CURSO HIBERNATE Y JPA

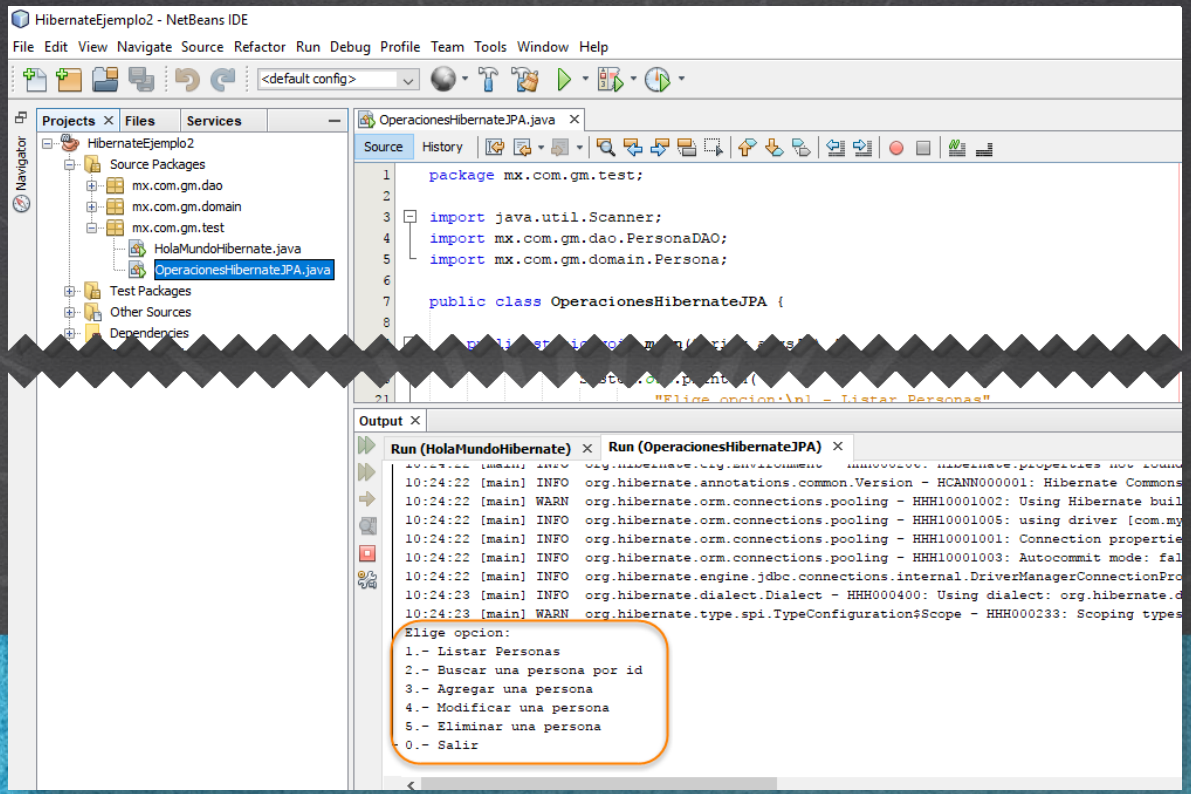
EJERCICIO

OPERACIONES BÁSICAS HIBERNATE Y JPA

Experiencia y Conocimiento para tu vida

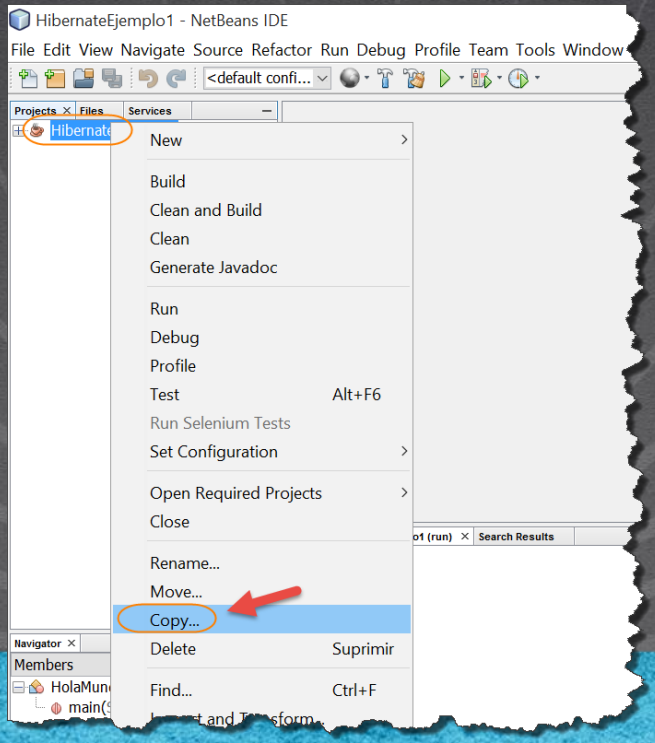
OBJETIVO DEL EJERCICIO

Crear un ejercicio para realizar las operaciones básicas utilizando Hibernate y JPA.
El resultado final es como sigue:



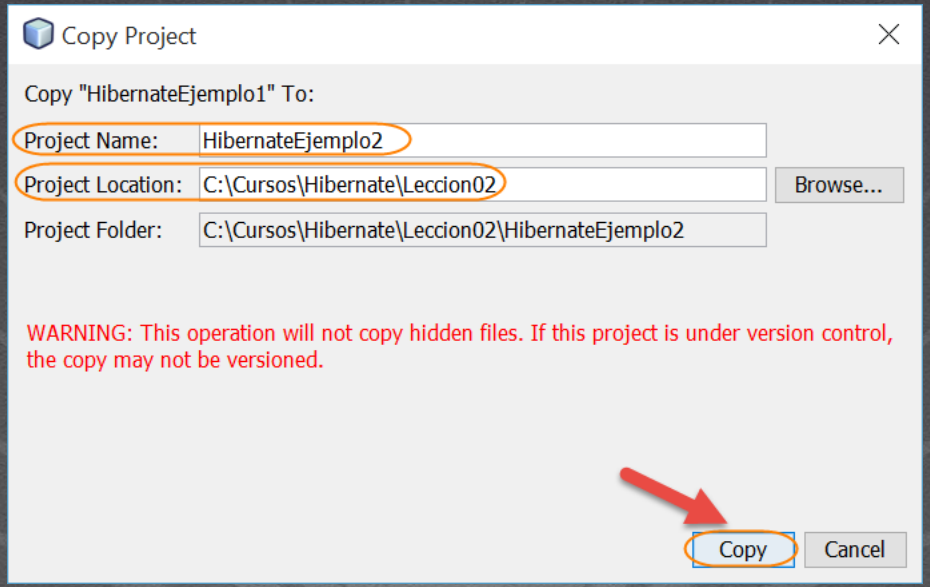
PASO 1. CREACIÓN DEL PROYECTO

Copiamos y pegamos el proyecto anterior:



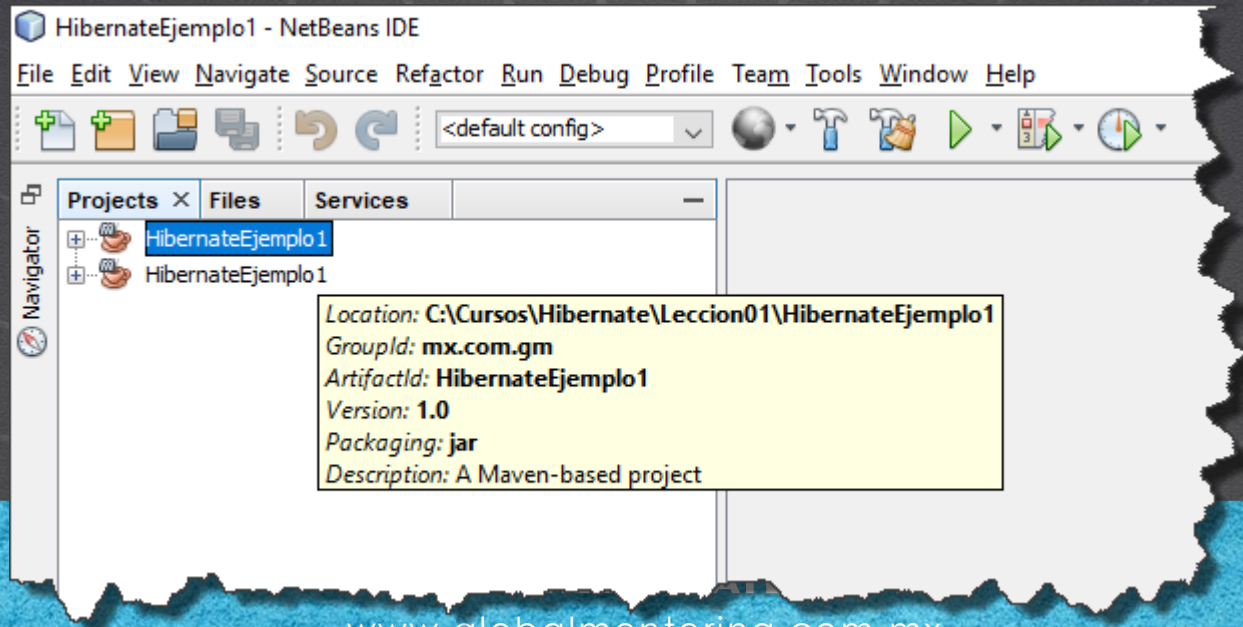
PASO 1. CREACIÓN DEL PROYECTO

Copiamos y pegamos el proyecto anterior:



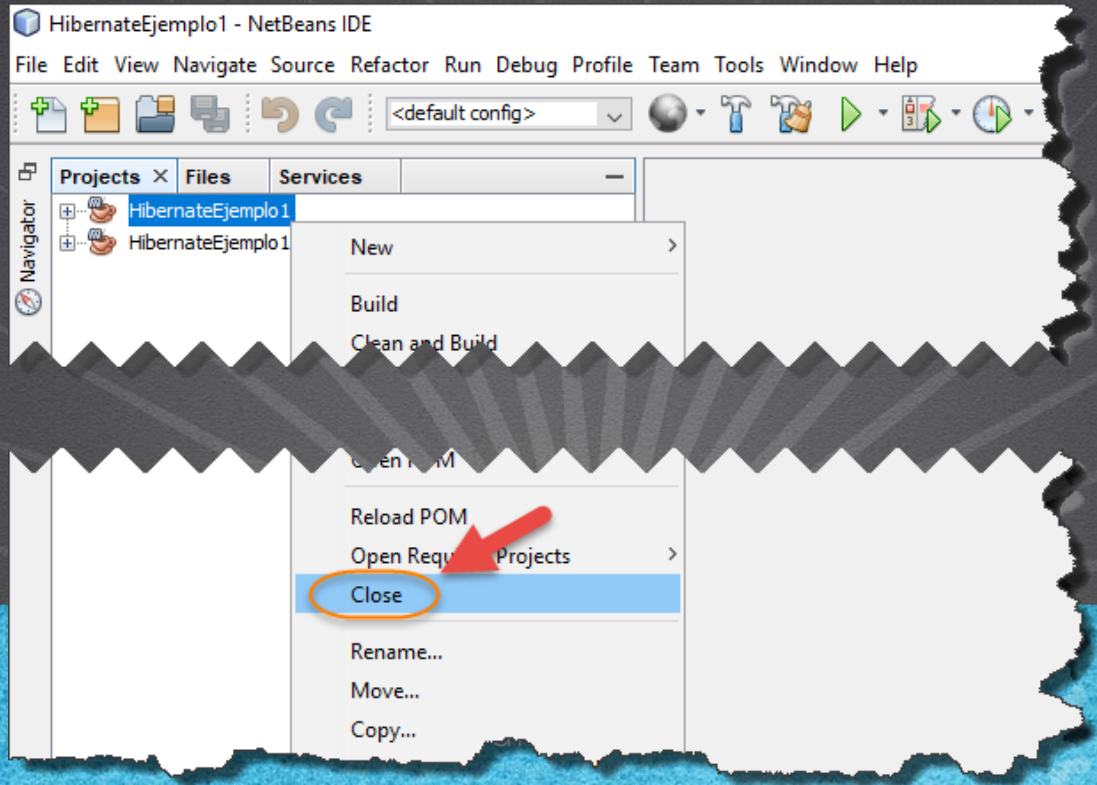
PASO 1. CREACIÓN DEL PROYECTO

Nos posicionamos sobre el proyecto para saber cual debemos cerrar y cual debemos renombrar. Cerramos el proyecto de la Leccion01:



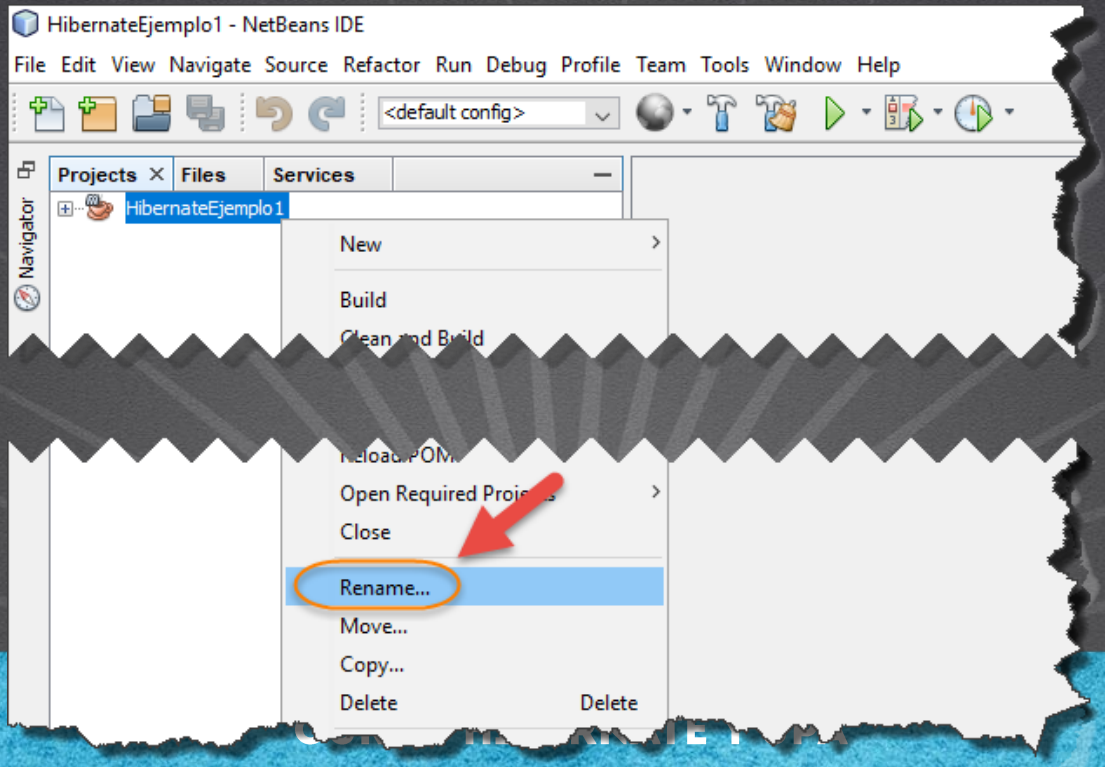
PASO 1. CREACIÓN DEL PROYECTO

Cerramos el proyecto de la Leccion01:



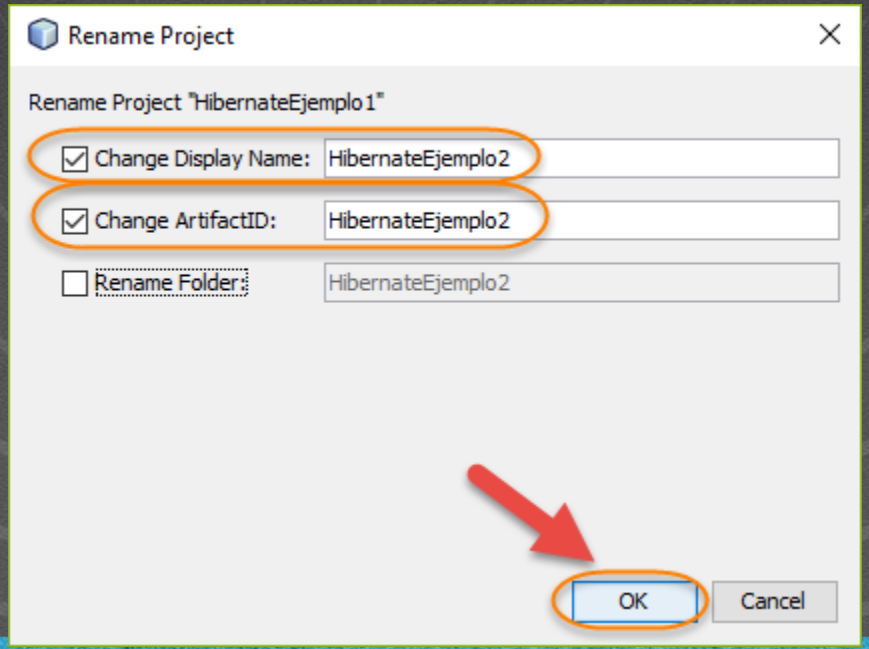
PASO 1. CREACIÓN DEL PROYECTO

Renombramos el proyecto de la Leccion02:



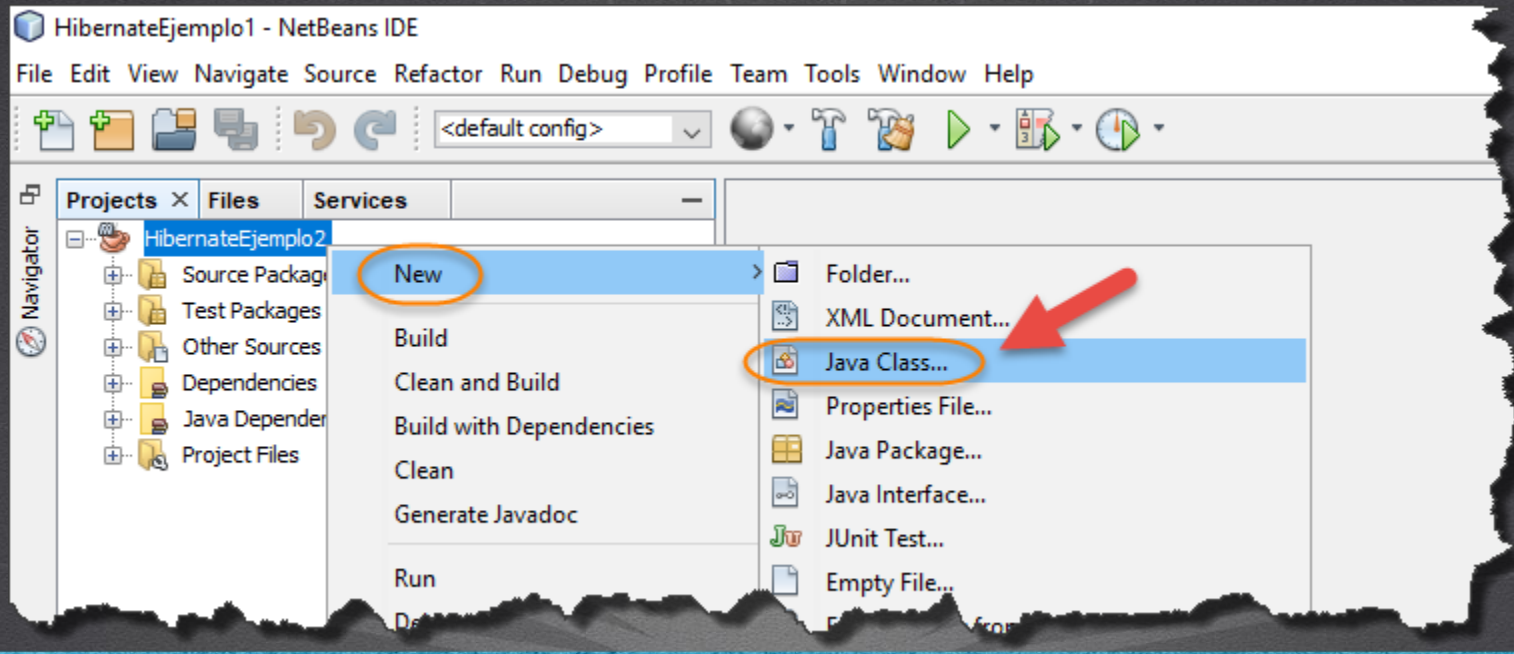
PASO 1. CREACIÓN DEL PROYECTO

Renombramos el proyecto de la Leccion02:



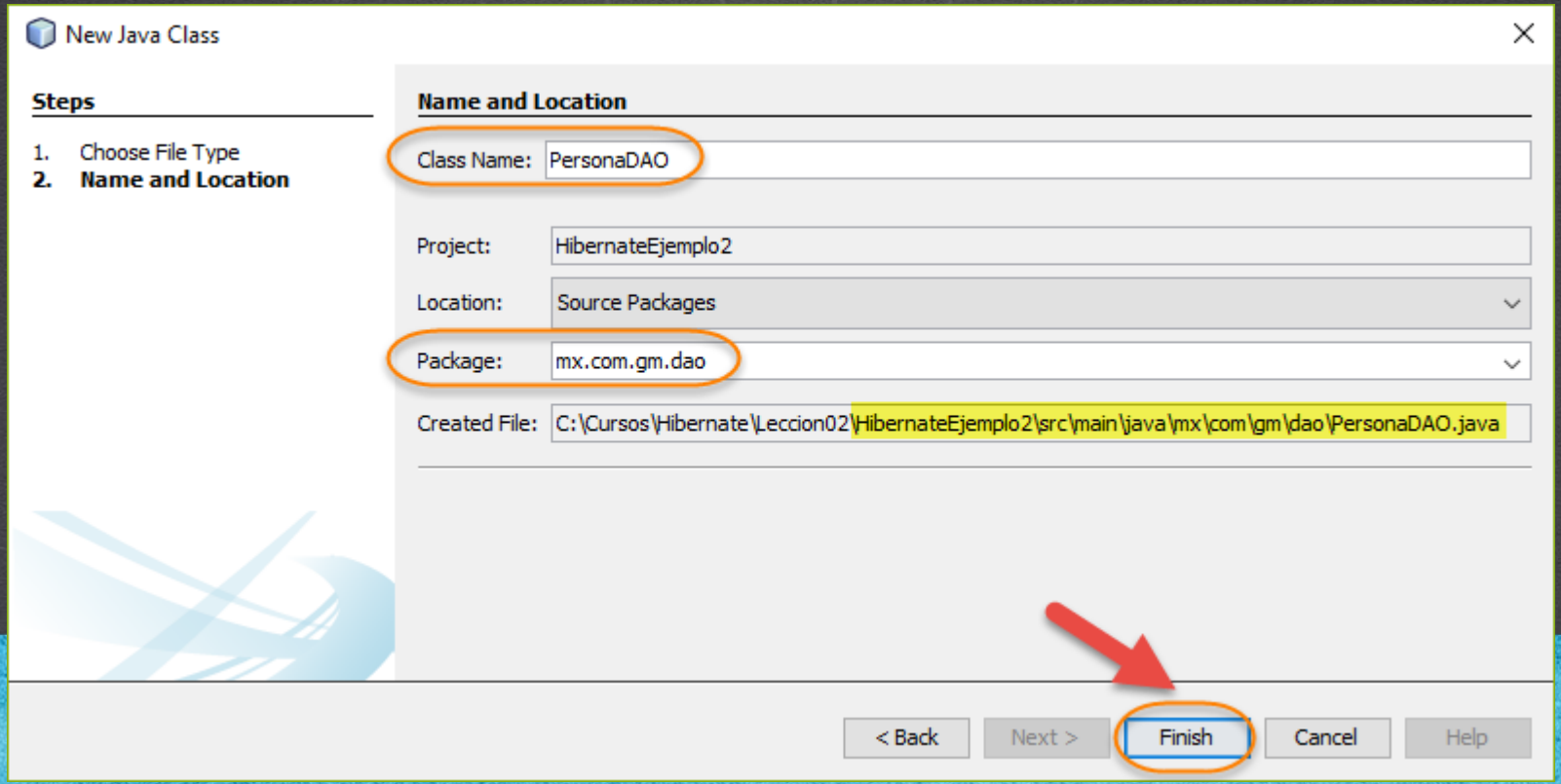
PASO 2. CREAR UNA CLASE

Creamos la clase PersonaDAO.java.



PASO 2. CREAR UNA CLASE

Creamos la clase PersonaDAO.java.



PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

```
package mx.com.gm.dao;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import mx.com.gm.domain.Persona;

public class PersonaDAO {

  protected EntityManager em;
  private EntityManagerFactory emf = null;

  public PersonaDAO() {
    // Usarmos el persistence unit
    emf = Persistence.createEntityManagerFactory("HibernatePU");
  }
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

```
public void listar() {
  // Consulta a ejecutar
  // No necesitamos crear una nueva transaccion
  String hql = "SELECT p FROM Persona p";
  em = getEntityManager();
  Query query = em.createQuery(hql);
  List<Persona> list = query.getResultList();
  for (Persona p : list) {
    System.out.println(p);
  }
}
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

```
public void insertar(Persona persona) {
  try {
    em = getEntityManager();
    // Iniciamos una transaccion
    em.getTransaction().begin();
    // Insertamos la nueva persona
    em.persist(persona);
    // Terminamos la transaccion
    em.getTransaction().commit();
  } catch (Exception ex) {
    System.out.println("Error al insetar objeto:" + ex.getMessage());
    // ex.printStackTrace();
  } finally {
    if (em != null) {
      em.close();
    }
  }
}
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

```
public void actualizar(Persona persona) {
  try {
    em = getEntityManager();
    // Iniciamos una transaccion
    em.getTransaction().begin();
    // Actualizamos al objeto persona
    em.merge(persona);
    // Terminamos la transaccion
    em.getTransaction().commit();
  } catch (Exception ex) {
    System.out.println("Error al actualizar objeto:" + ex.getMessage());
    // ex.printStackTrace();
  } finally {
    if (em != null) {
      em.close();
    }
  }
}
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

```
public void eliminar(Persona persona) {
  try {
    em = getEntityManager();
    // Iniciamos una transaccion
    em.getTransaction().begin();
    // Sincronizamos y eliminamos a la persona
    em.remove(em.merge(persona));
    // Terminamos la transaccion
    em.getTransaction().commit();
  } catch (Exception ex) {
    System.out.println("Error al eliminar objeto:" + ex.getMessage());
    // ex.printStackTrace();
  } finally {
    if (em != null) {
      em.close();
    }
  }
}
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo PersonaDAO.java:

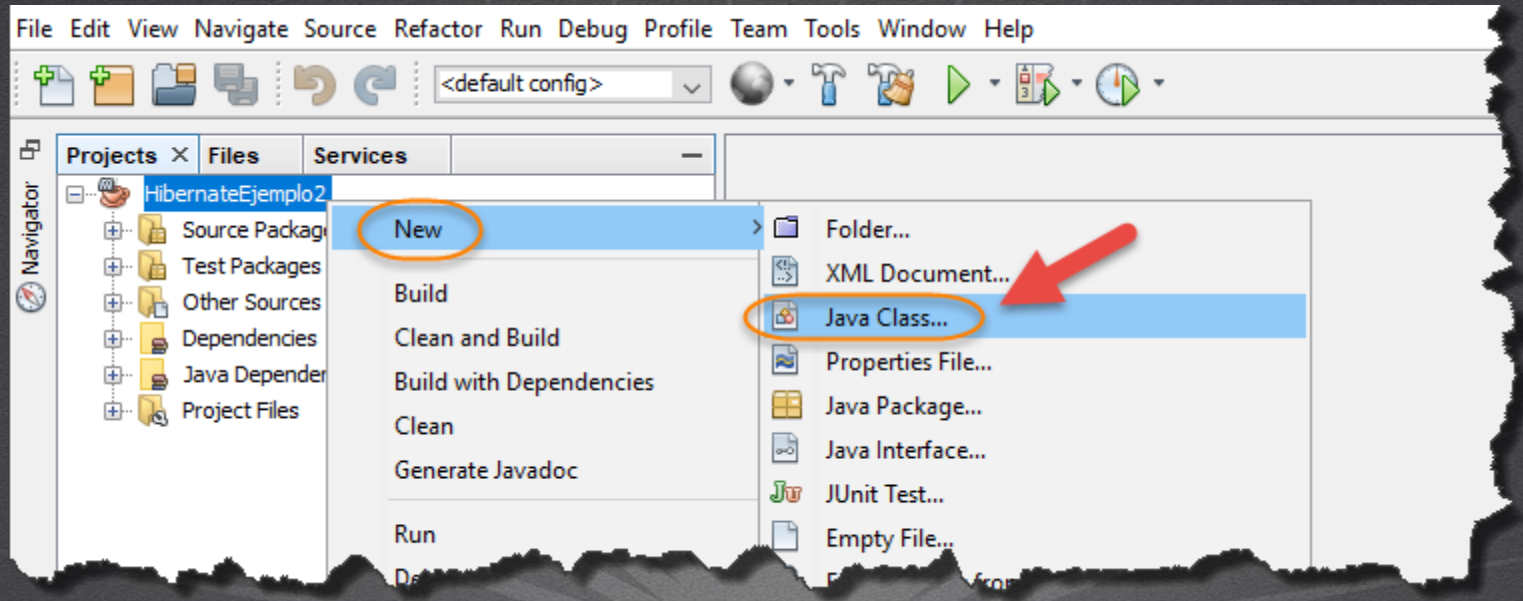
```
  public Persona buscarPorId(Persona p) {
    em = getEntityManager();
    return em.find(Persona.class, p.getIdPersona());
  }

  private EntityManager getEntityManager() {
    return emf.createEntityManager();
  }

}
```

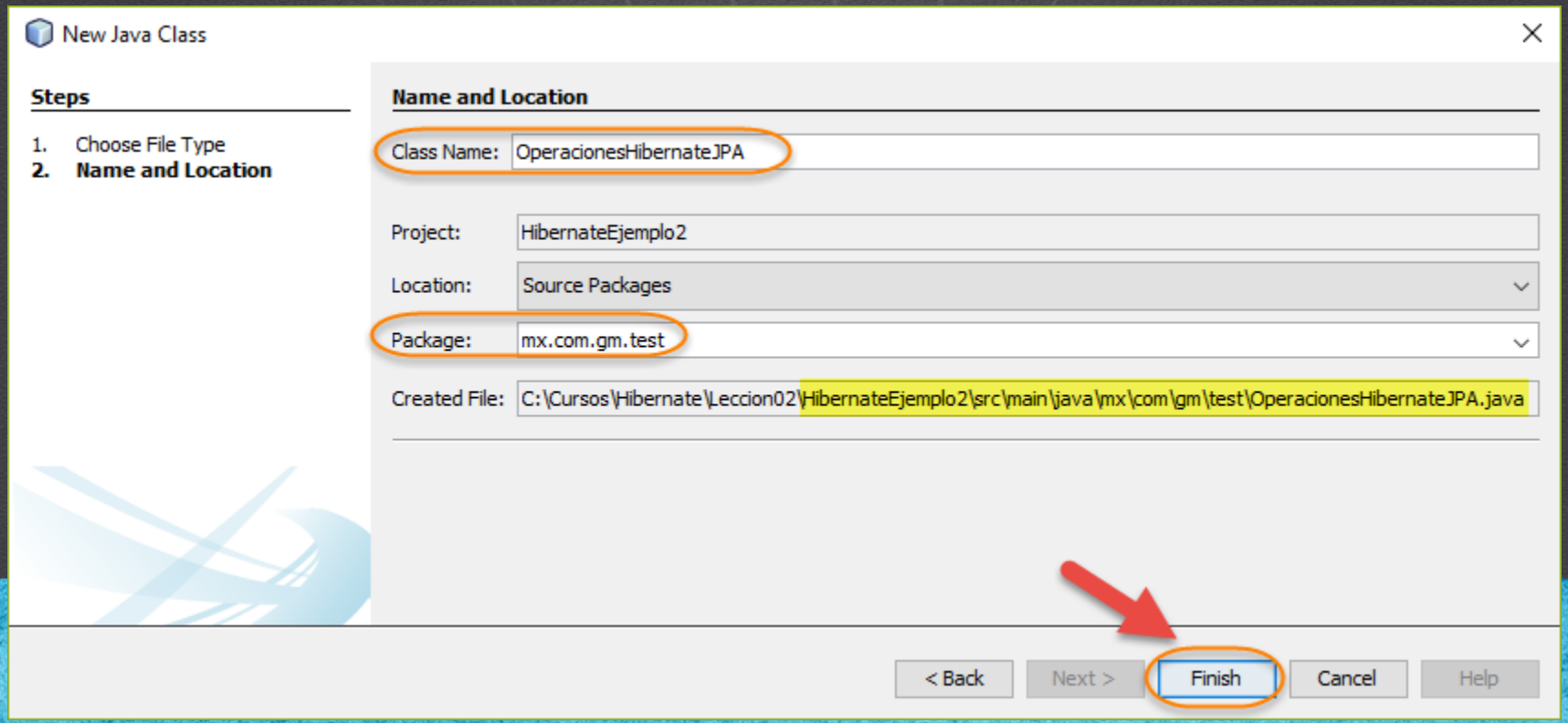
PASO 3. CREAR UNA CLASE

Creamos la clase OperacionesHibernateJPA.java:



PASO 3. CREAR UNA CLASE

Creamos la clase OperacionesHibernateJPA.java:



PASO 7. MODIFICAMOS EL CÓDIGO

Archivo OperacionesHibernateJPA.java:

```
package mx.com.gm.test;

import java.util.Scanner;
import mx.com.gm.dao.PersonaDAO;
import mx.com.gm.domain.Persona;

public class OperacionesHibernateJPA {

    public static void main(String args[]) {
        // Variables generales
        PersonaDAO pDao = new PersonaDAO();
        Persona p1 = null;
        int opcion = -1;
        Scanner scanner = new Scanner(System.in);
        String idStr, nombre, apellido;

        // Presiona 0 para salir
        while (opcion != 0) {
            try {
                System.out.println(
                        "Elige opcion:\n1.- Listar Personas"
                        + "\n2.- Buscar una persona por id "
                        + "\n3.- Agregar una persona"
                        + "\n4.- Modificar una persona\n"
                        + "5.- Eliminar una persona\n"
                        + "0.- Salir");
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo OperacionesHibernateJPA.java:

```
opcion = Integer.parseInt(scanner.nextLine());

// Ejemplo de switch case en Java
switch (opcion) {
    case 1:
        System.out.println("\n1.Listado**********");
        pDao.listar();
        break;
    case 2:
        System.out.println("\n2.Buscar por id**********");
        System.out.println("Introduce el id de la persona a buscar:");
        idStr = scanner.nextLine();
        p1 = new Persona();
        p1.setIdPersona(new Integer(idStr));
        p1 = pDao.buscarPorId(p1);
        System.out.println("Objeto encontado:" + p1);
        break;
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo OperacionesHibernateJPA.java:

```
case 3:
    System.out.println("\n3.Insertar**********");
    System.out.println("Introduce el nombre de la persona a agregar:");
    nombre = scanner.nextLine();
    System.out.println("Introduce el apellido de la persona a agregar:");
    apellido = scanner.nextLine();
    p1 = new Persona();
    p1.setNombre(nombre);
    p1.setApellido(apellido);
    // Guardamos el nuevo objeto
    pDao.insertar(p1);
    break;
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo OperacionesHibernateJPA.java:

```
case 4:
    System.out.println("\n4.Modificar***********");
    // Primero buscamos la persona a modificar
    System.out.println("Introduce el id de la persona a buscar:");
    idStr = scanner.nextLine();
    p1 = new Persona();
    p1.setIdPersona(new Integer(idStr));
    p1 = pDao.buscarPorId(p1);
    System.out.println("Introduce el nombre de la persona a modificar:");
    nombre = scanner.nextLine();
    System.out.println("Introduce el apellido de la persona a modificar:");
    apellido = scanner.nextLine();
    // Modificamos algun valor
    p1.setNombre(nombre);
    p1.setApellido(apellido);
    pDao.actualizar(p1);
    break;
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo OperacionesHibernateJPA.java:

```
                    case 5:
                        System.out.println("\n5. Eliminar***********");
                        // Primero buscamos la persona a eliminar
                        System.out.println("Introduce el id de la persona a eliminar:");
                        idStr = scanner.nextLine();
                        p1 = new Persona();
                        p1.setIdPersona(new Integer(idStr));
                        p1 = pDao.buscarPorId(p1);
                        // Eliminamos el objeto encontrado
                        pDao.eliminar(p1);
                        break;
                    case 0:
                        System.out.println("!Hasta pronto!");
                        break;
                    default:
                        System.out.println("Opcion no reconocida");
                        break;
                }
                System.out.println("\n");

            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }
}
```

PASO 8. MODIFICAMOS EL ARCHIVO XML

Modificamos el archivo persistence.xml para que ahora la unidad de persistencia se llame HibernatePU.

Veamos cómo queda nuestro archivo:

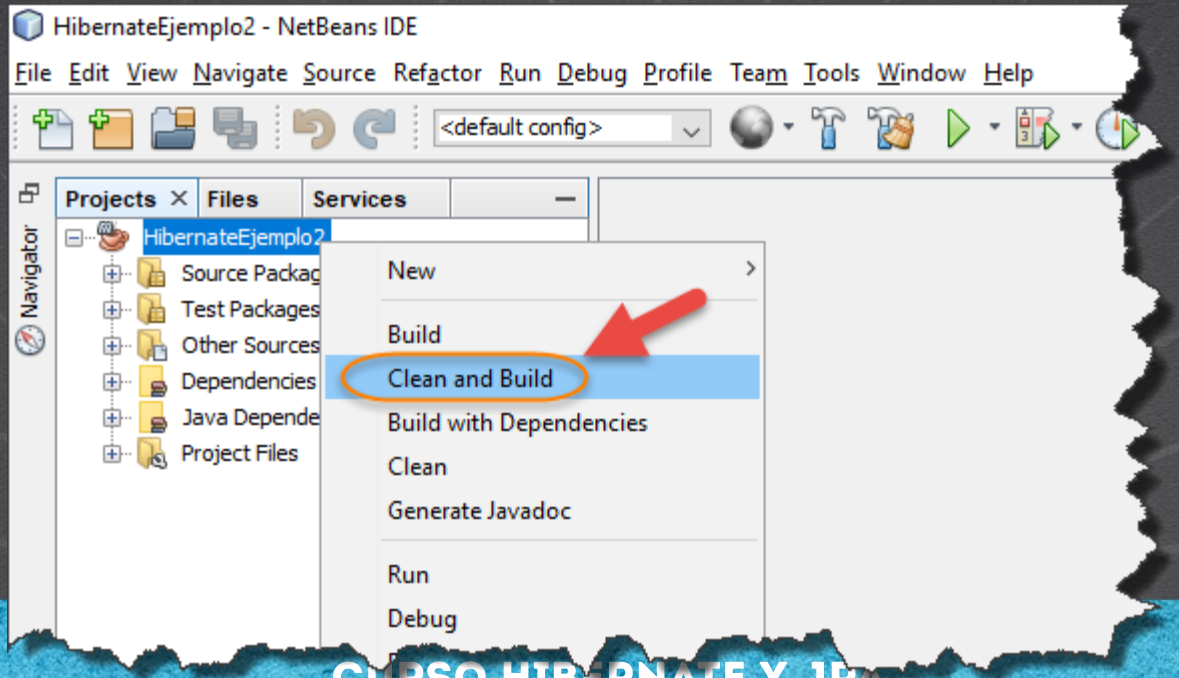
PASO 8. MODIFICAMOS EL CÓDIGO

Archivo persistence.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
             version="2.1">
    <persistence-unit name="HibernatePU" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <class>mx.com.gm.domain.Persona</class>
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/sga?useSSL=true"/>
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.password" value="admin"/>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
            <property name="hibernate.show_sql" value="true" />
        </properties>
    </persistence-unit>
</persistence>
```

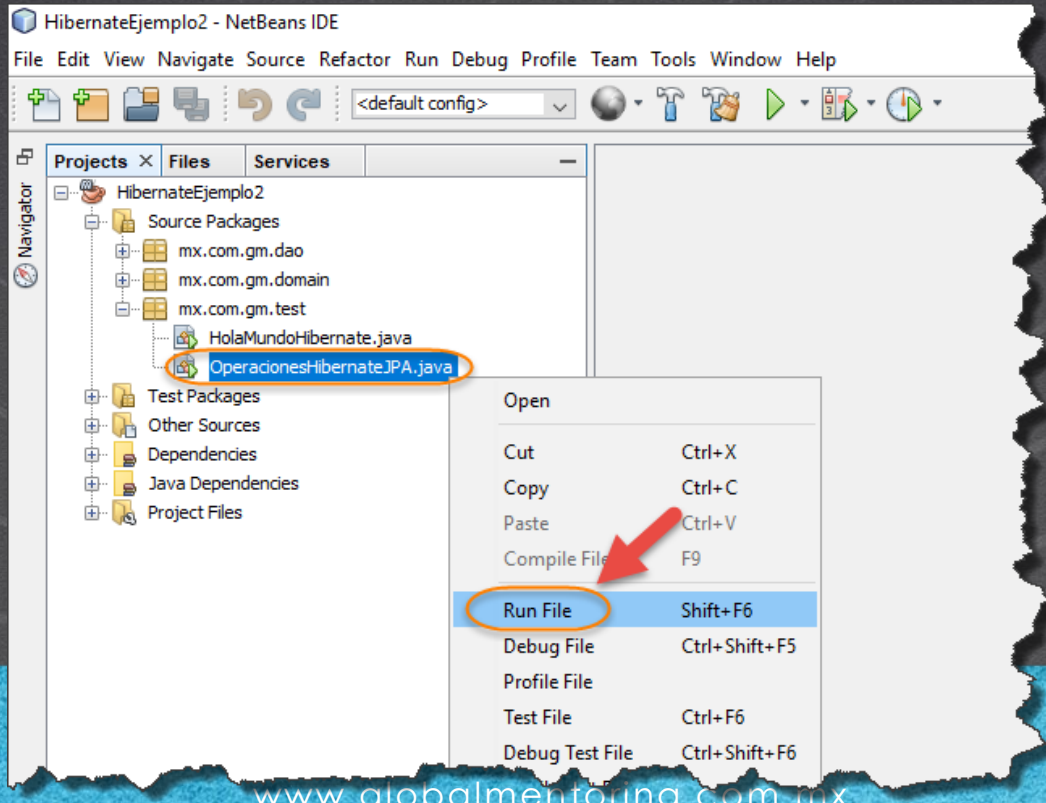
HACEMOS UN CLEAN & BUILD

Hacemos un Clean & Build del proyecto para que tengamos las ultimas versiones de nuestros archivos recién compilados:



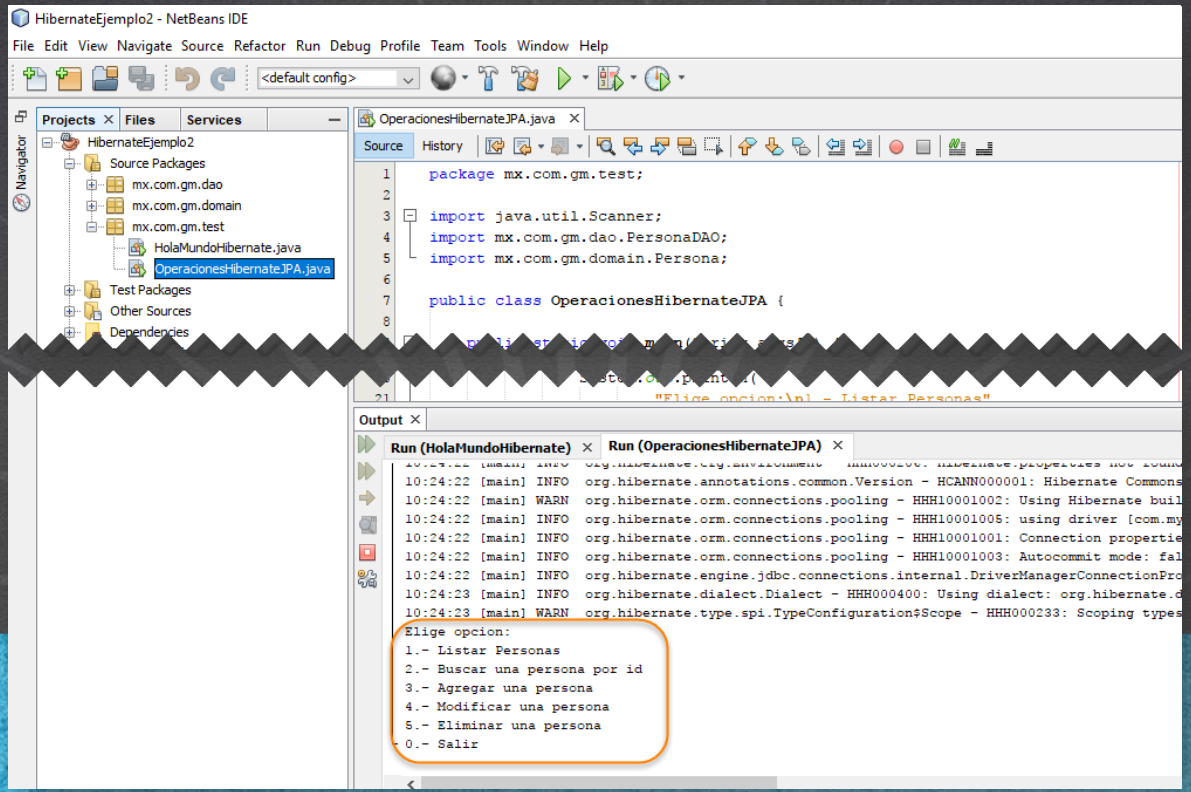
PASO 9. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



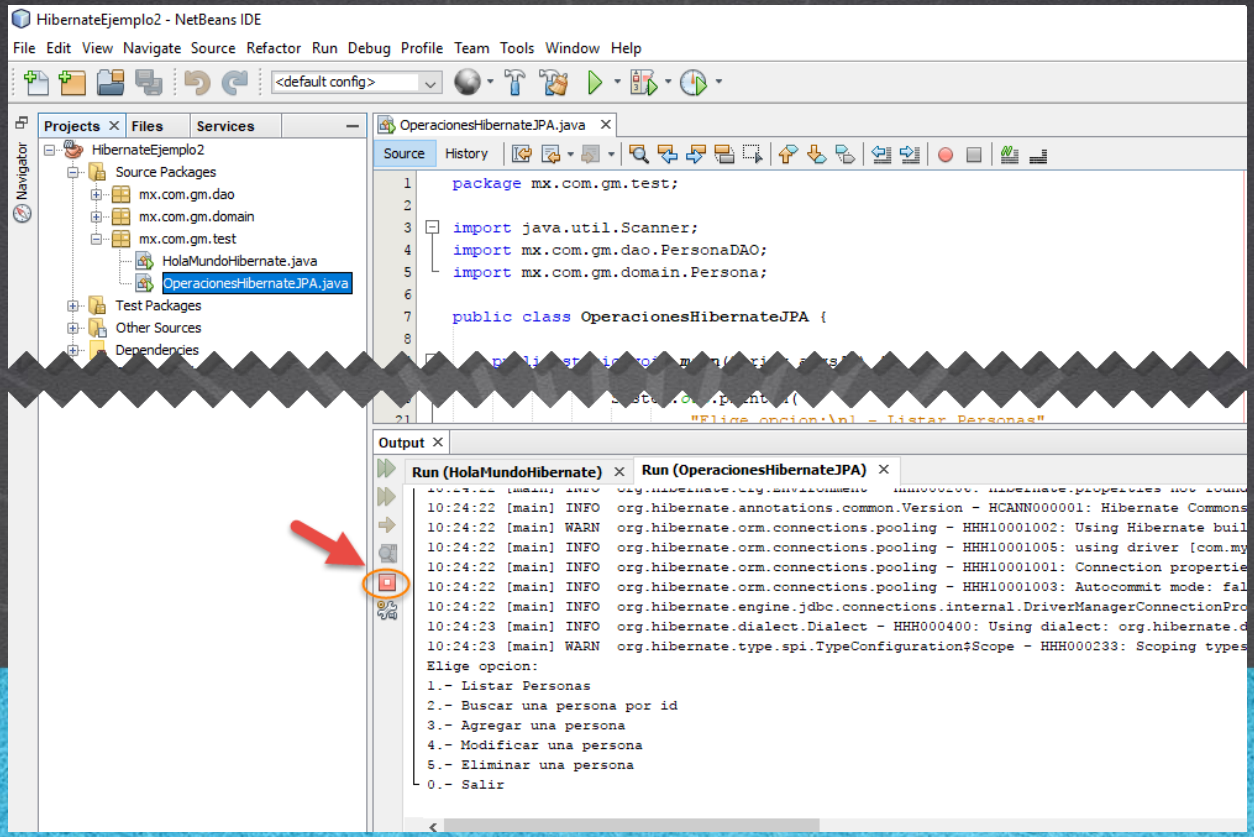
PASO 9. EJECUTAMOS EL PROYECTO (CONT)

Ejecutamos cada una de las acciones y verificamos la ejecución en la base de datos utilizada:



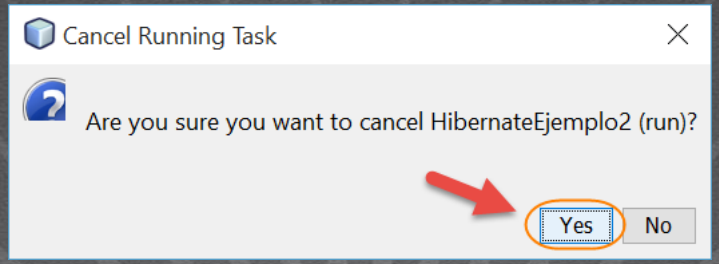
PASO 10. DETENEMOS EL PROGRAMA

En caso de ser necesario, detenemos la ejecución del programa:



PASO 10. DETENEMOS EL PROGRAMA

En caso de ser necesario, detenemos la ejecución del programa:



CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos puesto en práctica las operaciones básicas con Hibernate y JPA.

- Operaciones tales como listar, buscar por id, insertar, modificar y eliminar.

- Conforme vayamos avanzando iremos revisando a más detalle cada una de estas operaciones, así como el código utilizado, pero con este ejercicio ya tenemos una muy buena idea de la gran ayuda que nos brinda Hibernate y JPA para tanto persistir así como leer información en la base de datos.

CURSO ONLINE

HIBERNATE & JPA

Por: Ing. Ubaldo Acosta

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx