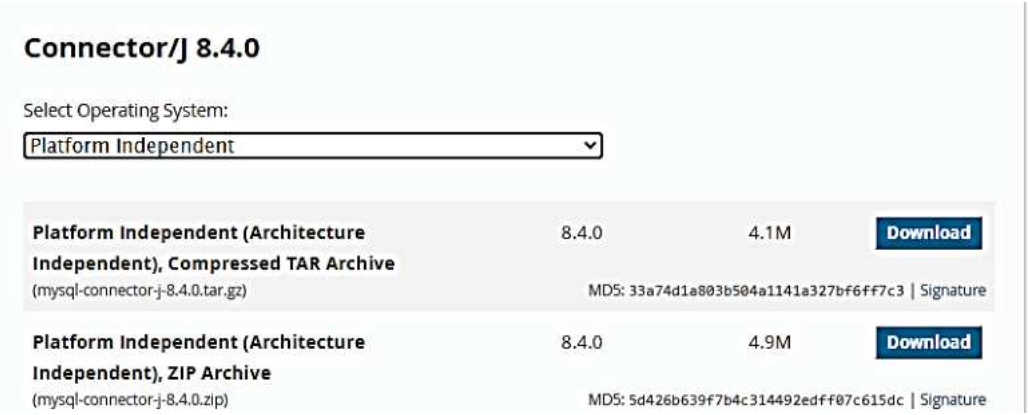
  Esto incluye el uso eficiente de índices, la minimización de consultas redundantes y el manejo de grandes conjuntos de datos para mejorar el rendimiento de la aplicación.

Requisitos

- Instalar xampp

https://www.apachefriends.org/es/download.html

- Instalar conector mysql

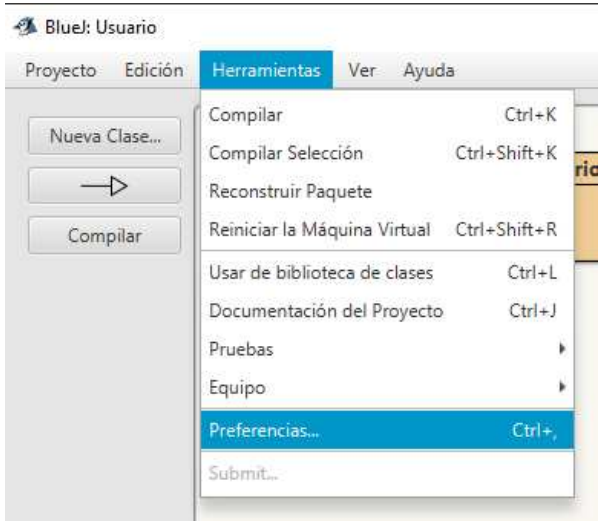


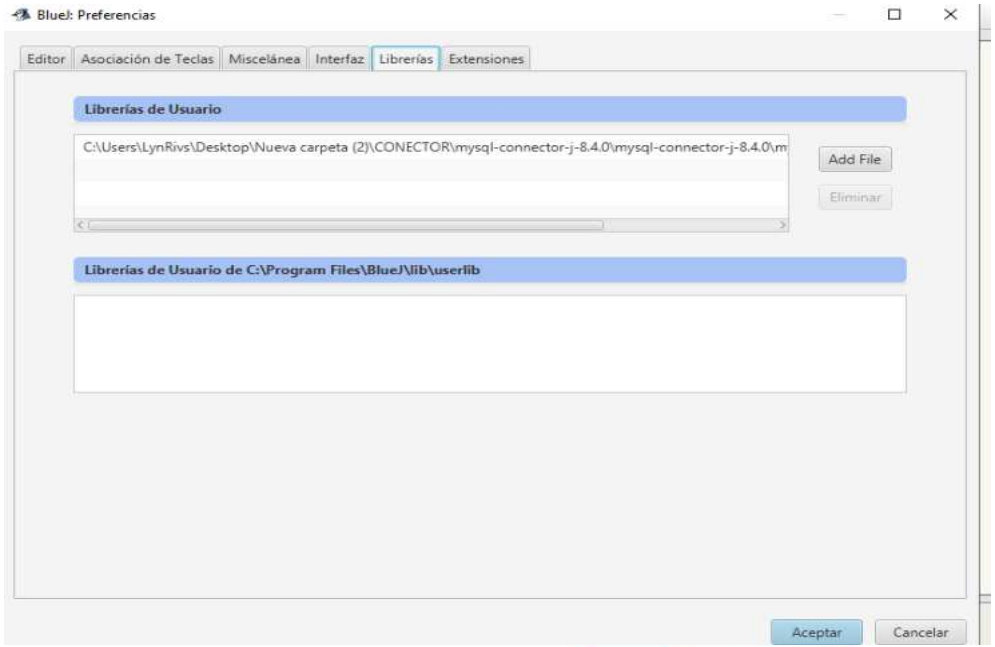
https://downloads.mysql.com/archives/c-j/

Universidad Tecnológica del Perú

Como instalar el conector

Para bluej deberás entrar en y colocar la dirección donde descargaste el conector.



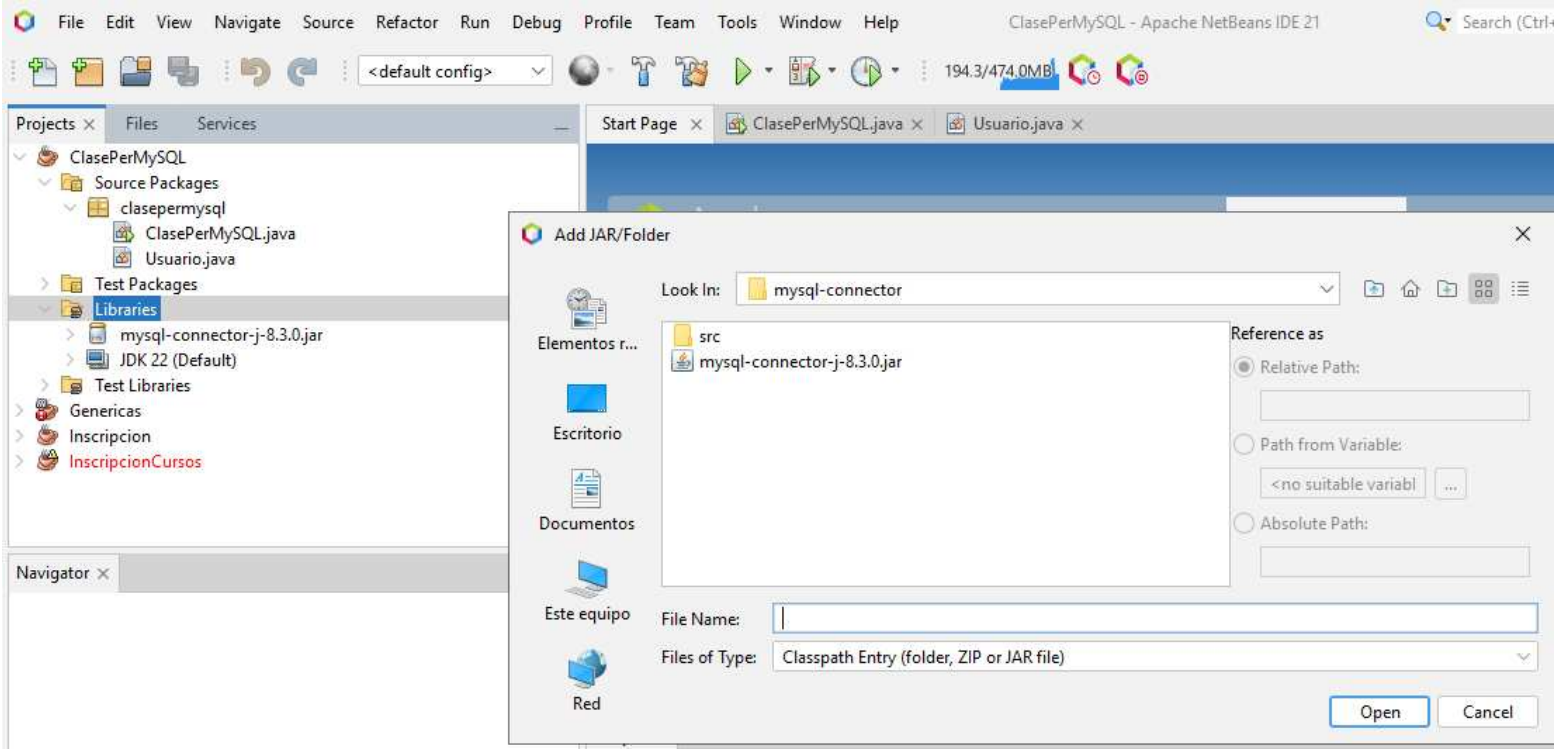


Universidad Tecnológica del Perú

Como instalar el conector

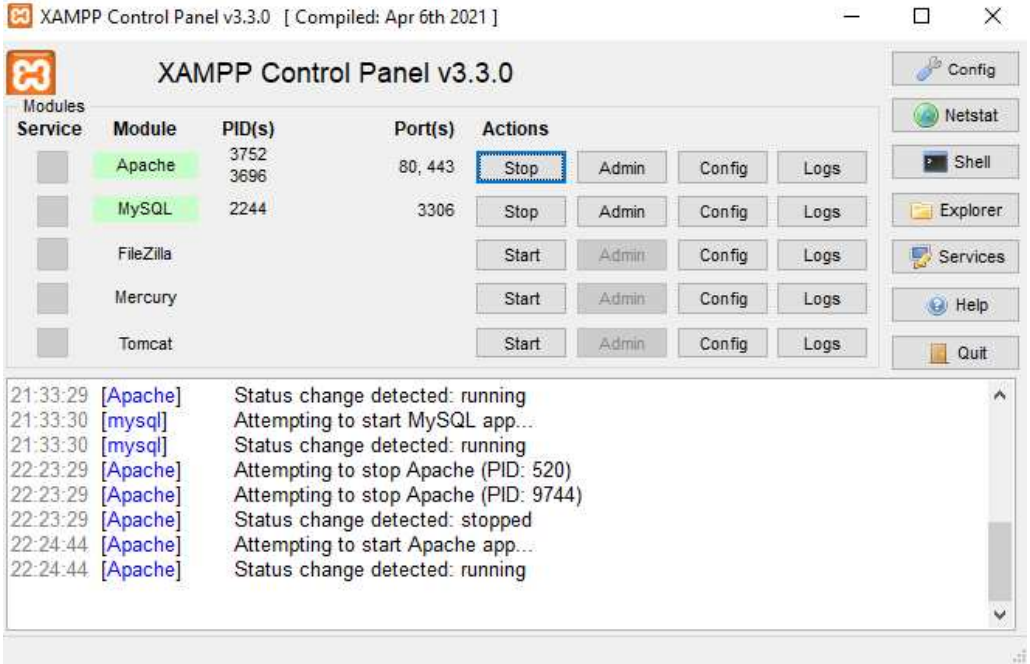
Para NetBeans en librerías/Add JAR/folder

Ubicar donde guardaste el conector descargado



Instalación de Xammp

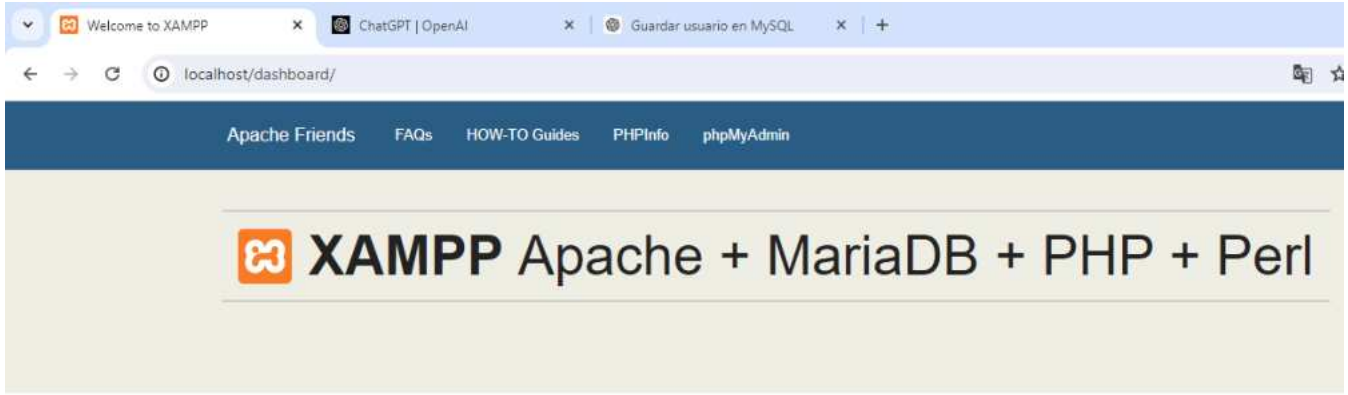
Luego de descargarlo lo deberás instalarlo y correr donde habilitaras el mysql



Universidad Tecnológica del Perú

Instalación de Xammp

Luego iras a una pagina web y colocaras localhost



Welcome to XAMPP for Windows 8.2.12

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others.

Start the XAMPP Control Panel to check the server status.

Universidad Tecnológica del Perú

Usando phpMyAdmin

Aquí deberás elegir phpMyAdmin y crear la base de datos a usar

Welcome to XAMPP for Windows 8.2.12

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others.

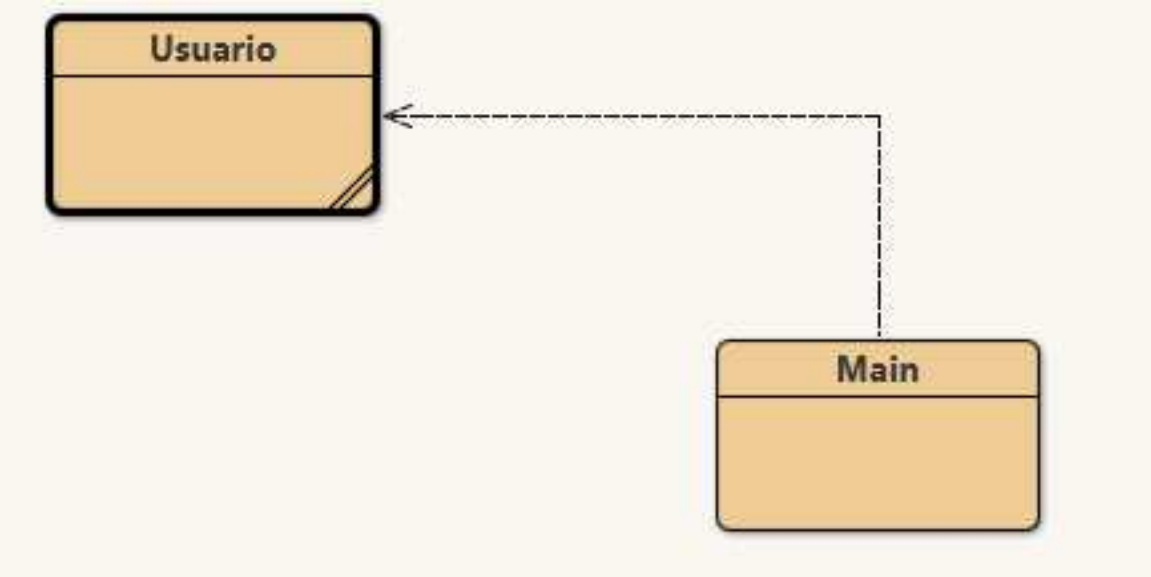
Universidad Tecnológica del Perú

Usando phpMyAdmin

Aquí deberás elegir phpMyAdmin y crear la base de datos a usar

```
--Crear la base de datos
 CREATE DATABASE tu_base_de_datos;
--Usar la base de datos
USE tu_base_de_datos;
 -- Crear la tabla Usuarios
CREATE TABLE Usuarios ( id INT
AUTO_INCREMENT PRIMARY KEY, nombre
VARCHAR(100) NOT NULL, email
VARCHAR(100) NOT NULL );
```

Ejercicio de conexión de mysql y java



```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Usuario {
    private int id;
    private String nombre;
    private String email;

    // Constructores, getters y setters
    public Usuario(int id, String nombre, String email) {
        this.id = id;
        this.nombre = nombre;
        this.email = email;
    }

    public Usuario() {
    }
```

Clase usuario

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
```

Clase usuario

Clase usuario

```
// Método para guardar el usuario en la base de datos
   public void guardarUsuario() {
      String url = "jdbc:mysql://localhost:3306/tu_base_de_datos";
      String user = "root";
      String password = ""; // Asegúrate de que la contraseña sea correcta

      try (Connection con = DriverManager.getConnection(url, user, password);
          PreparedStatement pst = con.prepareStatement("INSERT INTO Usuarios (nombre, email) VALUES (?, ?)")) {

         pst.setString(1, this.nombre);
         pst.setString(2, this.email);
         pst.executeUpdate();
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Clase Main

En tu proyecto, podrías instanciar un objeto Usuario, establecer sus atributos y luego llamar al método guardarUsuario() para almacenarlo en la base de datos:

```
public class Main {
    public static void main(String[] args) {
        Usuario usuario = new Usuario();
        usuario.setNombre("Ejemplo");
        usuario.setEmail("ejemplo@correo.com");
        usuario.guardarUsuario();
    }
}
```

Explicación del codigo

Este código Java se encarga de definir una clase Usuario que puede guardar sus datos en una base de datos MySQL. Aquí está el desglose:Importaciones Necesarias

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
```

- **java.sql.Connection:** Interface para la conexión a la base de datos
- **java.sql.DriverManager:** Clase para manejar la conexión a la base de **datos**
- **java.sql.PreparedStatement:** Interface para ejecutar consultas SQL precompiladas.
- **java.sql.SQLException:** Clase para manejar excepciones SQL.

Explicación del codigo

Campos de la Clase:
id, nombre, email: Variables que almacenan la información del usuario.
Constructores:

- **Constructor con parámetros**: Permite inicializar un objeto Usuario con valores específicos.
- **Constructor vacío**: Permite crear un objeto Usuario sin inicializar sus campos.
- **Getters y Setters:** Métodos para obtener y establecer los valores de id, nombre, y email.

Explicación del codigo

Método para Guardar el Usuario en la Base de Datos

```
// Método para guardar el usuario en la base de datos
  public void guardarUsuario() {
     String url = "jdbc:mysql://localhost:3306/tu_base_de_datos"; // URL de conexión a la base de datos
     String user = "root"; // Usuario de la base de datos
     String password = ""; // Contraseña de la base de datos

     try (Connection con = DriverManager.getConnection(url, user, password); // Establecer la conexión
        PreparedStatement pst = con.prepareStatement("INSERT INTO Usuarios (nombre, email) VALUES (?, ?)")) {
// Preparar la consulta SQL

        pst.setString(1, this.nombre); // Establecer el valor del primer parámetro (nombre)
        pst.setString(2, this.email); // Establecer el valor del segundo parámetro (email)
        pst.executeUpdate(); // Ejecutar la consulta
     } catch (SQLException e) {
        //e.printStackTrace(); // Manejar excepciones SQL
     }
  }
```

Explicación del codigo

Método para Guardar el Usuario en la Base de Datos

- **URL, usuario y contraseña:** Detalles de conexión a la base de datos.
- **try-with-resources:** Asegura que la conexión y el PreparedStatement se cierren automáticamente después de su uso.
- **DriverManager.getConnection:** Establece la conexión a la base de datos.
- **con.prepareStatement:** Prepara la consulta SQL para insertar un nuevo usuario.
- **pst.setString:** Asigna los valores de nombre y email a los parámetros de la consulta.
- **pst.executeUpdate:** Ejecuta la consulta para insertar los datos en la base de datos.
- **Manejo de excepciones:** Captura y maneja cualquier SQLException que pueda ocurrir.

Explicación del codigo

Clase Main

Esta clase contiene el método main, que es el punto de entrada de la aplicación. Aquí se crea un objeto Usuario, se le asignan valores y se guarda en la base de datos.

```
public class Main {
    public static void main(String[] args) {
        Usuario usuario = new Usuario(); // Crear una instancia de Usuario
        usuario.setNombre("Ejemplo"); // Establecer el nombre
        usuario.setEmail("ejemplo@correo.com"); // Establecer el email
        usuario.guardarUsuario(); // Guardar el usuario en la base de datos
    }
}
```

Explicación del codigo

Crear una instancia de Usuario:

•**Usuario usuario = new Usuario();:** Crea un nuevo objeto Usuario utilizando el constructor vacío.

Establecer valores:

•u**suario.setNombre("Ejemplo");:** Asigna el nombre "Ejemplo" al objeto Usuario.

•**usuario.setEmail("ejemplo@correo.com");:** Asigna el email "ejemplo@correo.com" al objeto Usuario.

Guardar el usuario en la base de datos:

•**usuario.guardarUsuario();**: Llama al método guardarUsuario para almacenar la información del usuario en la base de datos.

Explicación del codigo

- El código define una clase Usuario con métodos para manejar sus propiedades y persistir la información en una base de datos MySQL.
- La clase Main crea un objeto Usuario, establece sus propiedades y lo guarda en la base de datos.

Practica

Implementa el ejercicio visto.

Conclusiones:

- Este enfoque muestra cómo utilizar JDBC en Java para interactuar con una base de datos, promoviendo buenas prácticas de programación como el uso de try-with-resources para gestionar conexiones y recursos.

Universidad Tecnológica del Perú

Cierre

¿Qué hemos aprendido hoy?

Bibliografía

- MORENO PÉREZ, J. “Programación orientada a objetos”. RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. “Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java”. Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú