TALLER DE LENGUAJE DE PROGRAMACIÓN I

Entorno de CONSOLA y GRÁFICOS

UNIDAD 2

Programación Orientada a Objetos – POO

Serialización – Transferencia de Objetos

JAVA

OBJETIVO

- Qué es la Serialización?
- Cambio de Versión
  - serialVersionUID

Trabajo Independiente

- Ejercicio

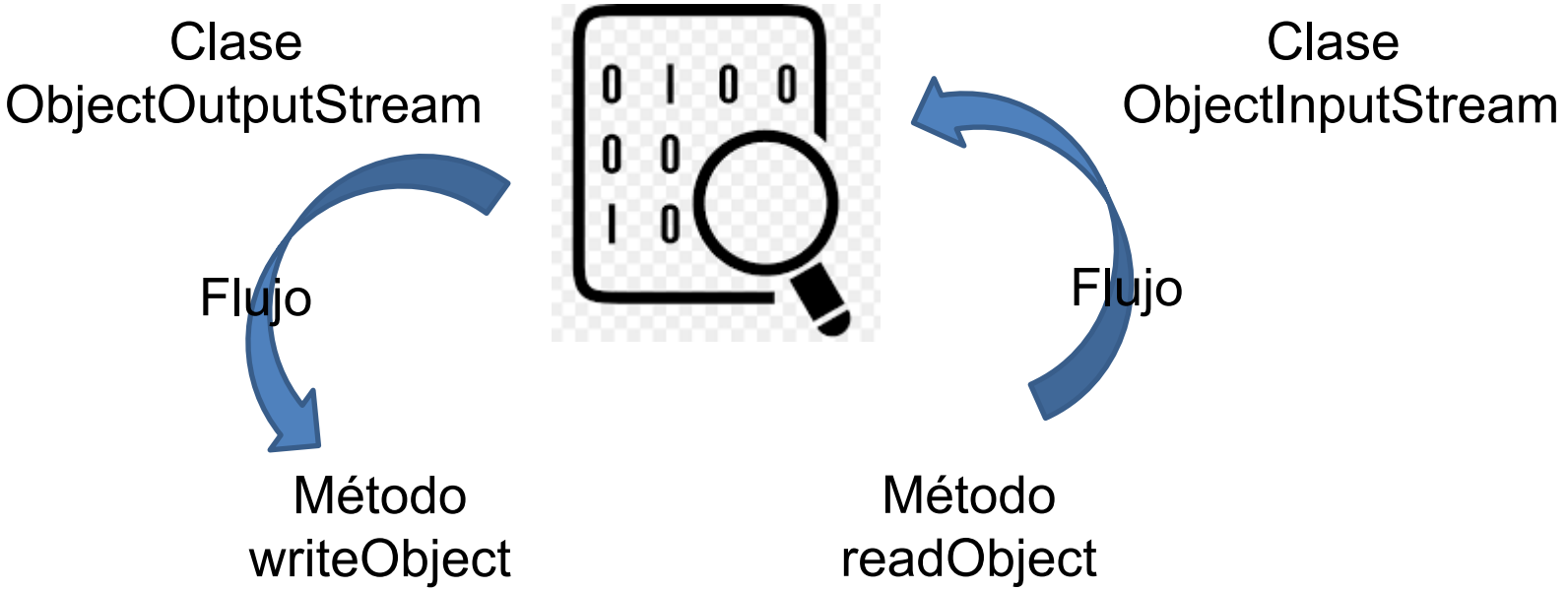
La Serialización

Es la opción de la entrada/salida de datos que en Java realizaremos por medio de ***flujos (Streams)*** de datos, a través de los cuales un programa podrá recibir o enviar datos en serie (binarios) constituidos como un objeto.

Si queremos transferir estructuras de datos complejas (objetos), debemos convertir estas estructuras en secuencias de bytes (Serializarlos) que puedan ser enviadas a través de un flujo.

Serialización (Flujo de datos)

Interface ***Serializable***



Aplicación de la Serialización

Retomando el ejercicio de Empleado y Empleado Jefe aplicamos los conceptos de serialización.

- Crear un proyecto con nombre pooNomina.
- Crear tres paquetes bean, logica y vista
  - Crear las clases
    - Paquete Bean: BEmpleado
    - Paquete Lógica: LEmpleado, LEmpleadoJefe
    - Paquete vista: VInicio, VPedirDatosEmpleado

Aplicación de la Serialización

Clase **BEmpleado**

```
package bean;

import java.time.LocalDate;


public class BEmpleado {

    private int    idEmp;
    private String nombreEmp;
    private long   sueldoEmp;
    private LocalDate fechaNacEmp;

    public BEmpleado() {}
}
```

```
public int getIdEmp() {
    return idEmp;
}

public void setIdEmp(int idEmp) {
    this.idEmp = idEmp;
}

public String getNombreEmp() {
    return nombreEmp;
}

public void setNombreEmp(String nombreEmp) {
    this.nombreEmp = nombreEmp;
}

public long getSueldoEmp() {
    return sueldoEmp;
}

public void setSueldoEmp(long sueldoEmp) {
    this.sueldoEmp = sueldoEmp;
}

public LocalDate getFechaNacEmp() {
    return fechaNacEmp;
}

public void setFechaNacEmp(LocalDate fechaNacEmp) {
    this.fechaNacEmp = fechaNacEmp;
}
```

Aplicación de la Serialización

Clase **LEmpleado**

```
package logica;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.time.LocalDate;


import bean.BEmpleado;

public class LEmpleado
                extends BEmpleado {

   public LEmpleado() {}
}
```

Método Constructor

```
public LEmpleado(int id, String nombre, long sueldo, int annio, int mes, int dia)
{
   super.setIdEmp(id);
   super.setNombreEmp(nombre);
   super.setSueldoEmp(sueldo);
   super.setFechaNacEmp(LocalDate.of(annio,mes,dia));}

public LEmpleado(BEmpleado empleado, LocalDate fechaNacimientoEmp)
{
   super.setIdEmp(empleado.getIdEmp());
   super.setNombreEmp(empleado.getNombreEmp());
   super.setSueldoEmp(empleado.getSueldoEmp());
   super.setFechaNacEmp(fechaNacimientoEmp);}

public LEmpleado(BEmpleado empleado)
{
   super.setIdEmp(empleado.getIdEmp());
   super.setNombreEmp(empleado.getNombreEmp());
   super.setSueldoEmp(empleado.getSueldoEmp());
   super.setFechaNacEmp(empleado.getFechaNacEmp());}
```

Aplicación de la Serialización

Clase **LEmpleado** → Método de Acción

```
public void LEmpleadoRegistrar(int id,String nombre, long sueldo, int annio, int mes, int dia) {
   super.setIdEmp(id);
   super.setNombreEmp(nombre);
   super.setSueldoEmp(sueldo);
   super.setFechaNacEmp(LocalDate.of(annio,mes,dia));

   LEmpleadoIngresar();
}

public void LEmpleadoRegistrar(BEmpleado empleado, LocalDate fechaNacimientoEmp)
{
    super.setIdEmp(empleado.getIdEmp());
    super.setNombreEmp(empleado.getNombreEmp());
    super.setSueldoEmp(empleado.getSueldoEmp());
    super.setFechaNacEmp(fechaNacimientoEmp);

    LEmpleadoIngresar();
}
```

Aplicación de la Serialización

Clase **LEmpleado** → Método de Acción (continuación)

```
public void LEmpleadoRegistrar(BEmpleado empleado)
{
   super.setIdEmp(empleado.getIdEmp());
   super.setNombreEmp(empleado.getNombreEmp());
   super.setSueldoEmp(empleado.getSueldoEmp());
   super.setFechaNacEmp(empleado.getFechaNacEmp());

   LEmpleadoIngresar();
}

public void LAumentoSueldo(double porcentaje)
{
            long aumento = (long)(getSueldoEmp() * porcentaje/100);
            super.setSueldoEmp(super.getSueldoEmp() + aumento);
}
```

Aplicación de la Serialización

Clase **LEmpleado** → Método de Acción (continuación)

```
public void LEmpleadoIngresar()
{
        try
        {
                RandomAccessFile archivo = new RandomAccessFile("./empleados.txt","rw");
                archivo.seek(archivo.length());

                archivo.writeInt(super.getIdEmp());
                archivo.writeUTF(super.getNombreEmp());
                archivo.writeLong(super.getSueldoEmp());
                archivo.writeUTF(String.valueOf(super.getFechaNacEmp()));

                archivo.close();
        }catch(IOException e) {System.out.print(e); }
}
```

Aplicación de la Serialización

Clase **LEmpleadoJefe** → Variables de clase y Método Constructor

```
package logica;

import java.time.LocalDate;

import bean.BEmpleado;

public class LEmpleadoJefe extends LEmpleado
{
		private long bonificacion;

		public LEmpleadoJefe(int id,String nombre, long sueldo, int annio, int mes, int dia) {
			super(id,nombre,sueldo,annio,mes,dia);
		}

		public long getSueldoEmp() {
			long salariojefe = super.getSueldoEmp();
			return (salariojefe + bonificacion);
		}
}
```

Aplicación de la Serialización

Clase **LEmpleadoJefe** → Método de Acción (continuación)

```
public LEmpleadoJefe(int id,String nombre, long sueldo, int annio, int mes, int dia) {
    super(id,nombre,sueldo,annio,mes,dia);
}

public void LEmpleadoJefeRegistrar(BEmpleado empleado, LocalDate fechaNacEmp)
{
    super.LEmpleadoRegistrar(empleado, fechaNacEmp);
}

public void estableceBonificacion(long valor) {
    bonificacion = valor;
}
```

Aplicación de la Serialización

Clase **VPedirDatosEmpleado** → Método Constructor y variables de trabajo

```
package vista;

import java.time.LocalDate;
import java.util.Scanner;

import logica.LEmpleado;
import logica.LEmpleadoJefe;

public class VPedirDatosEmpleado {

	public VPedirDatosEmpleado() {

		Scanner captura = new Scanner(System.in);

		System.out.println("Especifique el número de empleados para ingresar.");
		int nroEmpleados = 0;
		String tipoEmpleado;
		LEmpleado[] lempleado;
		nroEmpleados = captura.nextInt();
```

Ver el código de la siguiente diapositiva

```
}
```

Se ejecuta la captura de los datos del Empleado, aplicando el Polimorfismo

Aplicación de la Serialización

Clase **VPedirDatosEmpleado** → Lógica de captura de datos

```
if(nroEmpleados > 0){
    LEmpleado[] lempleado = new LEmpleado[nroEmpleados];
    for(int i=0; i < (lempleado.length); i++) {
        int annio, mes, dia;
        lempleado[i] = new LEmpleado();

        System.out.println("Especifique el Id del empleado " + (i+1));
        lempleado[i].setIdEmp(captura.nextInt());

        captura.nextLine();
        System.out.println("Especifique el Nombre del empleado " + (i+1));
        lempleado[i].setNombreEmp(captura.nextLine());
        lempleado[i].setSueldoEmp(1000000);

       System.out.println("Especifique el Año de Naciomiento del Empleado " + (i+1));
       annio= captura.nextInt();

       System.out.println("Especifique el mes de Naciomiento del Empleado " + (i+1));
       mes= captura.nextInt();

       System.out.println("Especifique el día de Naciomiento dle Empleado " + (i+1));
       dia= captura.nextInt();
       lempleado[i].setFechaNacEmp(LocalDate.of(annio, mes, dia));

       System.out.println("El Empleado es Jefe (S/N)?");
       tipoEmpleado = captura.next();

     if (tipoEmpleado.contentEquals("S")) {
       LEmpleadoJefe jefe_sistemas = new LEmpleadoJefe(lempleado[i].getIdEmp(),lempleado[i].getNombreEmp(),lempleado[i].getSueldoEmp(),annio,mes,dia);
       jefe_sistemas.estableceBonificacion(1000000);
       lempleado[i] = jefe_sistemas;
     }
}
```

Aquí se aplica el concepto del Polimorfismo

Ver el código de la siguiente diapositiva

Aplicación de la Serialización

Clase **VPedirDatosEmpleado** → Lógica de captura de datos continuación

```
      for(LEmpleado e : lempleado ) {
           e.LAumentoSueldo(3.0);
      }

      for(LEmpleado e : lempleado) {
           System.out.println("Datos de la Lista antes de enviar al archivo de datos");
           System.out.println("Empleado: " + e.getIdEmp() + " Nombre: " + e.getNombreEmp() + " Sueldo: " + e.getSueldoEmp() +
                              " Fecha Nac: " +  e.getFechaNacEmp());
           System.out.println("-------------------------------------------------------");

            LEmpleado emp = new LEmpleado();
            emp.setIdEmp(e.getIdEmp());
            emp.setNombreEmp(e.getNombreEmp());
            emp.setSueldoEmp(e.getSueldoEmp());
            emp.setFechaNacEmp(e.getFechaNacEmp());

             emp.LEmpleadoRegistrar(emp);
       }
} //Fin del cierto del if numero de empleado para capturar
else {
      System.out.println("Canceló la captura de empleados.");
}
captura.close();
}
}
```

Aplicación de la Serialización

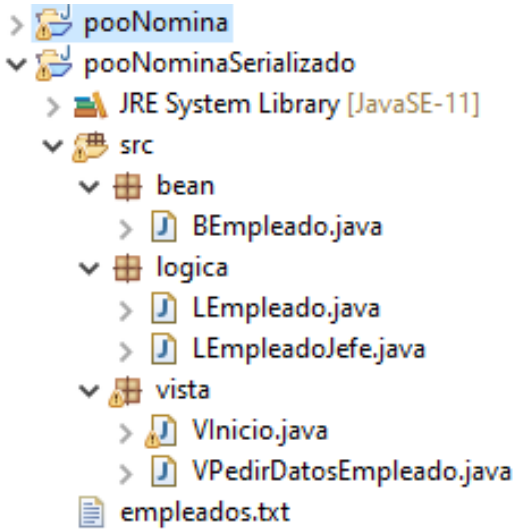
Clase **VInicio** → Lógica del metodio main

```
package vista;

public class VInicio {

        public static void main(String[] args)
       {

            VPedirDatosEmpleado empleados = new VPedirDatosEmpleado();
        }
}
```

Aplicación de la Serialización

Retomando el ejercicio de Empleado y Empleado Jefe aplicamos los conceptos de serialización.

Realice una copia del proyecto pooNomina como pooNominaSerializado



Aplicación de la Serialización

Se debe identificar el objeto que se desea serializar, esto quiere decir que podrán ser transmitidos (***flujos - Streams***) por la solicitud de un cliente (Proceso que solicita o entrega información de los Empleados).

Serialización – Transferencia de Datos (POO), continuación

```
package poo_nominaserializacion;

import java.io.*;
import java.util.Scanner;

public class Inicio {
        public static void main(String[] args)
        {
                Scanner captura = new Scanner(System.in);
                System.out.println("Especifique el núméor de empleados para ingresar.");
                int nroEmpleados = 0;
                String tipoEmpleado;
                int id;
                String nombre;
                long   sueldo;
                nroEmpleados = captura.nextInt();
                if(nroEmpleados > 0){
                        LEmpleado[] lempleado = new LEmpleado[nroEmpleados];

                        for(int i=0; i < (lempleado.length); i++) {
                                System.out.println("Especifique el Id del empleado " + (i+1));
                                id = captura.nextInt();
                                System.out.println("Especifique el Nombre del empleado " + (i+1));
                                nombre = captura.next();
                                sueldo = 1000000;

                                System.out.println("El Empleado es Jefe (S/N)?");
                                tipoEmpleado = captura.next();

                                if (tipoEmpleado.contentEquals("S")) {
                                        LEmpleadoJefe jefe_sistemas = new LEmpleadoJefe(id,nombre,sueldo,1998,8,28);
                                        jefe_sistemas.estableceBonificacion(1000000);
                                        lempleado[i] = jefe_sistemas; //Se aplica el Polimorfismo
                                }else{
                                        lempleado[i] = new LEmpleado(id,nombre,sueldo,1998,8,28);
                                }
                        }
                }
                else {
                        System.out.println("Canceló la captura de empleados.");
                }
                captura.close();
        }
}
```

El Objeto que se desea Serializar, en el cual se hereda la clase BEmpleado quien tiene la estructura de datos de la Entidad Empleado .

La Serialización

Se debe entonces a la clase que define el objeto (para este ejemplo LEmpleado) especificarle que implemente la interface ***Serializable***, *pero tenga presente que la clase LEmpleado hereda la clase BEmpleado quien define la estructura de datos de la Entidad, la clase entonces que se debería serializar es la clase BEmpleado*

Aplicación de la Serialización

Clase **BEmpleado**

```
package bean;

import java.io.Serializable;
import java.time.LocalDate;


public class BEmpleado
             implements Serializable
{

   private int    idEmp;
   private String nombreEmp;
   private long   sueldoEmp;
   private LocalDate fechaNacEmp;

   public BEmpleado() {}
}
```

Todas las clases que extiendan la clase **LEmpleado**, son susceptibles de serializar

```
public int getIdEmp() {
   return idEmp;
}

public void setIdEmp(int idEmp) {
   this.idEmp = idEmp;
}

public String getNombreEmp() {
   return nombreEmp;
}

public void setNombreEmp(String nombreEmp) {
   this.nombreEmp = nombreEmp;
}

public long getSueldoEmp() {
   return sueldoEmp;
}

public void setSueldoEmp(long sueldoEmp) {
   this.sueldoEmp = sueldoEmp;
}

public LocalDate getFechaNacEmp() {
   return fechaNacEmp;
}

public void setFechaNacEmp(LocalDate fechaNacEmp) {
   this.fechaNacEmp = fechaNacEmp;
}
```

La Serialización

Adicionar los métodos que generan el Flujo de datos (Stream) ya sea para recibir o para enviar datos binarios en la clase LEmpleado.

Aplicación de la Serialización

Clase **LEmpleado** → Métodos que serializan para salida

```
public void LEmpleadoSerializar()
{
		ArrayList<BEmpleado> listaEmpleados = recuparDatosEmpleados();
		BEmpleado[] lempleado = new BEmpleado[listaEmpleados.size()];

		for(int i=0; i < listaEmpleados.size(); i++)
		{
				lempleado[i] = new BEmpleado();
				lempleado[i].setIdEmp(listaEmpleados.get(i).getIdEmp());
				lempleado[i].setNombreEmp(listaEmpleados.get(i).getNombreEmp());
				lempleado[i].setSueldoEmp(listaEmpleados.get(i).getSueldoEmp());
				lempleado[i].setFechaNacEmp(listaEmpleados.get(i).getFechaNacEmp());
		}

		try
		{
		    FileOutputStream ruta_salida = new FileOutputStream("./empleadosSerializados.txt");
		    ObjectOutputStream archivo_salida = new ObjectOutputStream(ruta_salida);

		    archivo_salida.writeObject(lempleado);
		    archivo_salida.close();

		}catch(Exception e)
		{
				System.out.println("Problemas con el acceso al archivo de datos - " + e);
		 }
}
```

Aplicación de la Serialización

Clase **LEmpleado** → Métodos que serializan para leer el archivo serializado

```
public void LEmpleadoLeerSerializado()
        {
                try
                {
                  FileInputStream ruta_entrada = new FileInputStream("./empleadosSerializados.txt");
                  ObjectInputStream leer_datos = new ObjectInputStream(ruta_entrada);

                  BEmpleado[] datos_leidos = (BEmpleado[]) leer_datos.readObject();
                  leer_datos.close();

                  String cadena = "";
                  for(int i=0; i < datos_leidos.length;i++)
                  {
                                cadena = datos_leidos[i].getIdEmp() + " - " + datos_leidos[i].getNombreEmp() ;
                                System.out.println(cadena);

                  }

                }catch(Exception e) {
                    System.out.println(e);
                   System.out.println("Problemas con el acceso al archivo de datos.");}
        }
```

La Serialización

Una vez capturados los datos y almacenados en un archivo plano aleatorio, se procede a la generación del archivo de datos **Serializado** y que pueda ser transmitido.

Aplicación de la Serialización

Clase **VPedirDatosEmpleado** → Lógica de captura de datos continuación

```
    for(LEmpleado e : lempleado ) {
        e.LAumentoSueldo(3.0);
    }

    for(LEmpleado e : lempleado) {
        System.out.println("Datos de la Lista antes de enviar al archivo de datos");
        System.out.println("Empleado: " + e.getIdEmp() + " Nombre: " + e.getNombreEmp() + " Sueldo: " + e.getSueldoEmp() +
                           " Fecha Nac: " +  e.getFechaNacEmp());
        System.out.println("-------------------------------------------------------");

         LEmpleado emp = new LEmpleado();
         emp.setIdEmp(e.getIdEmp());
         emp.setNombreEmp(e.getNombreEmp());
         emp.setSueldoEmp(e.getSueldoEmp());
         emp.setFechaNacEmp(e.getFechaNacEmp());
                                                  S
          emp.LEmpleadoSerializar();
    }
    emp.LEmpleadoLeerSerializado();
} //Fin del cierto del if numero de empleado para capturar
else {
    System.out.println("Canceló la captura de empleados.");
}
captura.close();
}
}
```

Se hace el llamado del método que serializa los datos

Trabajo Independiente

- Desarrollar los ejercicios del Taller de modelamiento e identificar y crear los proyectos por cada uno de ellos, definiendo clases, objetos, métodos constructores, métodos modificadores de datos, construir composiciones, Encapsulamiento, Herencia, Polimorfismo, Abstracciones, Interfaces y Serialización.