

ACTIVIDAD

Diseño de programas para la gestión de bases de datos relacionales y persistencia de objetos

Desarrollo de Aplicaciones Web /
Desarrollo de Aplicaciones Multiplataforma

Programación



Actividad

Diseño de programas para la gestión de bases de datos relacionales y persistencia de objetos

Objetivos



- Gestionar información almacenada en bases de datos relacionales.
- Programar conexiones a bases de datos.
- Escribir código para almacenar información en bases de datos, así como editarla y consultarla.
- Programar aplicaciones que almacenen objetos en bases de datos objeto-relacionales.
- Realizar programas para recuperar, actualizar y eliminar objetos de las bases de datos objeto-relacionales.

¿Cómo lo hago?

1. Rellena los datos que se piden en la tabla “Antes de empezar”.
2. Haz uso de fuentes comunes como Arial, Calibri, Times New Roman etc.
3. Utiliza el color negro para desarrollar tus respuestas y usa otros colores para destacar contenidos o palabras que creas necesario resaltar.
4. Recuerda entregar la actividad en formato PDF a no ser que el profesor o profesora indique lo contrario.
5. Recuerda nombrar el archivo siguiendo estas indicaciones:
 - Ciclo_Módulo o crédito_Tema_ACT_número actividad_Nombre y apellido
 - Ejemplo: AF_M01_T01_ACT_01_Maria Garcia

Antes de empezar...

Nombre	Vadym
Apellidos	Volokhov
Módulo/Crédito	M03
UF (solo ciclos LOE)	UF6
Título de la actividad	ACT_07





Se debe entregar un zip que contenga todos los archivos .java que has creado y el código SQL para crear la base de datos. Para poder aprobar un ejercicio, éste debe poder ejecutarse sin errores. Crea los archivos .java dentro de una carpeta de nombre actividad07

1- Crea dentro de un package de nombre “**actividad07.libreria**” un programa para gestionar información sobre libros almacenando la información en MySQL.

1. Crea la clase **LibreriaMain** que muestra al usuario un menú con 6 opciones:
 - a. Restablecer la base de datos: esta opción debe:
 - i. Borrar la base de datos si ya existe
 - ii. Crear la base de datos
 - iii. Crear una tabla libro para almacenar: título, autor, país, paginas, género y un identificador autoincremental.
 - iv. Crear una tabla autor para almacenar: nombre y los apellidos del autor y un identificador autoincremental
 - v. Añadir dos autores
 - vi. Para cada uno de los dos autores anteriores, añadir un libro suyo.
 - b. Mostrar autores: esta opción debe de mostrar un listado de los autores
 - c. Mostrar libros: esta opción debe mostrar un listado de los libros
 - d. Modificar un autor: esta opción debe
 - i. Mostrar un listado de los autores
 - ii. Pedir qué autor se quiere modificar según su identificador
 - iii. Pedir qué dato se quiere modificar del autor
 - iv. Actualizar en la base de datos la información del autor.
 - e. Eliminar un libro: esta opción debe:
 - i. Mostrar un listado de los libros
 - ii. Pedir qué libro se quiere eliminar según su identificador
 - iii. Borrar el libro de la base de datos

A continuación se muestra un ejemplo de ejecución:

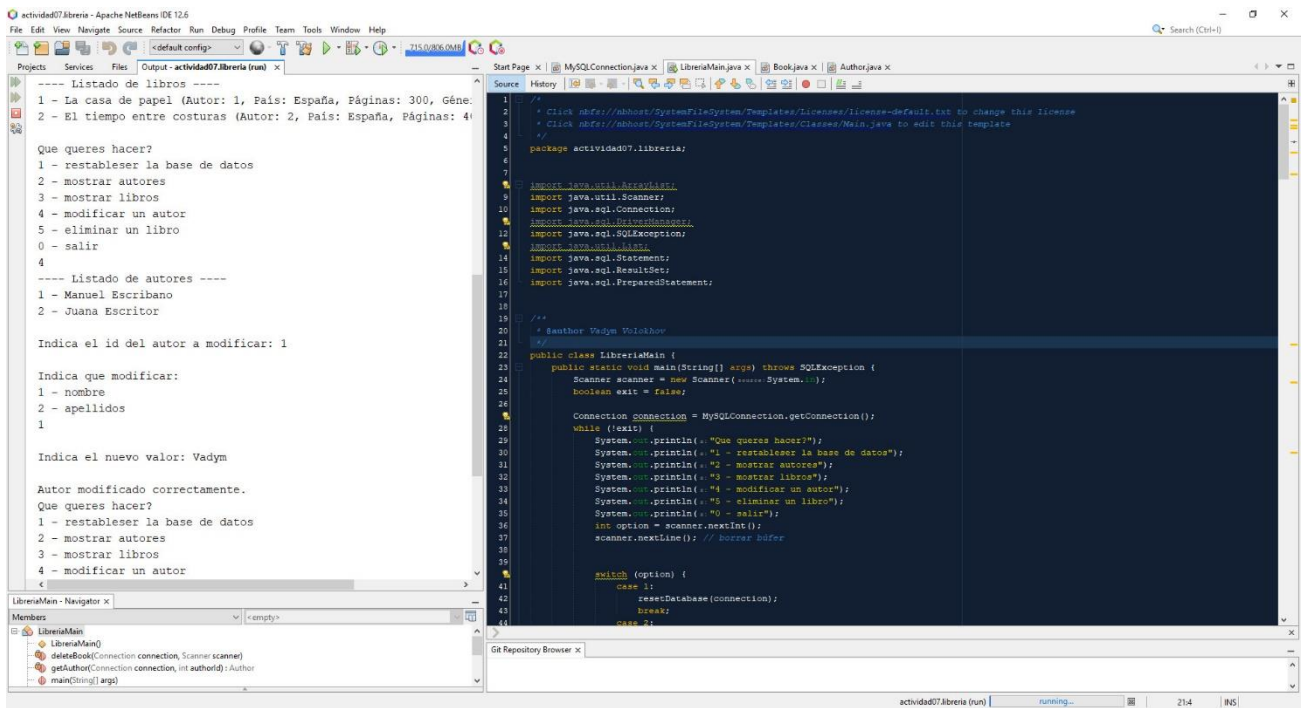
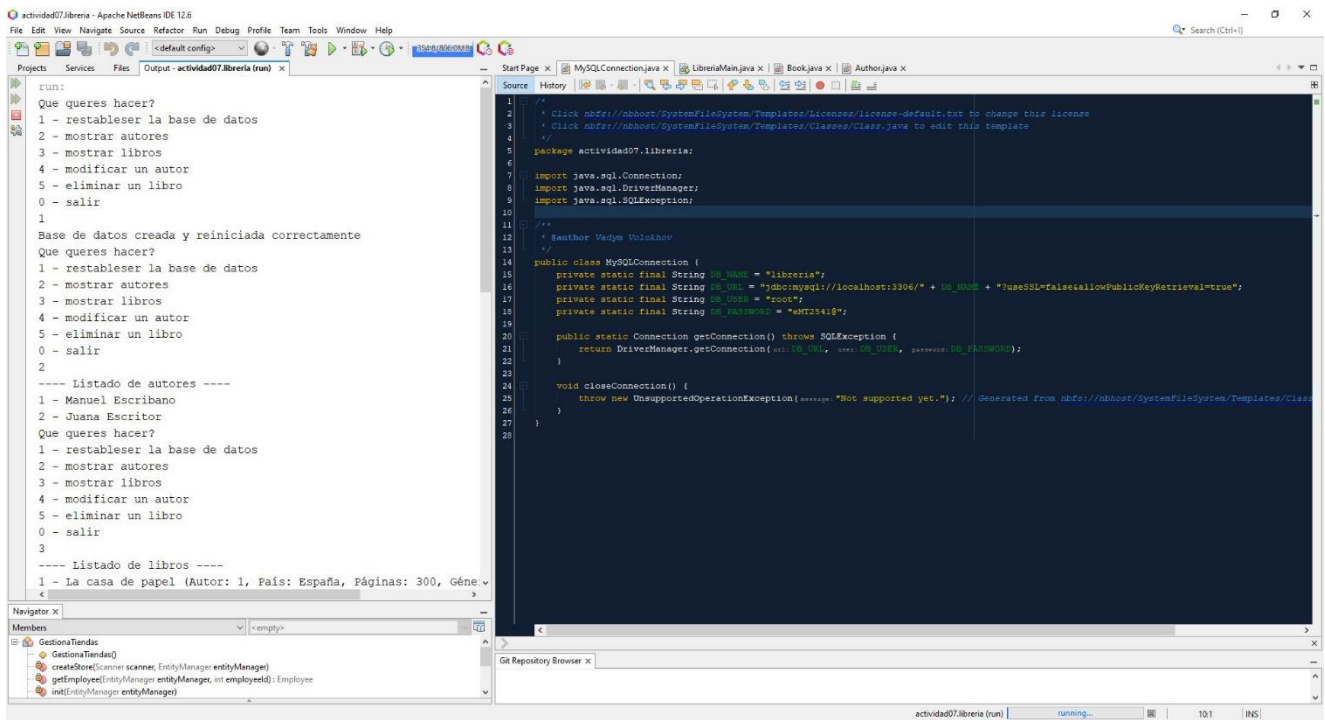


Qué quieres hacer?
1-restablecer la base de datos
2-mostrar autores
3-mostrar libros
4-modificar un autor.
5-eliminar un libro.
0-Salir

4
----LISTADO DE AUTORES---
1 - Manuel Escribano
2 - Juana Escritor

Indica el id del autor a modificar
1
Indica qué modificar:
1-nombre
2-apellidos
2
Indica el nuevo valor
Escribnatitos
Qué quieres hacer?
1-restablecer la base de datos
2-mostrar autores
3-mostrar libros
4-modificar un autor.
5-eliminar un libro.
0-Salir

2
----LISTADO DE AUTORES---
1 - Manuel Escribnatitos
2 - Juana Escritor



actividad07/libreria - Apache NetBeans IDE 12.6

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Output - actividad07.libreria (run) x

2 - Juana Escritor

Indica el id del autor a modificar: 1

Indica que modificar:

1 - nombre

2 - apellidos

1

Indica el nuevo valor: Vadym

Autor modificado correctamente.

Que quieres hacer?

1 - restablecer la base de datos

2 - mostrar autores

3 - mostrar libros

4 - modificar un autor

5 - eliminar un libro

0 - salir

2

---- Listado de autores ----

1 - Vadym Escribano

2 - Juana Escritor

Que quieres hacer?

1 - restablecer la base de datos

2 - mostrar autores

3 - mostrar libros

4 - modificar un autor

5 - eliminar un libro

0 - salir

LibreriaMain - Navigator x

Members

LibreriaMain

LibreriaMain()

deleteBook(Connection connection, Scanner scanner)

getAuthor(Connection connection, int authorId): Author

main(String[] args)

Source

```

1  //
2  // Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  // Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4  //
5  package actividad07.libreria;
6
7
8
9  import java.util.Scanner;
10 import java.sql.Connection;
11 import java.sql.DriverManager;
12 import java.sql.SQLException;
13 import java.sql.Statement;
14 import java.sql.ResultSet;
15 import java.sql.PreparedStatement;
16
17
18
19
20 //author Vadym Voichkov
21
22 public class LibreriaMain {
23     public static void main(String[] args) throws SQLException {
24         Scanner scanner = new Scanner(System.in);
25         boolean exit = false;
26
27         Connection connection = MySqlConnection.getConnection();
28         while (!exit) {
29             System.out.println("Que quieres hacer?");
30             System.out.println("1 - restablecer la base de datos");
31             System.out.println("2 - mostrar autores");
32             System.out.println("3 - mostrar libros");
33             System.out.println("4 - modificar un autor");
34             System.out.println("5 - eliminar un libro");
35             System.out.println("0 - salir");
36             int option = scanner.nextInt();
37             scanner.nextLine(); // border buffer
38
39             switch (option) {
40                 case 1:
41                     resetDatabase(connection);
42                     break;
43                 case 2:
44

```

Git Repository Browser x

actividad07.libreria (run) | running... | 214 | INS

127.0.0.1:81/openserver/phpmyadmin/index.php?route=/sql&server=1&db=actividad07&table=autor&pos=0

Tools & Settings - P... Plek Onyx 17.5.3 Kubesoluciones Sit... Bot de WhatsApp... Vodafone Xp... 3asoposkame ko... Odladenns d... KS Loach | Contact... Anna Zadorozhaya Panel de Control | d... G8 CRM - Task Ma... Nike New Arrivals...

phpMyAdmin

Recent Favorites

New

actividad05

actividad07

New

autor

libro

actividad4

drupal7

drupal6

emerya

embo

equipos

information_schema

libreria

mysql

neo

pedidos

performance_schema

plugin

sys

transparencia

Server: 127.0.0.1:3306 Database: actividad07 Table: autor

Showing rows 0 - 1 (2 total. Query took 0.0028 seconds)

SELECT * FROM `autor`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 50 | Filter rows: Search this table | Sort by key: None

Options

id	nombre	apellidos
1	Vadym	Escribano
2	Juana	Escritor

Check all | With selected

Show all | Number of rows: 50 | Filter rows: Search this table | Sort by key: None

Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Console

2- Crea dentro de un package de nombre “**actividad07.libreria**” un programa para gestionar información en **objectDB** sobre tiendas y sus empleados.

En la base de datos se debe almacenar la siguiente información

1. Sobre un Empleado:
 - a. nombre : nombre del empleado
 - b. apellido: apellido del empleado
2. Sobre una tienda
 - a. id: campo autogenerado
 - b. direccion : la dirección de la tienda
 - c. ventas : el número de ventas que ha efectuado una tienda en el transcurso de la semana.
 - d. empleados : conjunto de empleados que tiene una tienda

Crea la clase **GestionaTiendas** con el método *main* para que:

1. Inicialmente cree tres tiendas y tres empleados y los almacene en la base de datos (los datos estarán preestablecidos por vosotros).
2. Muestre un menú con las siguientes opciones programada cada una en una función aparte. El programa solo finaliza al seleccionar la opción de salir:
 1. Mostrar los empleados
 2. Mostrar las tiendas
 3. Mostrar tiendas ordenadas por ventas
 4. Editar un empleado
 5. Crear una nueva tienda
 0. Salir

A continuación, se muestra un ejemplo de ejecución del programa:

Menú inicial:

Introduzca la operación a realizar del siguiente menú de opciones:

- 1 - Muestra los empleados
- 2 - Muestra las tiendas
- 3 - Mostrar tiendas ordenadas por ventas
- 4 - Modificar un empleado.
- 5 - Añade una tienda.
- 0 - Salir

Mostrar los empleados:

```
1
index:[0] Autor: [ id=4, nombre=Black,apellido=Orchid]
index:[1] Autor: [ id=5, nombre=Volvo,apellido=Soul]
index:[2] Autor: [ id=6, nombre=Blanca,apellido=Street]
```


Mostrar las tiendas:

```
2
Tienda [ id=1, direccion=Calle Elm 84, ventas=100,
empleados=[Autor: [ id=4, nombre=Black,apellido=Orchid]]]
Tienda [ id=2, direccion=Calle Malasana 32, ventas=90,
empleados=[Autor: [ id=5, nombre=Voldo,apellido=Soul]]]
Tienda [ id=3, direccion=Calle cloverfield 10, ventas=110,
empleados=[Autor: [ id=5, nombre=Voldo,apellido=Soul], Autor: [ id=6, nombre=Blanca,apellido=Street]]]
```

Mostrar tiendas por ventas descendiente

```
3
Ordenar segun ventas Ascendiente o Descendiente?
1- Ascendiente
2-Descendiente
2
Tienda [ id=3, direccion=Calle cloverfield 10, ventas=110,
empleados=[Autor: [ id=5, nombre=Voldo,apellido=Soul], Autor: [ id=6, nombre=Blanca,apellido=Street]]]
Tienda [ id=1, direccion=Calle Elm 84, ventas=100,
empleados=[Autor: [ id=4, nombre=Black,apellido=Orchid]]]
Tienda [ id=2, direccion=Calle Malasana 32, ventas=90,
empleados=[Autor: [ id=5, nombre=Voldo,apellido=Soul]]]
```

Editar un empleado:

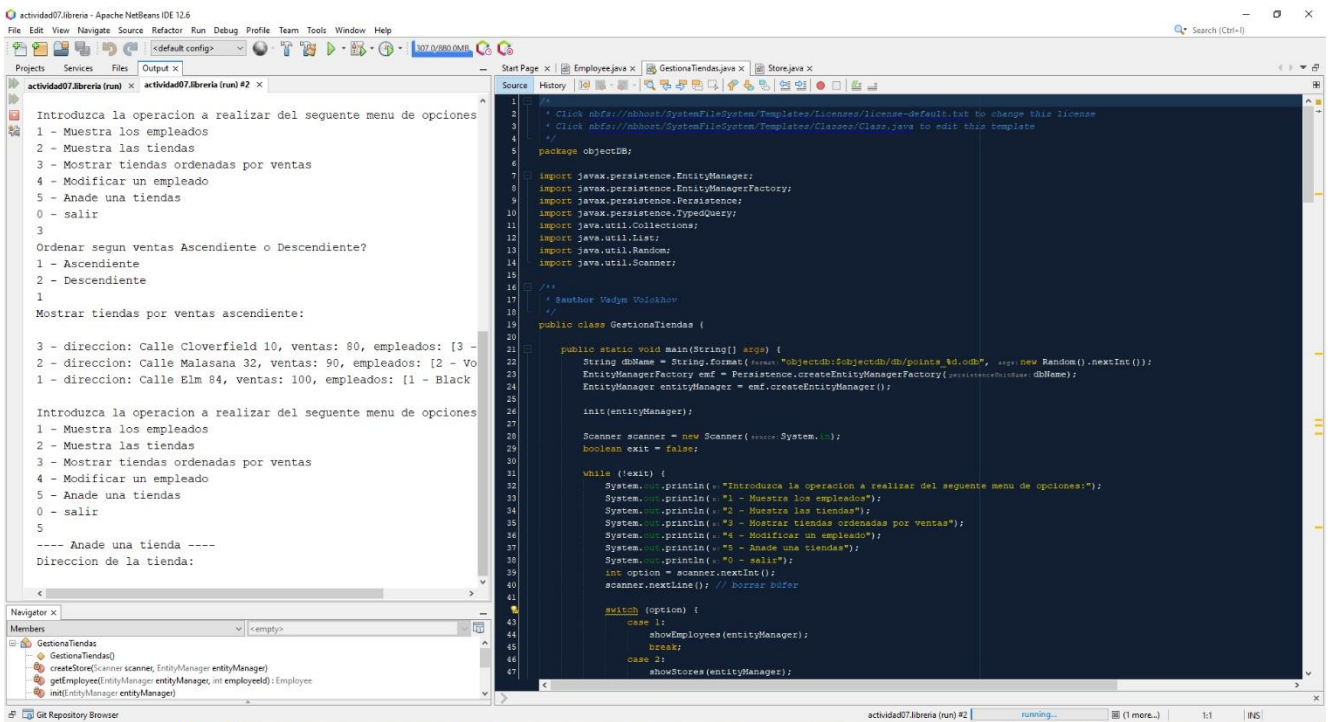
```
4
index:[0] Autor: [ id=4, nombre=Black,apellido=Orchid]
index:[1] Autor: [ id=5, nombre=Voldo,apellido=Soul]
index:[2] Autor: [ id=6, nombre=Blanca,apellido=Street]
Id del empleado a modificar
5
El empleado seleccionado es: Autor: [ id=5, nombre=Voldo,apellido=Soul]
Indica el atributo a modificar:
1 - Nombre
2 - Apellido
0 - Ninguno
1
Indica el nuevo nombre:
Polo
```

Crear una tienda con el empleado Polo y Blanca:

5
 Direccion de la tienda
 Hyrule Street
 ventas
 999
 index:[0] Autor: [id=4, nombre=Black,apellido=Orchid]
 index:[1] Autor: [id=5, nombre=Polo,apellido=Soul]
 index:[2] Autor: [id=6, nombre=Blanca,apellido=Street]
 Index del empleado que quieras añadir a la tienda:
 1
 Empleado añadido correctametne
 Quieres añadir un nuevo empleado:
 1- SI
 0-No
 1
 Index del empleado que quieras añadir a la tienda:
 2
 Empleado añadido correctametne
 Quieres añadir un nuevo empleado:
 1- SI
 0-No
 0

The screenshot shows an IDE with the following components:

- Console (Run):** Displays the execution of the application. It shows a menu of options (1-5) and the user's input. The application lists employees and stores, and the user adds a new employee.
- Source:** Shows the source code of the `GestionaTiendas.java` file. The code includes imports for `EntityManager`, `EntityManagerFactory`, `Persistence`, `TypedQuery`, `Collection`, `List`, `Random`, and `Scanner`. It defines a `GestionaTiendas` class with a `main` method that handles the menu options.
- Navigator:** Shows the project structure with the `GestionaTiendas` package and its classes.



actividad07.libreria - Apache NetBeans IDE 12.6

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Output x

actividad07.libreria (run) actividad07.libreria (run) #2 x

Introduzca la operacion a realizar del siguiente menu de opciones

- 1 - Muestra los empleados
- 2 - Muestra las tiendas
- 3 - Mostrar tiendas ordenadas por ventas
- 4 - Modificar un empleado
- 5 - Anade una tienda
- 0 - salir

3

Ordenar segun ventas Ascendiente o Descendiente?

- 1 - Ascendiente
- 2 - Descendiente

1

Mostrar tiendas por ventas ascendiente:

- 3 - direccion: Calle Cloverfield 10, ventas: 80, empleados: [3 -
- 2 - direccion: Calle Malasana 32, ventas: 90, empleados: [2 - Vo
- 1 - direccion: Calle Elm 84, ventas: 100, empleados: [1 - Black

Introduzca la operacion a realizar del siguiente menu de opciones

- 1 - Muestra los empleados
- 2 - Muestra las tiendas
- 3 - Mostrar tiendas ordenadas por ventas
- 4 - Modificar un empleado
- 5 - Anade una tienda
- 0 - salir

5

---- Anade una tienda ----

Direccion de la tienda:

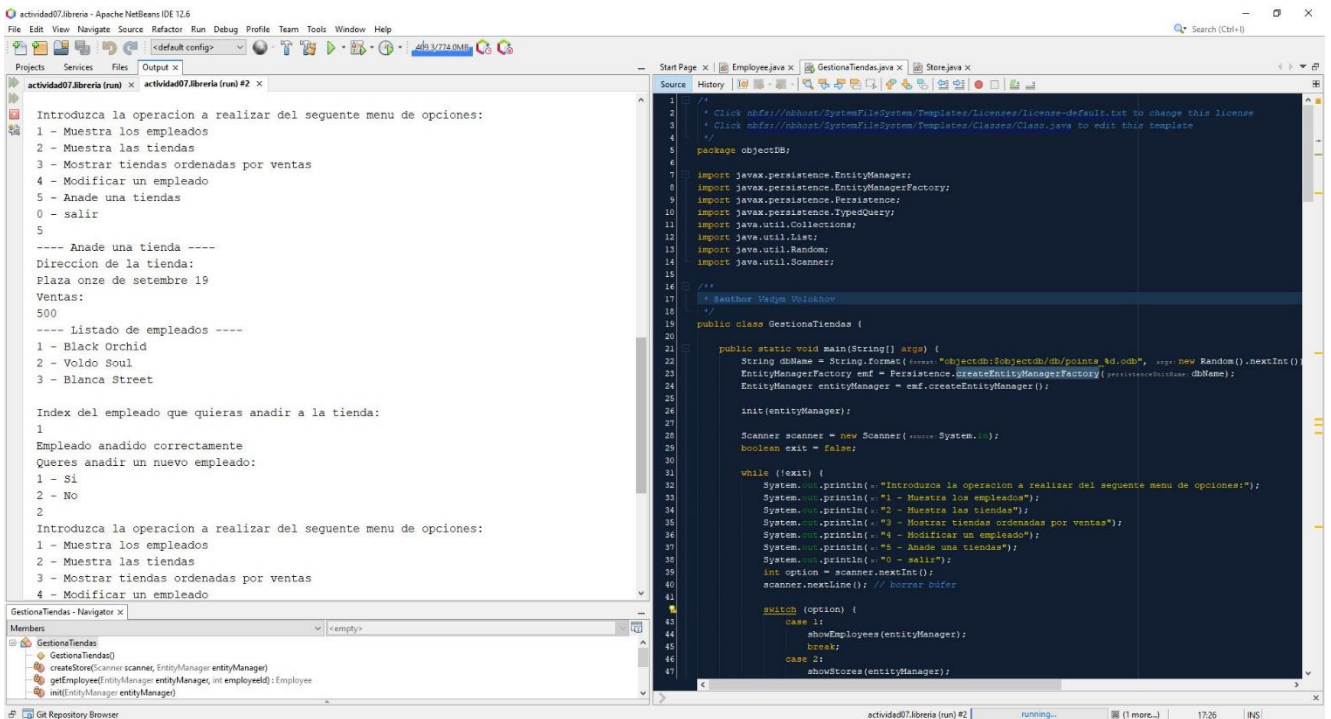
Members

- GestionaTiendas
- GestionaTiendas()
- createStore(Scanner scanner, EntityManager entityManager)
- getEmployee(EntityManager entityManager, int employeeId): Employee
- init(EntityManager entityManager)

Source

```
1 //
2 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 //
5 package objectDB;
6
7 import javax.persistence.EntityManager;
8 import javax.persistence.EntityManagerFactory;
9 import javax.persistence.Persistence;
10 import javax.persistence.TypedQuery;
11 import java.util.Collections;
12 import java.util.List;
13 import java.util.Random;
14 import java.util.Scanner;
15
16 /**
17  * @author Vadym Volokhov
18  */
19 public class GestionaTiendas {
20
21     public static void main(String[] args) {
22         String dbName = String.format("objectdb:%s/objectdb/db/points_1d.ojb", args[0] + new Random().nextInt());
23         EntityManagerFactory emf = Persistence.createEntityManagerFactory("objectdb:" + dbName);
24         EntityManager entityManager = emf.createEntityManager();
25
26         init(entityManager);
27
28         Scanner scanner = new Scanner(System.in);
29         boolean exit = false;
30
31         while (!exit) {
32             System.out.println("Introduzca la operacion a realizar del siguiente menu de opciones:");
33             System.out.println("1 - Muestra los empleados");
34             System.out.println("2 - Muestra las tiendas");
35             System.out.println("3 - Mostrar tiendas ordenadas por ventas");
36             System.out.println("4 - Modificar un empleado");
37             System.out.println("5 - Anade una tienda");
38             System.out.println("0 - salir");
39             int option = scanner.nextInt();
40             scanner.nextLine(); // borra buffer
41
42             switch (option) {
43                 case 1:
44                     showEmployees(entityManager);
45                     break;
46                 case 2:
47                     showStores(entityManager);
48             }
49         }
50     }
51 }
```

actividad07.libreria (run) #2 running... (1 more...) 1:1 [NS]



actividad07.libreria - Apache NetBeans IDE 12.6

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Output x

actividad07.libreria (run) actividad07.libreria (run) #2 x

Introduzca la operacion a realizar del siguiente menu de opciones:

- 1 - Muestra los empleados
- 2 - Muestra las tiendas
- 3 - Mostrar tiendas ordenadas por ventas
- 4 - Modificar un empleado
- 5 - Anade una tienda
- 0 - salir

5

---- Anade una tienda ----

Direccion de la tienda:

Plaza onze de setembre 19

Ventas:

500

---- Listado de empleados ----

- 1 - Black Orchid
- 2 - Voldo Soul
- 3 - Blanca Street

Index del empleado que quieras anadir a la tienda:

1

Empleado anadido correctamente

Quieres anadir un nuevo empleado:

- 1 - Si
- 2 - No

2

Introduzca la operacion a realizar del siguiente menu de opciones:

- 1 - Muestra los empleados
- 2 - Muestra las tiendas
- 3 - Mostrar tiendas ordenadas por ventas
- 4 - Modificar un empleado

Members

- GestionaTiendas
- GestionaTiendas()
- createStore(Scanner scanner, EntityManager entityManager)
- getEmployee(EntityManager entityManager, int employeeId): Employee
- init(EntityManager entityManager)

Source

```
1 //
2 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 //
5 package objectDB;
6
7 import javax.persistence.EntityManager;
8 import javax.persistence.EntityManagerFactory;
9 import javax.persistence.Persistence;
10 import javax.persistence.TypedQuery;
11 import java.util.Collections;
12 import java.util.List;
13 import java.util.Random;
14 import java.util.Scanner;
15
16 /**
17  * @author Vadym Volokhov
18  */
19 public class GestionaTiendas {
20
21     public static void main(String[] args) {
22         String dbName = String.format("objectdb:%s/objectdb/db/points_1d.ojb", args[0] + new Random().nextInt());
23         EntityManagerFactory emf = Persistence.createEntityManagerFactory("objectdb:" + dbName);
24         EntityManager entityManager = emf.createEntityManager();
25
26         init(entityManager);
27
28         Scanner scanner = new Scanner(System.in);
29         boolean exit = false;
30
31         while (!exit) {
32             System.out.println("Introduzca la operacion a realizar del siguiente menu de opciones:");
33             System.out.println("1 - Muestra los empleados");
34             System.out.println("2 - Muestra las tiendas");
35             System.out.println("3 - Mostrar tiendas ordenadas por ventas");
36             System.out.println("4 - Modificar un empleado");
37             System.out.println("5 - Anade una tienda");
38             System.out.println("0 - salir");
39             int option = scanner.nextInt();
40             scanner.nextLine(); // borra buffer
41
42             switch (option) {
43                 case 1:
44                     showEmployees(entityManager);
45                     break;
46                 case 2:
47                     showStores(entityManager);
48             }
49         }
50     }
51 }
```

actividad07.libreria (run) #2 running... (1 more...) 17:26 [NS]

