

Carrera: Licenciatura en Sistemas

Materia: Orientación a Objetos II

Equipo docente:

Titular: Prof. María Alejandra Vranić

alejandravranic@gmail.com

Ayudantes: Prof. Leandro Ríos

leandro.rios.unla@gmail.com

Prof. Gustavo Siciliano

gussiciliano@gmail.com

Prof. Romina Mansilla

romina.e.mansilla@gmail.com



Año: 2018

IIDE: Eclipse

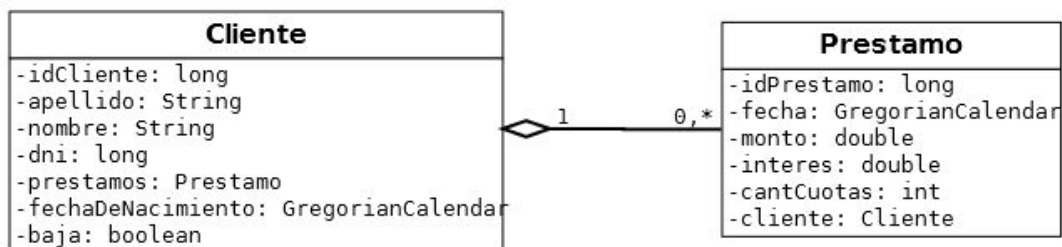
Persistencia de datos: feriado.xml, MySQL

Bibliografía: ver programa Hibernate

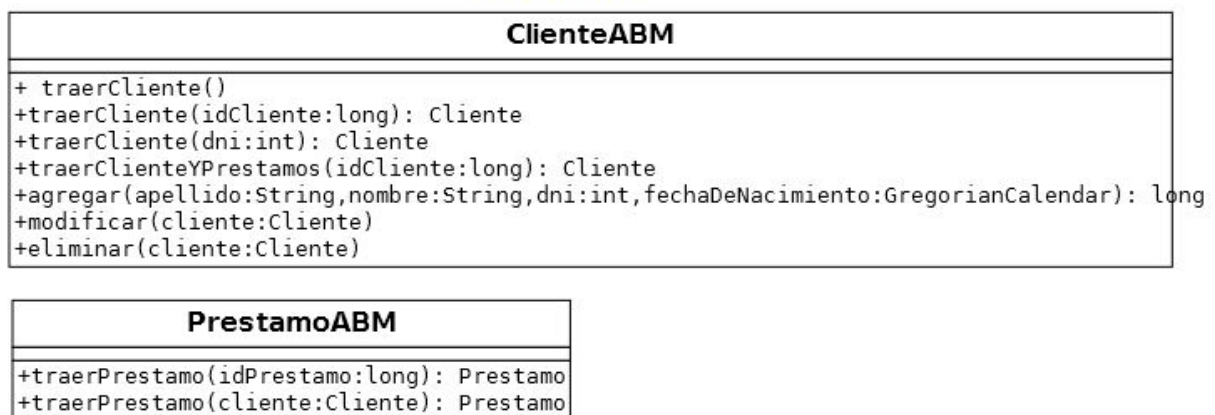
Framework Hibernate Relación Uno-a-Muchos

- Importar bd-hibernate-uno-a-muchos.sql
- Crear Proyecto HibernateUno-A-Muchos con los mismos fuentes del proyecto HibernateUnaEntidad
- Vamos a preparar la persistencia la clase Prestamo, para lo cual necesitamos definir una relación Uno-a-Muchos.

Datos



Negocio



A la clase Cliente anterior agregar las lineas resaltadas.

```
package datos;

import java.util.GregorianCalendar;
import java.util.Set;
import funciones.Funciones;

public class Cliente {
    private long idCliente;
    private String apellido;
    private String nombre;
    private int dni;
    private GregorianCalendar fechaDeNacimiento;
    private boolean baja;
    private Set<Prestamo> prestamos; //¿Porque utilizamos Set y no List?

    public Cliente(){}

    public Cliente(String apellido, String nombre, int dni,
        GregorianCalendar fechaDeNacimiento) {
        super();
        this.apellido = apellido;
        this.nombre = nombre;
        this.dni= dni;
        this.fechaDeNacimiento = fechaDeNacimiento;
        this.baja=false;
    }

    public long getIdCliente() {
        return idCliente;
    }

    protected void setIdCliente(long idCliente) {
        this.idCliente = idCliente;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getNombre() {
        return nombre;
    }

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

```

    public int getDni() {
        return dni;
    }

    public void setDni(int dni) {
        this.dni = dni;
    }

    public GregorianCalendar getFechaDeNacimiento() {
        return fechaDeNacimiento;
    }

    public void setFechaDeNacimiento(GregorianCalendar fechaDeNacimiento) {
        this.fechaDeNacimiento = fechaDeNacimiento;
    }

    public boolean isBaja() {
        return baja;
    }

    public void setBaja(boolean baja) {
        this.baja = baja;
    }

    public String toString(){
        return (idCliente+" "+apellido+" "+nombre+" DNI: "+dni+" F.de Nacimiento: "+Funciones.traerFechaCorta(fechaDeNacimiento)+" "+baja);
    }

    public Set<Prestamo> getPrestamos() {
        return prestamos;
    }

    public void setPrestamos(Set<Prestamo> prestamos) {
        this.prestamos = prestamos;
    }

}

```

- Crear la clase Prestamo en la capa de datos

```

package datos;

import java.util.GregorianCalendar;
import funciones.Funciones;

public class Prestamo {
    private long idPrestamo;
    private GregorianCalendar fecha;
    private double monto;
    private double interes;
    private int cantCuotas;
}

```

```

private Cliente cliente;

public Prestamo(){}

public Prestamo(GregorianCalendar fecha, double monto, double interes,
                int cantCuotas, Cliente cliente) {
    super();
    this.fecha = fecha;
    this.monto = monto;
    this.interes = interes;
    this.cantCuotas = cantCuotas;
    this.cliente = cliente;
}

public long getIdPrestamo() {
    return idPrestamo;
}

protected void setIdPrestamo(long idPrestamo) {
    this.idPrestamo = idPrestamo;
}

public GregorianCalendar getFecha() {
    return fecha;
}

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

public double getMonto() {
    return monto;
}

public void setMonto(double monto) {
    this.monto = monto;
}

public double getInteres() {
    return interes;
}

public void setInteres(double interes) {
    this.interes = interes;
}

public int getCantCuotas() {
    return cantCuotas;
}

public void setCantCuotas(int cantCuotas) {
    this.cantCuotas = cantCuotas;
}

public Cliente getCliente() {
    return cliente;
}

public void setCliente(Cliente cliente) {
    this.cliente = cliente;
}

public String toString(){
    String prestamo= "Prestamo: $ "+monto+"\nFecha:
"+Funciones.tracerFechaCorta(fecha)+"\nInteres: "+interes+"\nCant.de Cuotas:
"+cantCuotas;
    return prestamo;
}

}

```

- En el paquete mapeos en el archivo Cliente.hbm.xml agregar las líneas resaltadas

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<class name="datos.Cliente" table="cliente">
  <id column="idCliente" name="idCliente">
    <generator class="identity"/>
  </id>
  <property column="apellido" name="apellido" type="string"/>
  <property column="nombre" name="nombre" type="string"/>
  <property column="dni" name="dni" type="int"/>
  <property column="fechaDeNacimiento" name="fechaDeNacimiento" type="calendar"/>
  <property column="baja" name="baja" type="boolean"/>

  <set name="prestamos" table="prestamo" order-by="idPrestamo asc" inverse="true"
lazy="true" fetch="select">
    <key column="idCliente" not-null="true" />
    <one-to-many class="datos.Prestamo" />
  </set>

</class>
</hibernate-mapping>
```

- En el paquete mapeos crear el archivo Prestamo.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

<class name="datos.Prestamo" table="prestamo">

  <id column="idPrestamo" name="idPrestamo">
    <generator class="identity"/>
  </id>

  <property column="fecha" name="fecha" type="calendar"/>
  <property column="monto" name="monto" type="float"/>
  <property column="interes" name="interes" type="float"/>
  <property column="cantCuotas" name="cantCuotas" type="int"/>

  <many-to-one name="cliente" class="datos.Cliente"
    column="idCliente" not-null="true"/>

</class>
</hibernate-mapping>
```

- En el archivo de configuración hibernate.cfg.xml agregar la línea para que levante el mapeo de Prestamo (resaltada)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="connection.url">jdbc:mysql://localhost/bd-hibernate-uno-a-muchos</property>
    <property name="connection.username">root</property>
    <property name="connection.password">root</property>
    <property name="connection.pool_size">1</property>
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="show_sql">true</property>    <!-- en true muestra hql en consola-->

    <!--Mapeo Entidades -->
    <mapping resource="mapeos/Cliente.hbm.xml"/>
    <mapping resource="mapeos/Prestamo.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

- En ClienteDao agregar el código resaltado:

```
package dao;

import java.util.List;
import org.hibernate.Hibernate;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;
import datos.Cliente;

public class ClienteDao {
    private static Session session;
    private Transaction tx;

    private void iniciaOperacion() throws HibernateException {
        session = HibernateUtil.getSessionFactory().openSession();
        tx = session.beginTransaction();
    }

    private void manejaExcepcion(HibernateException he) throws HibernateException {
        tx.rollback();
        throw new HibernateException("ERROR en la capa de acceso a datos", he);
    }

    public int agregar(Cliente objeto) {
        int id = 0;
        try {
```

```

        iniciaOperacion();
        id = Integer.parseInt(session.save(objeto).toString());
        tx.commit();
    } catch (HibernateException he) {
        manejaExcepcion(he);
        throw he;
    } finally {
        session.close();
    }
    return id;
}

public void actualizar(Cliente objeto) throws HibernateException {
    try {
        iniciaOperacion();
        session.update(objeto);
        tx.commit();
    } catch (HibernateException he) {
        manejaExcepcion(he);
        throw he;
    } finally {
        session.close();
    }
}

public void eliminar(Cliente objeto) throws HibernateException {
    try {
        iniciaOperacion();
        session.delete(objeto);
        tx.commit();
    }
    catch (HibernateException he) {
        manejaExcepcion(he);
        throw he;
    }
    finally {
        session.close();
    }
}

    public Cliente traerCliente(long idCliente) throws HibernateException {
        Cliente objeto = null;
        try {
            iniciaOperacion();
            objeto = (Cliente) session.get(Cliente.class, idCliente);
        } finally {
            session.close();
        }
        return objeto;
    }

public Cliente traerCliente(int dni) throws HibernateException {
    Cliente objeto = null;

    try {
        iniciaOperacion();
        objeto = (Cliente) session.createQuery("from Cliente c where c.dni
="+dni).uniqueResult();

```

```

        } finally {
            session.close();
        }
        return objeto;
    }
    @SuppressWarnings("unchecked")
    public List<Cliente> traerCliente() throws HibernateException {
        List<Cliente> lista=null;
        try {
            iniciaOperacion();
            lista=session.createQuery("from Cliente c order by c.apellido asc, c.nombre asc").list();

        } finally {
            session.close();
        }

        return lista;
    }

    public Cliente traerClienteYPrestamos(long idCliente) throws HibernateException {
        Cliente objeto = null;

        try {
            iniciaOperacion();
            String hql="from Cliente c where c.idCliente ="+idCliente;
            objeto=(Cliente) session.createQuery(hql).uniqueResult();
            Hibernate.initialize(objeto.getPrestamos());
        }
        finally {
            session.close();
        }
        return objeto;
    }
}

```

- Para la capa Acceso a datos DAO (Data Object Access), crear la clase PrestamoDao donde están implementado método sobrecargado traerPrestamo por idPrestamo o por cliente.
- Implementar alta, modificación de un objeto Prestamo

```
package dao;
```

```
import java.util.List;
```

```
import org.hibernate.HibernateException;
```

```
import org.hibernate.Session;
```

```
import org.hibernate.Transaction;
```

```
import datos.Cliente;
```

```
import datos.Prestamo;
```

```
public class PrestamoDao {
```

```
    private static Session session;
```

```
    private Transaction tx;
```

```
    private void iniciaOperacion() throws HibernateException {
```

```
        session = HibernateUtil.getSessionFactory().openSession();
```



```

        tx = session.beginTransaction();
    }

    private void manejaExcepcion(HibernateException he) throws HibernateException {
        tx.rollback();
        throw new HibernateException("ERROR en la capa de acceso a datos", he);
    }

    public Prestamo traerPrestamo(long idPrestamo) throws HibernateException {

        Prestamo obj = null;
        try {
            iniciaOperacion();
            String hQL="from Prestamo p inner join fetch p.cliente c where
p.idPrestamo="+idPrestamo;
            obj = (Prestamo) session.createQuery(hQL).uniqueResult();
        } finally {
            session.close();
        }
        return obj;
    }

    @SuppressWarnings("unchecked")
    public List<Prestamo> traerPrestamo(Cliente c) throws HibernateException {
        List<Prestamo> lista=null;
        try {
            iniciaOperacion();
            String hQL="from Prestamo p inner join fetch p.cliente c where
c.idCliente="+c.getIdCliente();
            lista = session.createQuery(hQL).list();
        } finally {
            session.close();
        }
        return lista;
    }
}

```

- En negocio crear la clase PrestamoABM

```

package negocio;
import dao.PrestamoDao;
import java.util.List;
import datos.Cliente;
import datos.Prestamo;

public class PrestamoABM {
    private PrestamoDao dao=new PrestamoDao();

    public Prestamo traerPrestamo(long idPrestamo){
        //Implementar: si el no existe el prestamo lanzar la excepción
        Prestamo p =dao.traerPrestamo(idPrestamo);
        return p;
    }

    public List<Prestamo> traerPrestamo(Cliente c) {return dao.traerPrestamo(c);}
}

```

```

    /* Pendiente implementar
    * Alta, Modificar
    */
}

```

- Test

```
package test;
```

```
import java.util.List;
```

```
import datos.Cliente;
```

```
import datos.Prestamo;
```

```
import negocio.ClienteABM;
```

```
import negocio.PrestamoABM;
```

```
public class TestTraerPrestamo {
```

```
    public static void main(String[] args) {
```

```
        PrestamoABM prestamoABM=new PrestamoABM();
```

```
        long idPrestamo=1;
```

```
        System.out.println("\n---> TraerPrestamo idPrestamo="+idPrestamo+
"\n\n");
```

```
        Prestamo p=prestamoABM.traerPrestamo(idPrestamo);
```

```
        System.out.println(p + "\nPertenece a "+p.getCliente());
```

```
        //implementar Si este cliente no tiene prestamos otorgados imprima el
mensaje
```

```
        ClienteABM clienteABM=new ClienteABM();
```

```
        int dni=14000000;
```

```
        Cliente c= clienteABM.traerCliente(dni);
```

```
        System.out.println("\n---> TraerPrestamos del Cliente="+c+ "\n\n");
```

```
        List<Prestamo> prestamos=prestamoABM.traerPrestamo(c);
```

```
        //implementar Si este cliente no tiene prestamos otorgados imprima el
mensaje
```

```
        for (Prestamo o: prestamos) System.out.println(o + "\nPertenece a
"+o.getCliente());
```

```
    }
```

```
}
```

```
package test;
```

```
import negocio.ClienteABM;
```

```
import datos.Cliente;
```

```
import datos.Prestamo;
```

```
public class TestTraerClienteYPrestamos {
```

```
    public static void main(String[] args) {
```

```
        long idCliente=1;
```

```

        ClienteABM cliAbm=new ClienteABM();

        Cliente c= cliAbm.traerClienteYPrestamos(idCliente);
        System.out.println("\n---> Traer Cliente y Prestamos
idCliente="+idCliente);
        System.out.println("\n"+c);

        //implementar Si este cliente no tiene prestamos otorgados imprima el
mensaje
        for (Prestamo p: c.getPrestamos()) System.out.println("\n"+p);

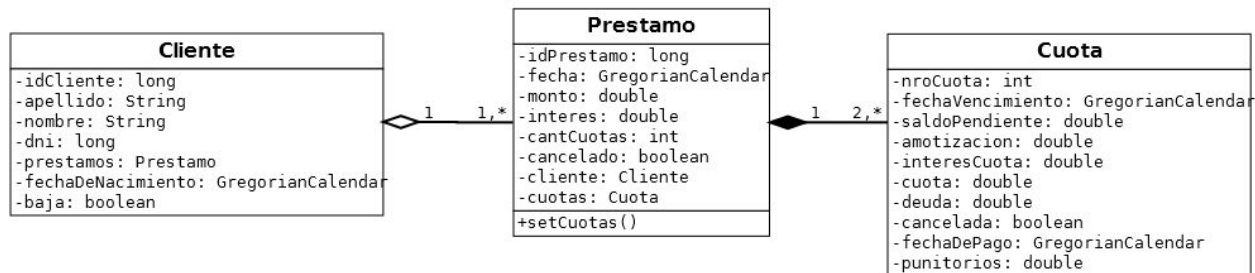
    }
}

```

Trabajo Práctico

- Dar persistencia a las cuotas del préstamo según el siguiente Diagrama de Clases

Datos



- El atributo cancelado se encontrará en true si el préstamo tiene todas sus cuotas pagas.
- En cuota el atributo cancelada cambiará a true cuando se paga la cuota. El atributo punitorios será calculado por un interés por mora en el pago de la cuota que ingresará por parámetro, por ejemplo 2% mensual.

¿Cómo realizar la persistencia el objeto Prestamo y sus objetos Cuota?

Agregando en el mapeo `Prestamo.hbm` en la relación `one-to-many` la propiedad `cascade="save-update"` cuando realizamos `agregar(Prestamo p)` de `PrestamoABM` Hibernate realiza el insert del préstamo y el de todas las cuotas a pagar automáticamente.

En el caso de producirse un error en el insert de todos los objetos ocurrirá un rollback (devuelve la base de datos al estado previo, por la sentencia `tx.commit()`; en el método `agregar` de `PrestamoDao` e Hibernate levantará una excepción).

```

<set name="prestamos" cascade="save-update" table="prestamo" order-by="idPrestamo asc"
inverse="true" lazy="true" fetch="select" >
    <key column="idCliente" not-null="true" />
    <one-to-many class="datos.Prestamo" />
</set>

```

En el caso de que un cliente venga a pagar una cuota se invocará al método `traerCuota` de `CuotaABM` se se "setearán" los atributos: `cancelada`, `fechaDePago`, `punitorios` y por último `modificarCuota` de `CuotaABM`.

Prestamos por Sistema Francés (cuota fija).

El sistema genera la cuota de la siguiente forma:

Un Préstamo bancario amortizado por el Sistema Francés que se pagará en n cuotas, el sistema deberá determinar el valor de cada cuota según el siguiente algoritmo:

Calculo de la 1° Cuota:

1. Entonces el primer *saldoPendiente* será el monto solicitado del crédito
2. Calculo de la amortizacion $amortizacion = \frac{saldoPendiente*interes}{(1+interes)^n - 1}$
3. Calculo del interés $interesCuota = saldoPendiente*interes$
4. Entonces el valor de la cuota será: $cuota = amortizacion + interesCuota$
5. Entonces la deuda pendiente será: $deuda = saldoPendiente - amortizacion$
6. Entonces el saldo pendiente será: $SaldoPendiente = SaldoPendiente - amortizacion$

Calculo de la 2° Cuota:

1. Calculo de la amortizacion $amortizacion = \frac{saldoPendiente*interes}{(1+interes)^{(n-1)} - 1}$
2. Calculo del interés $interesCuota = saldoPendiente*interes$
3. Entonces el valor de la cuota será: $cuota = amortizacion + interesCuota$
4. Entonces la deuda pendiente será: $deuda = saldoPendiente - amortizacion$
5. Entonces el saldo pendiente será: $SaldoPendiente = SaldoPendiente - amortizacion$

Así sucesivamente hasta obtener la cuota enésima.

Fecha de Vencimiento de la Cuota:

La fecha de vencimiento es mensual y la primera cuota vence al mes siguiente de la fecha de otorgamiento del préstamo. Será siempre días hábiles que son todos los que no sean sábado, domingo o feriado nacional. En el caso de ser el vencimiento un día feriado se pasará a siguiente día hábil.