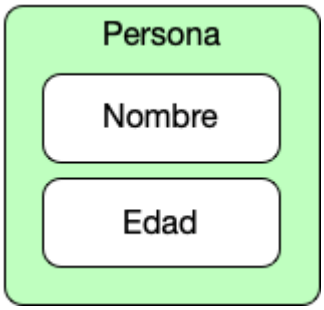
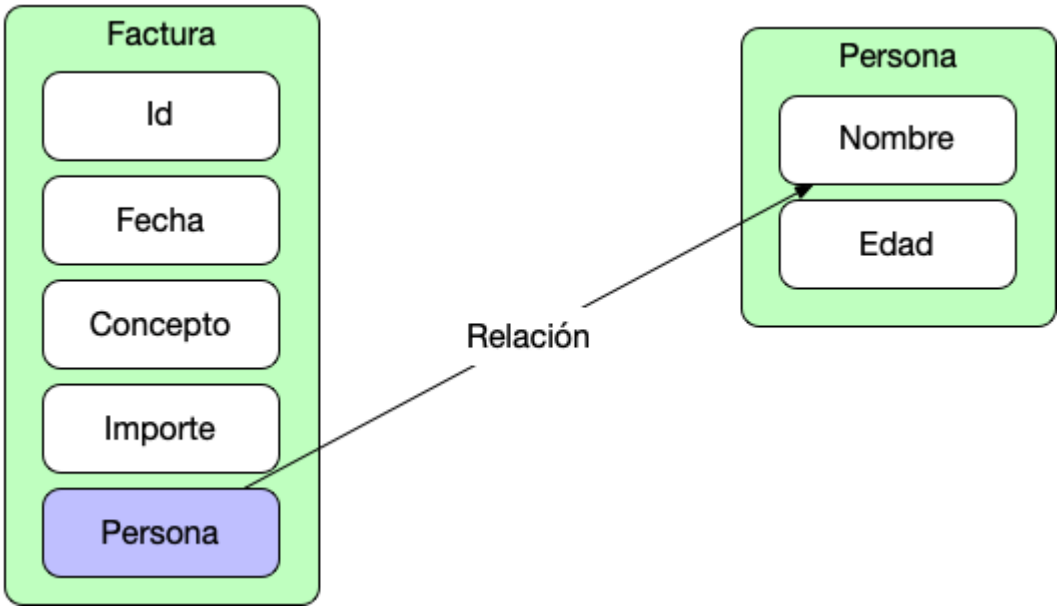
Vamos a ver como construir una relación JPA @ManyToOne entre dos clases Factura y Persona que tienen una relación de 1 a n entre ellas. Una Persona tiene varias Facturas .

Mientras que el concepto de Persona es mucho mas sencillo de entrada ya que solo dispone de nombre y edad.



En estos momentos por su definición ambas entidades son independientes . Es momento de relacionarlas para ello vamos a comenzar con la clase Factura. Toda Factura pertenece a una Persona en concreto y por lo tanto deberemos crear un campo de tipo Persona en la clase Factura.

Al crear una referencia estamos construyendo una relación entre ambos conceptos.



Vamos a verlo en código :

```
package com.arquitecturajava;
```

```
import java.util.Date;

public class Factura {
    private int id;
    private Date fecha;
    private double importe;
    private String concepto;
    private Persona persona;

    public Factura() {
        // No es necesario llamar a super() explícitamente aquí
    }

    public Factura(Date fecha, double importe, String concepto,
Persona persona) {
        this.fecha = fecha;
        this.importe = importe;
        this.concepto = concepto;
        this.persona = persona;
    }

    public Persona getPersona() {
        return persona;
    }

    public void setPersona(Persona persona) {
        this.persona = persona;
    }

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

```
    }

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

    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public double getImporte() {
        return importe;
    }

    public void setImporte(double importe) {
        this.importe = importe;
    }

    public String getConcepto() {
        return concepto;
    }

    public void setConcepto(String concepto) {
        this.concepto = concepto;
    }
}
```

Por ahora lo único que hemos creado es una referencia a la clase Persona sin utilizar JPA

para nada .El siguiente paso será anotar la clase con anotaciones de JPA para que se construya la relación a nivel de persistencia.

```
package es.curso.bo;

import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Factura {

    @Id
    private int id;
    private Date fecha;
    private double importe;
    private String concepto;

    @ManyToOne
    @JoinColumn(name="persona_nombre")
    private Persona persona;

    public Factura() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Factura(int id, Date fecha, double importe, String
concepto, Persona persona) {
```

```
        super();
        this.id = id;
        this.fecha = fecha;
        this.importe = importe;
        this.concepto = concepto;
        this.persona = persona;
    }

    public int getId() {
        return id;
    }

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

    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public double getImporte() {
        return importe;
    }

    public void setImporte(double importe) {
        this.importe = importe;
    }
```

```
    public String getConcepto() {
        return concepto;
    }

    public void setConcepto(String concepto) {
        this.concepto = concepto;
    }

    public Persona getPersona() {
        return persona;
    }

    public void setPersona(Persona persona) {
        this.
```

Disponemos de dos anotaciones :

@ManyToOne :Es una de las anotaciones mas habituales a nivel de JPA y se encarga de generar una relación de muchos a uno .
@JoinColumn:Esta anotación sirve en JPA para hacer referencia a la columna que es clave externa en la tabla y que se encarga de definir la relación . En nuestro caso la tabla Factura contendrá una columna persona_nombre con el nombre de la persona propietaria de la factura.

Una vez realizadas estas operaciones ya tenemos la primera parte de la relación construida. Es momento de ver como quedaría construida la otra parte en la que tenemos una Persona que contiene una lista de Facturas.

```
package es.curso.bo;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Persona {

    @Id
    private String nombre;
    private int edad;

    @OneToMany(mappedBy = "persona")
    private List<Factura> facturas;

    public Persona() {
```

```
        facturas = new ArrayList<>();
    }

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
        facturas = new ArrayList<>();
    }

    public String getNombre() {
        return nombre;
    }

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

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public List<Factura> getFacturas() {
        return facturas;
    }

    public void setFacturas(List<Factura> facturas) {
        this.facturas = facturas;
```

```
    }

    public void addFactura(Factura factura) {
        facturas.add(factura);
    }

    public void removeFactura(Factura factura) {
        facturas.remove(factura);
    }
}
```

En este caso usamos la anotación mappedBy para definir la otra parte de la relación .Con estas clases construidas podríamos generar un programa main que nos seleccione las facturas de una persona en concreto a través de su relación:

```
package es.curso;

import es.curso.bo.Factura;
import es.curso.bo.Persona;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("unidadpersonas");
        EntityManager em = emf.createEntityManager();
```

```
        // Nombre de la persona para la que queremos buscar las
facturas
        String nombrePersona = "Nombre de la persona"; // Coloca el
nombre de la persona que desees buscar

        // Obtener la persona
        Persona persona = em.find(Persona.class, nombrePersona);

        // Obtener la lista de facturas de la persona
        List<Factura> facturas = persona.getFacturas();

        // Imprimir las facturas encontradas
        System.out.println("Facturas de " + nombrePersona + ":");
        for (Factura factura : facturas) {
            System.out.println("ID: " + factura.getId() + ", Fecha: "
+ factura.getFecha() +
                    ", Importe: " + factura.getImporte() + ",
Concepto: " + factura.getConcepto());
        }

        em.close();
        emf.close();
    }
}
```

Acabamos de construir una relación entre dos Entidades haciendo hincapie en @ManyToOne como parte importante de la relación ya que es la que define la clave externa. El resultado de este programa es :

```
Facturas de Juan:
ID: 1, Fecha: 2024-02-10, Importe: 100.0, Concepto: Compra de
alimentos
```

```
ID: 2, Fecha: 2024-02-12, Importe: 50.0, Concepto: Gasolina
```

- ¿Que es un Java Stream?
- Java Fechas y el principio SRP
- JPA Remove y Objetos gestionados
- Java LocalDate y el manejo de Fecha
- Spring Boot REST JPA y JSON