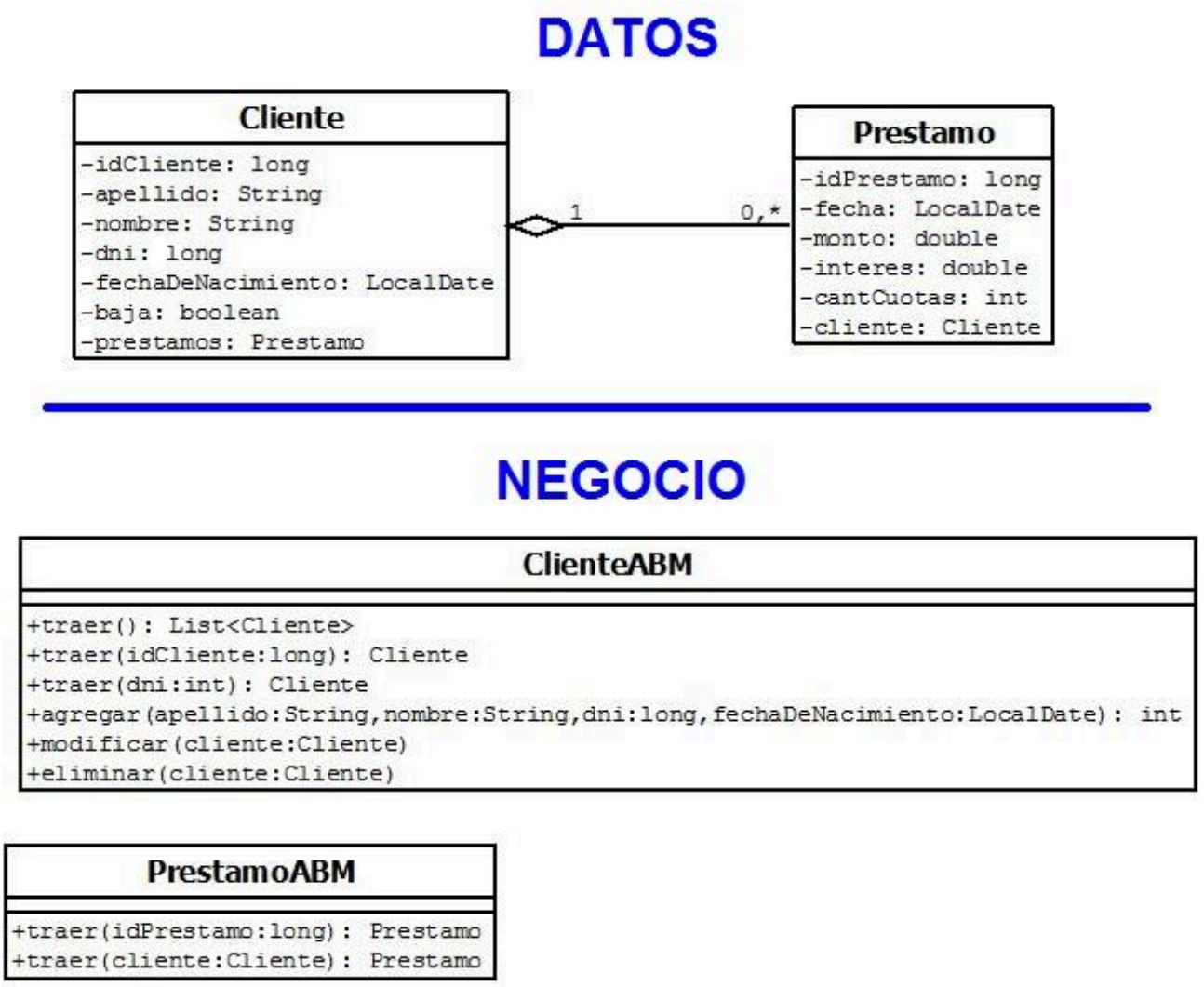
Licenciatura en Sistemas - Orientación a Objetos II – 2024

Prof. Titular: Mg. María Alejandra Vranić

Prof. Ayudantes: Esp. Lic. Gustavo Siciliano
Lic. Ezequiel Scordamaglia
Lic. Oscar Ruina

Índice

1) Diagrama de clases



2) Cambios sobre la clase Cliente.java

Se agregan el siguiente atributo, acompañando de los métodos get y set:

- **private** Set<Prestamo> prestamos; //¿Qué tiene de raro este atributo?
- **public** Set<Prestamo> getPrestamos() {
  return prestamos;
  }
- **public void** setPrestamos(Set<Prestamo> prestamos) {
  this.prestamos = prestamos;
  }

Nota: No agregar el atributo “prestamos” en el toString() al menos que sea necesitaría por un CU puntual.

3) Clase Prestamo.java

```
package datos;
import java.time.LocalDate;

public class Prestamo {
	private long idPrestamo;
	private LocalDate fecha;
	private double monto;
	private double interes;
	private int cantCuotas;
	private Cliente cliente;

	public Prestamo() {}

	public Prestamo(LocalDate 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 LocalDate getFecha() {
		return fecha;
	}
	public void setFecha(LocalDate 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;
	}

	@Override
	public String toString() {
		return "Prestamo [idPrestamo=" + idPrestamo + ", fecha=" + fecha + ",
monto=" + monto + ", interes=" + interes + ", cantCuotas=" + cantCuotas + "]";
	}
}
```

4) Cambios en el mapeo de Cliente.hbm.xml

Se agrega la siguiente etiqueta después de las properties para establecer la relación entre un cliente y sus préstamos.

```
<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>
```

5) Mapeo de 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="LocalDate"/>
	    <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>
```

6) Cambio en el archivo de configuración hibernate.cfg.xml

Debajo del mapping de Cliente se agrega el mapping de Prestamo. Además, hay que poner el nombre correcto de la BD (bd-hibernate-uno-a-muchos).

```
<mapping resource="mapeos/Cliente.hbm.xml"/>
<mapping resource="mapeos/Prestamo.hbm.xml"/>
```

7) Cambios en ClienteDao.java

Se agrega el siguiente método:

```
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).setParameter("idCliente", idCliente).uniqueResult();
         Hibernate.initialize(objeto.getPrestamos());
      }
      finally {
         session.close();
      }
      return objeto;
  }
```

8) Clase PrestamoDao.java

```
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 traerSinCliente(long idObjeto) {
		Prestamo objeto = null;
		try {
			iniciaOperacion();
			objeto = (Prestamo) session.get(Prestamo.class, idObjeto);
		} finally {
			session.close();
		}
		return objeto;
	}

	//IMPOTANTE el armado del HQL
	public Prestamo traer(long idPrestamo) {
		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).setParameter("idPrestamo",
idPrestamo).uniqueResult();
		} finally {
			session.close();
		}
		return obj;
	}

	//IMPOTANTE el armado del HQL
	public List<Prestamo> traer(Cliente c) {
		List<Prestamo> lista = null;
		try {
			iniciaOperacion();
			String hQL = "from Prestamo p inner join fetch p.cliente c where
c.idCliente=:idCliente";
			lista = session.createQuery(hQL, Prestamo.class).setParameter("idCliente",
c.getIdCliente()).getResultList();
		} finally {
			session.close();
		}
		return lista;
	}

	public int agregar(Prestamo 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(Prestamo objeto) {
		try {
			iniciaOperacion();
			session.update(objeto);
			tx.commit();
		} catch (HibernateException he) {
			manejaExcepcion(he);
			throw he;
		} finally {
			session.close();
		}
	}
}
```

9) Cambios en ClienteABM.java

Se agrega el siguiente método:

```
public Cliente traerClienteYPrestamos(long idCliente) {
	return dao.traerClienteYPrestamos(idCliente);
}
```

10) Clase PrestamoABM.java

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

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

	public Prestamo traerPrestamo(long idPrestamo){
		return dao.traer(idPrestamo);
	}

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

	public int agregar(LocalDate fecha, double monto, double interes, int cantCuotas, Cliente cliente){
		// Pendiente implementar lógica de negocio
		Prestamo c = new Prestamo(fecha, monto, interes, cantCuotas, cliente);
		return dao.agregar(c);
	}

	public void modificar(Prestamo c) {
		// Pendiente implementar lógica de negocio
		dao.actualizar(c);
	}
}
```

11) Testeos

TestTraerPrestamo

```
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.printf("TraerPrestamo idPrestamo=%d\n", idPrestamo);
		Prestamo prestamo = prestamoABM.traerPrestamo(idPrestamo);
		System.out.printf("\n%s pertenece a %s\n", prestamo, prestamo.getCliente());

		ClienteABM clienteABM = new ClienteABM();
		int dni = 14000000;
		Cliente cliente = clienteABM.traer(dni);

		System.out.printf("\nTraerPrestamos del Cliente=%s\n", cliente);
		List<Prestamo> prestamos = prestamoABM.traerPrestamo(cliente);
		for (Prestamo p : prestamos)
			System.out.printf("\n%sPertenece a %s\n", p, p.getCliente());
	}
}
```

TestTraerClienteYPrestamos

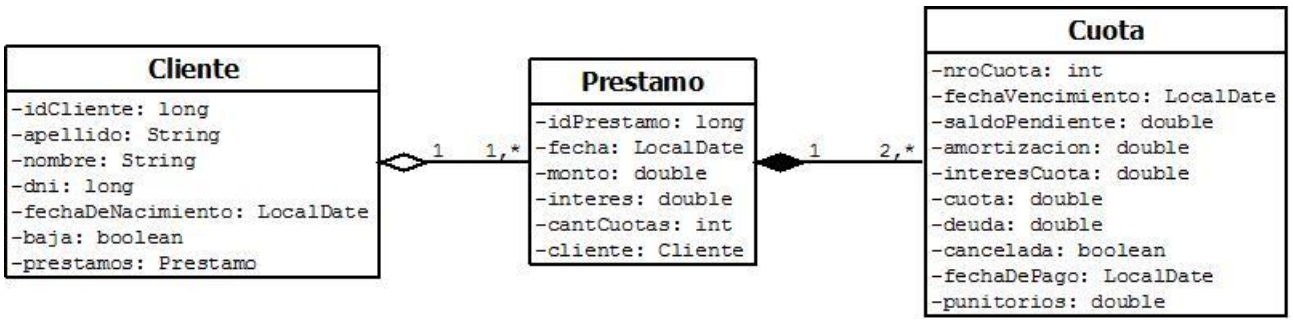
```
package test;
import negocio.ClienteABM;
import datos.Cliente;

public class TestTraerClienteYPrestamos {
	public static void main(String[] args) {

		long idCliente=1;
		ClienteABM cliAbm=new ClienteABM();
		Cliente cliente = cliAbm.traerClienteYPrestamos(idCliente);
		System.out.printf("Traer Cliente y Prestamos idCliente=%d\n", idCliente);
		System.out.printf("\n%s\n", cliente);
		System.out.printf("\n%s", cliente.getPrestamos());
	  }
}
```

12) Trabajo Práctico

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



- Tener en cuenta que el cliente debe tener una lista de tipo Set de préstamos.
- Agregar un id en Cuota para manejarlo como PK a nivel BD.
- Agregar un atributo cancelado en la clase Prestamo, el mismo 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.

Préstamos por Sistema Francés (cuota fija).

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 amortizaron $amortizacion = \frac{saldoPendiente * interes}{(1+interes)^{n}-1}$
3. Calculo del interés $interesCuota = saldoPendiente * interes$
4. Entonces el valor de la cuota será: $cuota = amoritizacion + interesCuota$
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 amortizaron $amortizacion = \frac{saldoPendiente * interes}{(1+interes)^{(n-1)}-1}$
2. Calculo del interés $interesCuota = saldoPendiente * interes$
3. Entonces el valor de la cuota será: $cuota = amoritizacion + interesCuota$
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 al siguiente día hábil.