Universidad Distrital Francisco José de caldas
Tecnología en Sistematización de Datos

Persistencia Con Archivos

Programación Multinivel
Sonia Alexandra Pinzón Nuñez
2021

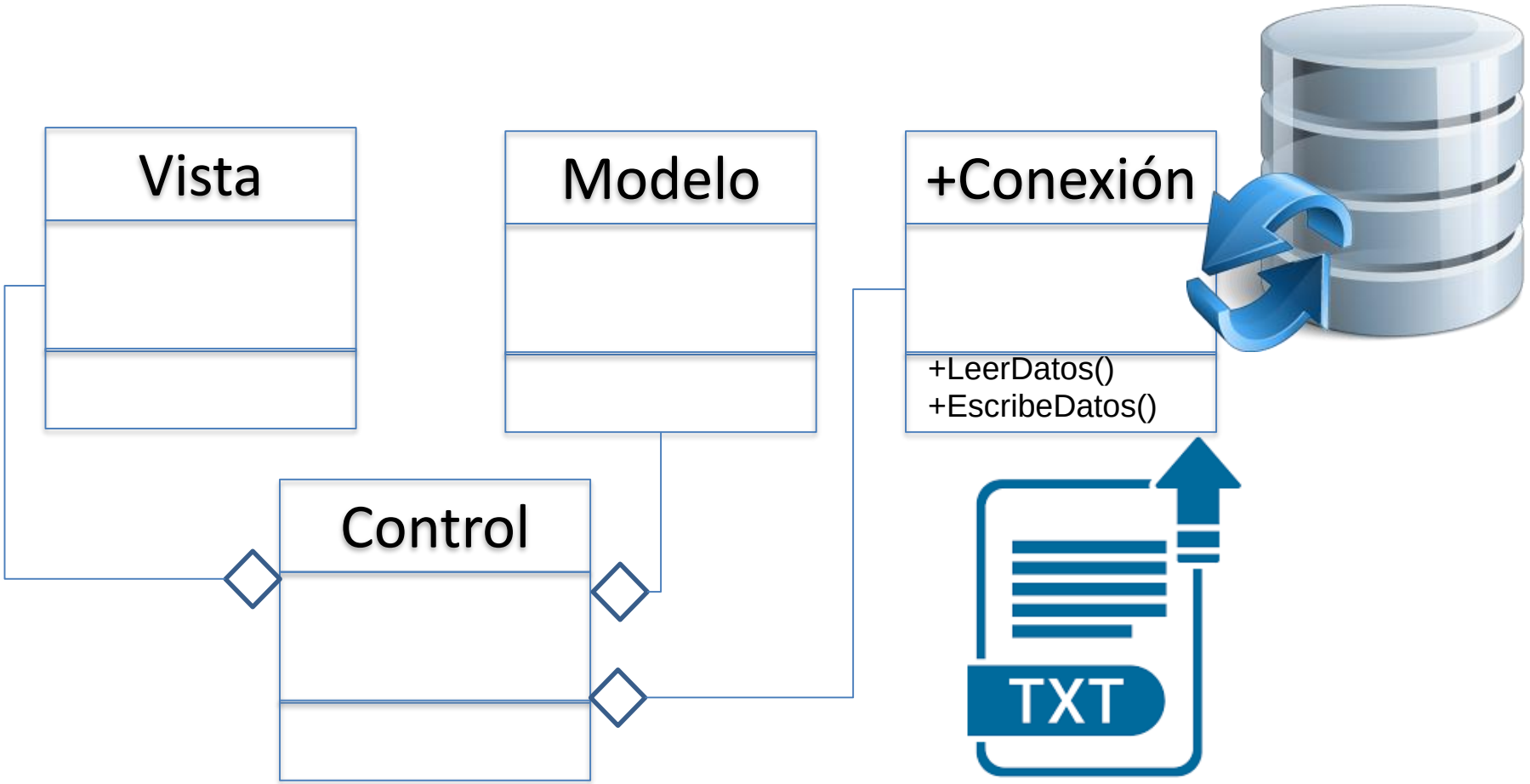
Introducción

Hasta el momento las aplicaciones implementadas obtienen y generan los datos para la ejecución de la entrada y salida estándar significa que:

- Los datos solo estarán disponibles mientras dura la ejecución de la aplicación.
- Los datos deben ingresarse cada vez que se ejecuta la aplicación (no perduran)

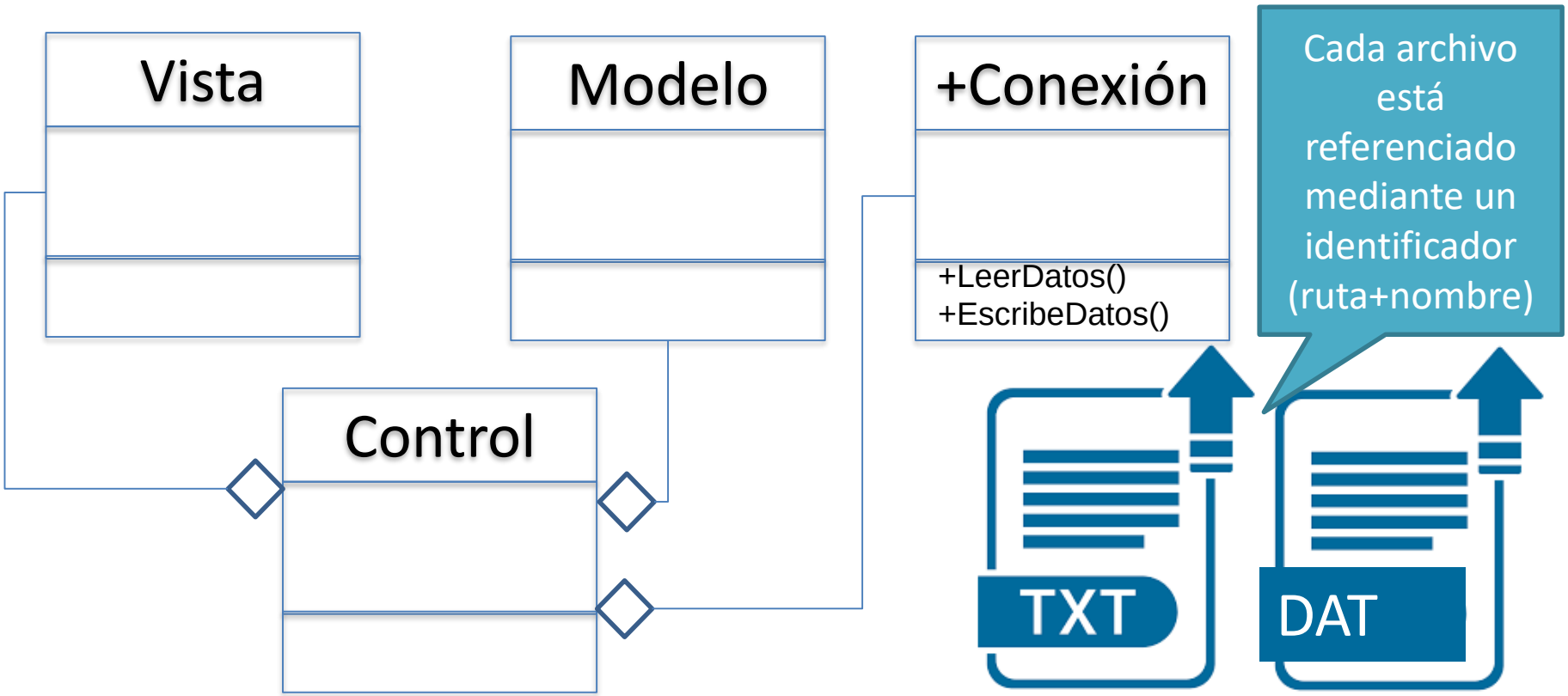
Introducción

Para garantizar que los datos perduren es necesario almacenarlos en Archivos o BD

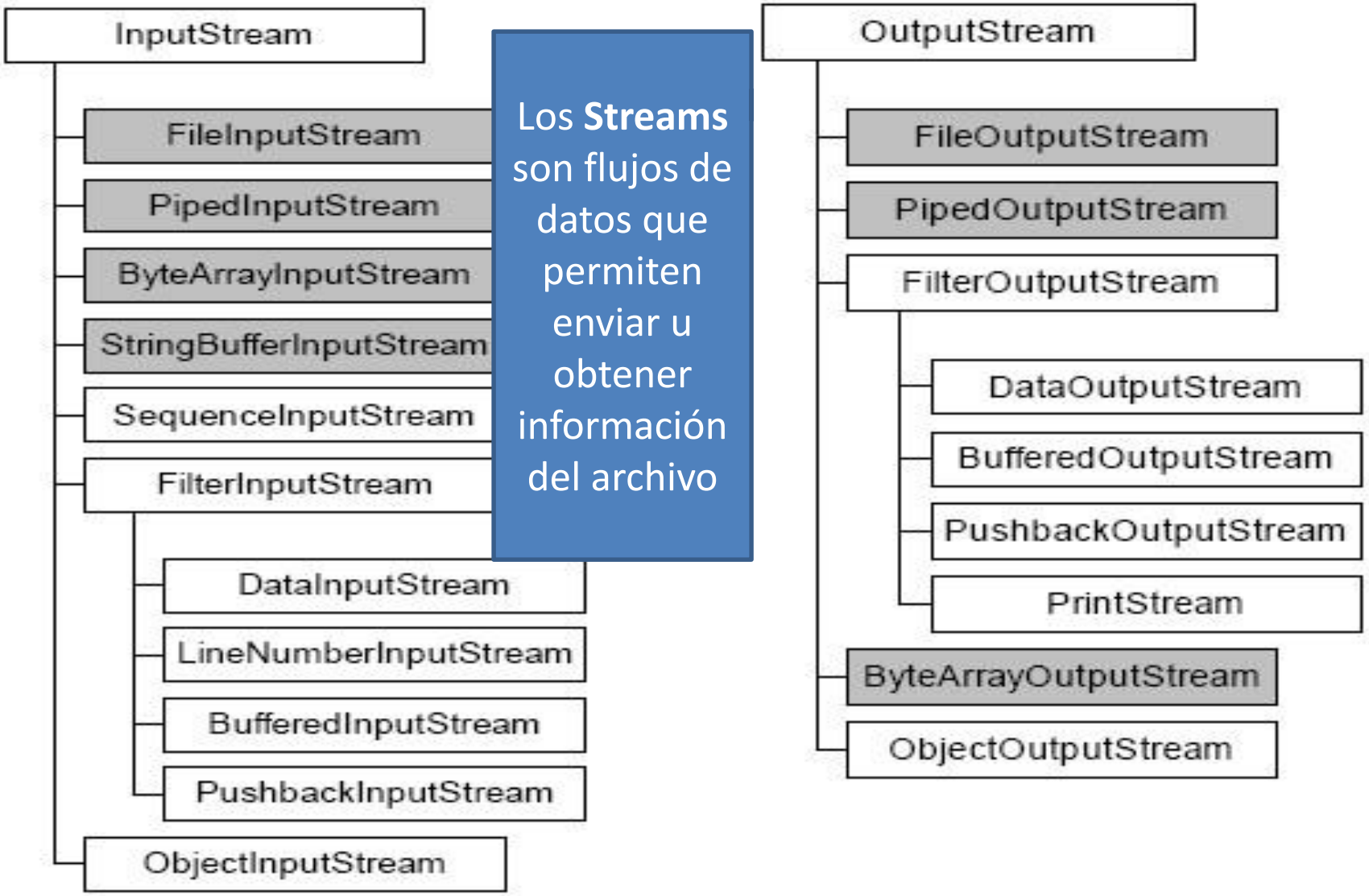


Persistencia con Archivos

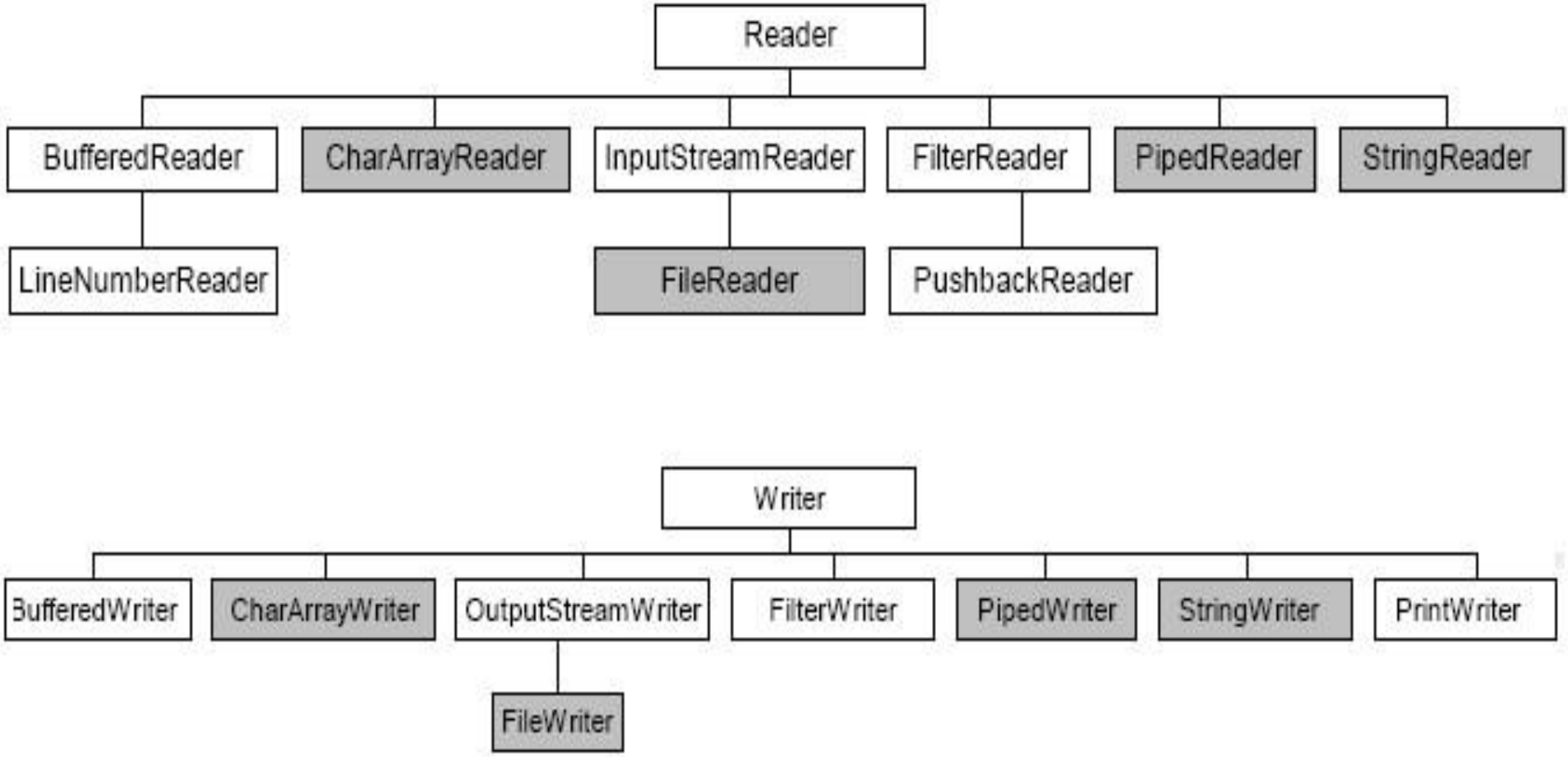
Los datos almacenados en archivos son permanentes permitiendo su manipulación en cualquier momento.



Clases Java para Lectura y escritura de archivos



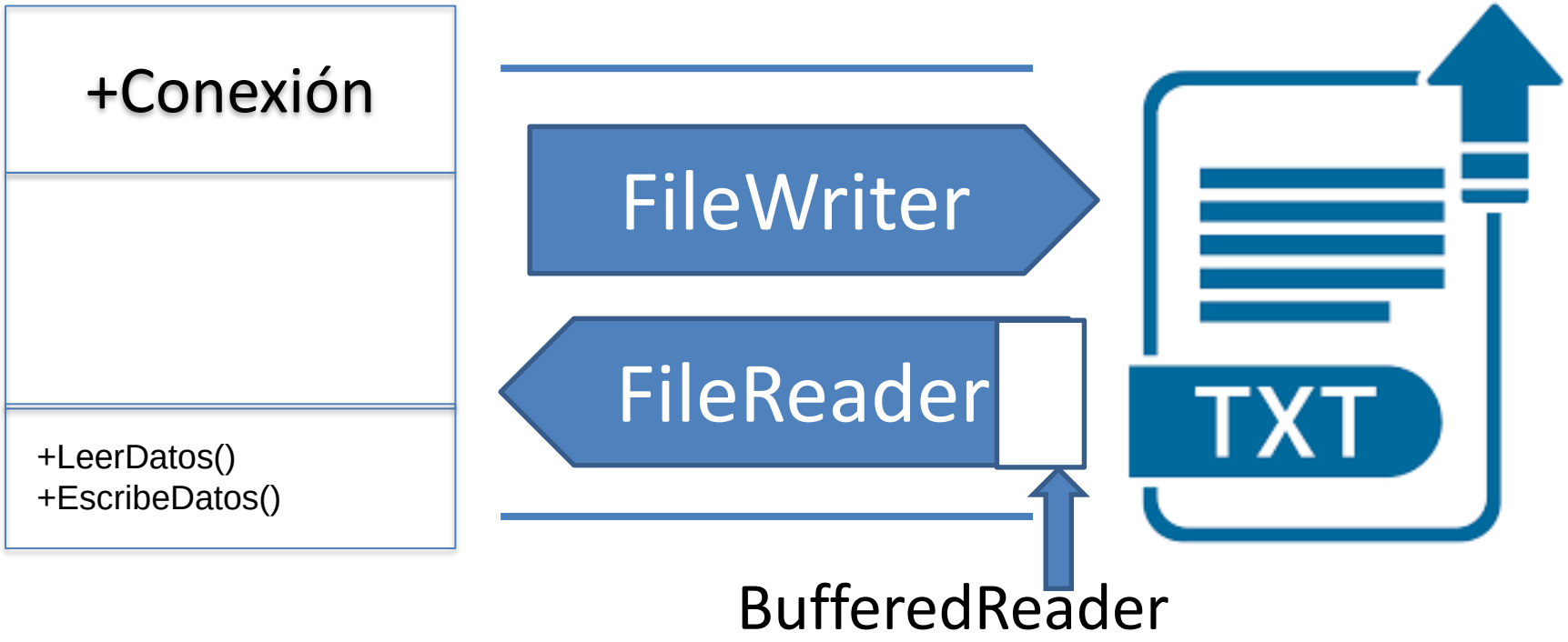
Clases Java Reader y Writer



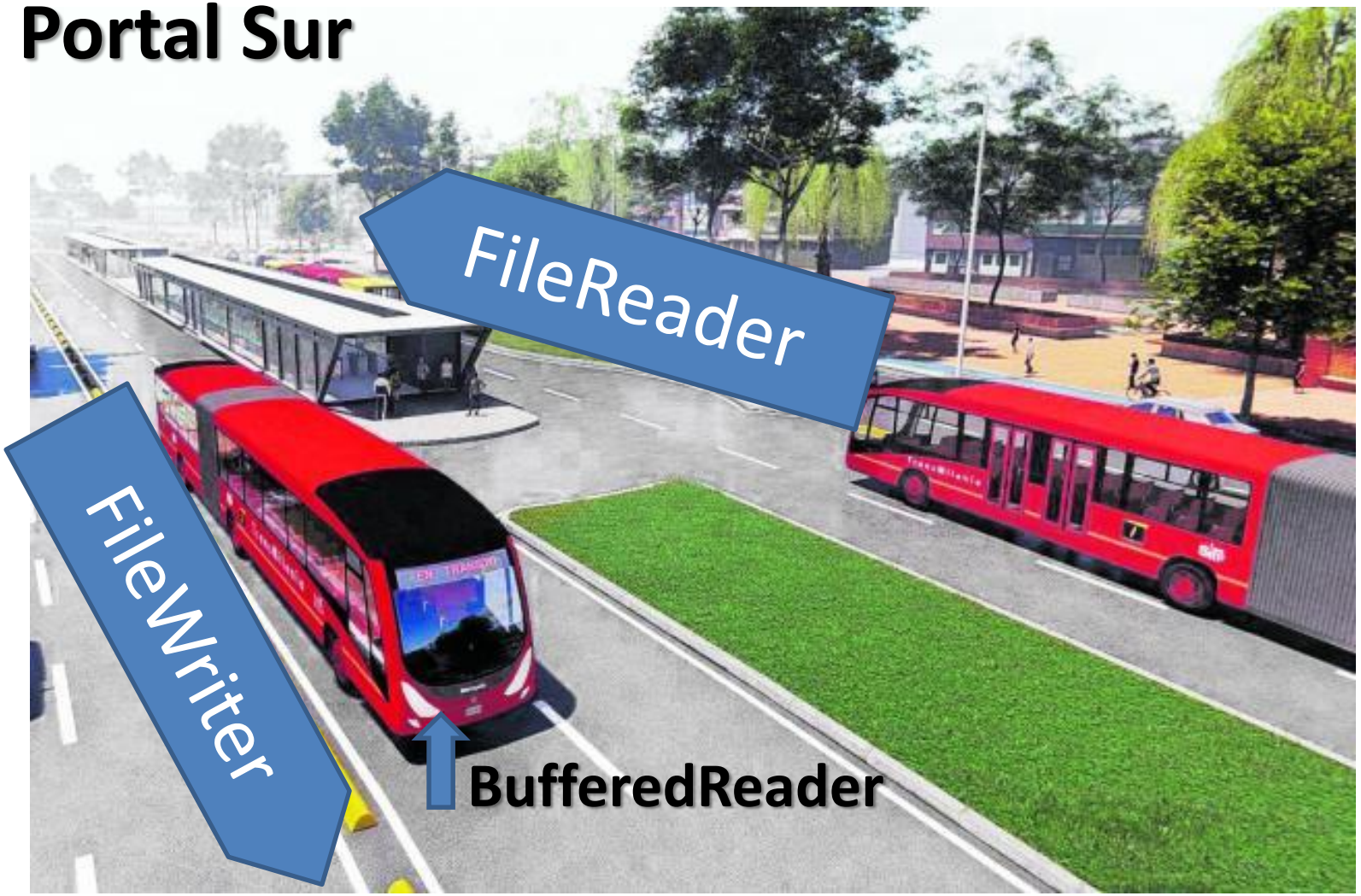
Son clases que permiten realizar las acciones de lectura/escritura

Persistencia con Archivos

Los Flujos permiten enviar y obtener los datos del archivo y los Buffered son contenedores para almecenarlos.



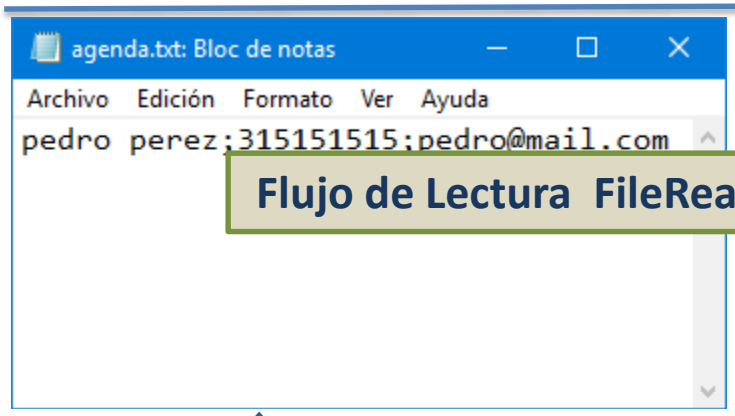
Ejemplo Transmilenio



Clases Java Reader y Writer

Clases	Función que realizan
FileReader, FileWriter, FileInputStream y FileOutputStream	Son las clases que leen y escriben en archivos de disco.
StringReader, StringWriter, CharArrayReader, CharArrayWriter, ByteArrayInputStream, ByteArrayOutputStream, StringBufferInputStream	Estas clases tienen en común que se comunican con la memoria del ordenador. En vez de acceder del modo habitual al contenido de un String, por ejemplo, lo leen como si llegara carácter a carácter. Son útiles cuando se busca un modo general e idéntico de tratar con todos los dispositivos que maneja un programa.
PipedReader, PipedWriter, PipedInputStream, PipedOutputStream	Se utilizan como un "tubo" o conexión bilateral para transmisión de datos. Por ejemplo, en un programa con dos threads pueden permitir la comunicación entre ellos. Un thread tiene el objeto PipedReader y el otro el PipedWriter. Si los streams están
	conectados, lo que se escriba en el PipedWriter queda disponible para que se lea del PipedReader. Tambíen puede comunicar a dos programas distintos.

Flujos de Entrada/Salida en archivos

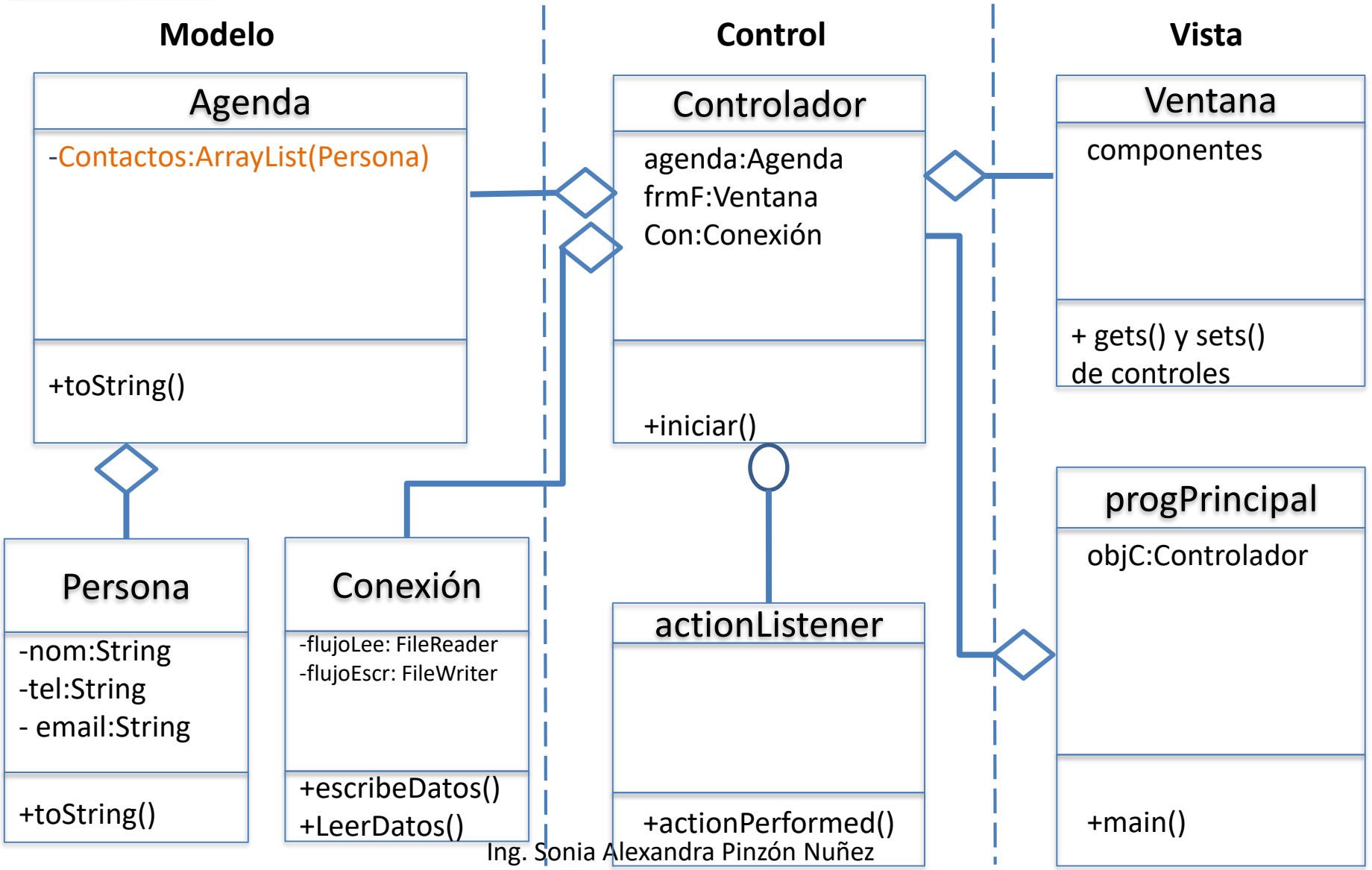


Flujo de Lectura FileReader

Flujo de Escritura FileWriter

```
public String leerDatos() throws IOException{
    this.flujoLee = new FileReader("agenda.txt");
    bufEntrada = new BufferedReader(flujoLee);
    String datos="";
    String linea = this.bufEntrada.readLine();
    while (linea != null) {
        datos+=linea+"\n";
    linea = bufEntrada.readLine();
    }
    bufEntrada.close();
    return datos;
}
public void EscribeDatos(String datos) throws IOException{
    flujoEscr= new FileWriter("agenda.txt",true);
    bufSalida = new PrintWriter(flujoEscr);
    bufSalida.println(datos);
    bufSalida.close();
}
```

Ejemplo Agenda: Diagrama de Clases



Clase Persona - modelo

```
package modelo;
public class Persona {
  private String nom,tel,email;
    public Persona(String nom, String tel, String email) {...5 lines }
   public Persona() {
        this.nom = "";
        this.tel = "";
        this.email = "";
    }
    public String datos() {
        return nom + ";"+tel + ";"+email ;
    }
    @Override
    public String toString() {
        return " Nombre: " + nom + "\n Teléfono" + tel + "\n Email=" + email ;
    }
    public String getNom() {...3 lines }
    public void setNom(String nom) {...3 lines }
    public String getTel() {...3 lines }
    public void setTel(String tel) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }
}
```

Método usado para enviar los datos al archivo, retorna en una línea los datos separados por punto y coma.

Clase Agenda - modelo

```
package modelo;
import java.util.ArrayList;
public class Agenda {
    ArrayList <Persona> contactos;
    public Agenda(ArrayList<Persona> contactos) {
        this.contactos = contactos;
    }
     public Agenda() {
        this.contactos = new <Persona> ArrayList();
    }
    public ArrayList<Persona> getContactos() {
        return contactos;
    }
    public void setContactos(ArrayList<Persona> contactos) {
        this.contactos = contactos;
    }
    @Override
    public String toString() {
        String datos="";
        for (int i= 0; i< contactos.size();i++)
           datos+= contactos.get(i).toString()+ "\n";
        return datos;
    }
}
```

Clase Conexion - modelo

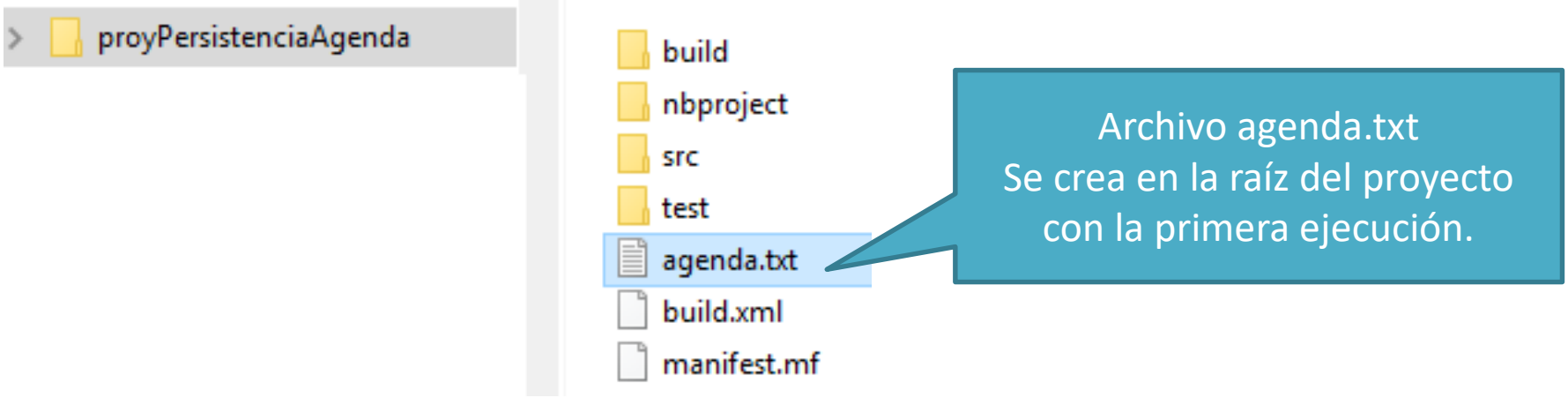
```
public class Conexion {
 protected BufferedReader bufEntrada ;
 protected FileReader flujoLee;
 protected FileWriter flujoEscr;
 protected PrintWriter bufSalida;
public Conexion(BufferedReader entrada, FileReader archLee, FileWriter archEscr,
public Conexion() throws IOException {
    this.bufEntrada = null;
   this.flujoLee = null;
   this.flujoEscr = null;
    this.bufSalida = null;
}
public String leerDatos() throws IOException{
    this.flujoLee = new FileReader("agenda.txt");
    bufEntrada = new BufferedReader(flujoLee);
    String datos="";
    String linea = this.bufEntrada.readLine();
    while (linea != null) {
        datos+=linea+"\n";
    linea = bufEntrada.readLine();
    }
    bufEntrada.close();
    return datos;
}
public void EscribeDatos(String datos) throws IOException{
    flujoEscr= new FileWriter("agenda.txt",true);
    bufSalida = new PrintWriter(flujoEscr);
    bufSalida.println(datos);
    bufSalida.close();
}
}
```

Flujo de Lectura

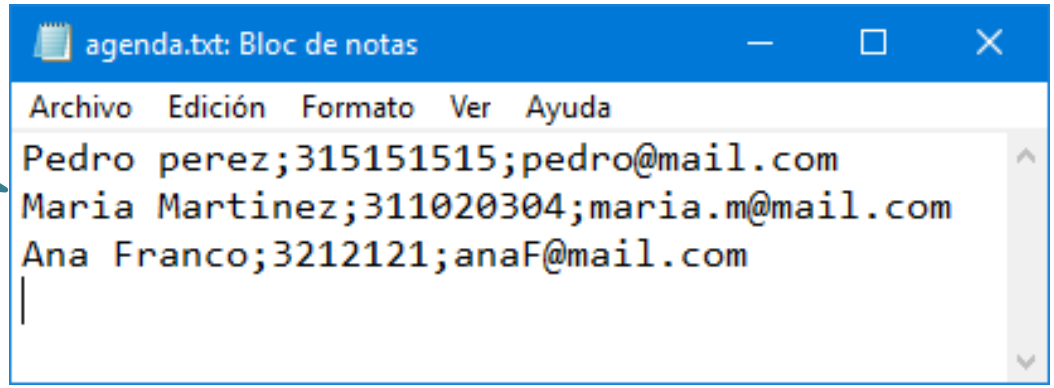
Flujo de Escritura

Archivo agenda.txt

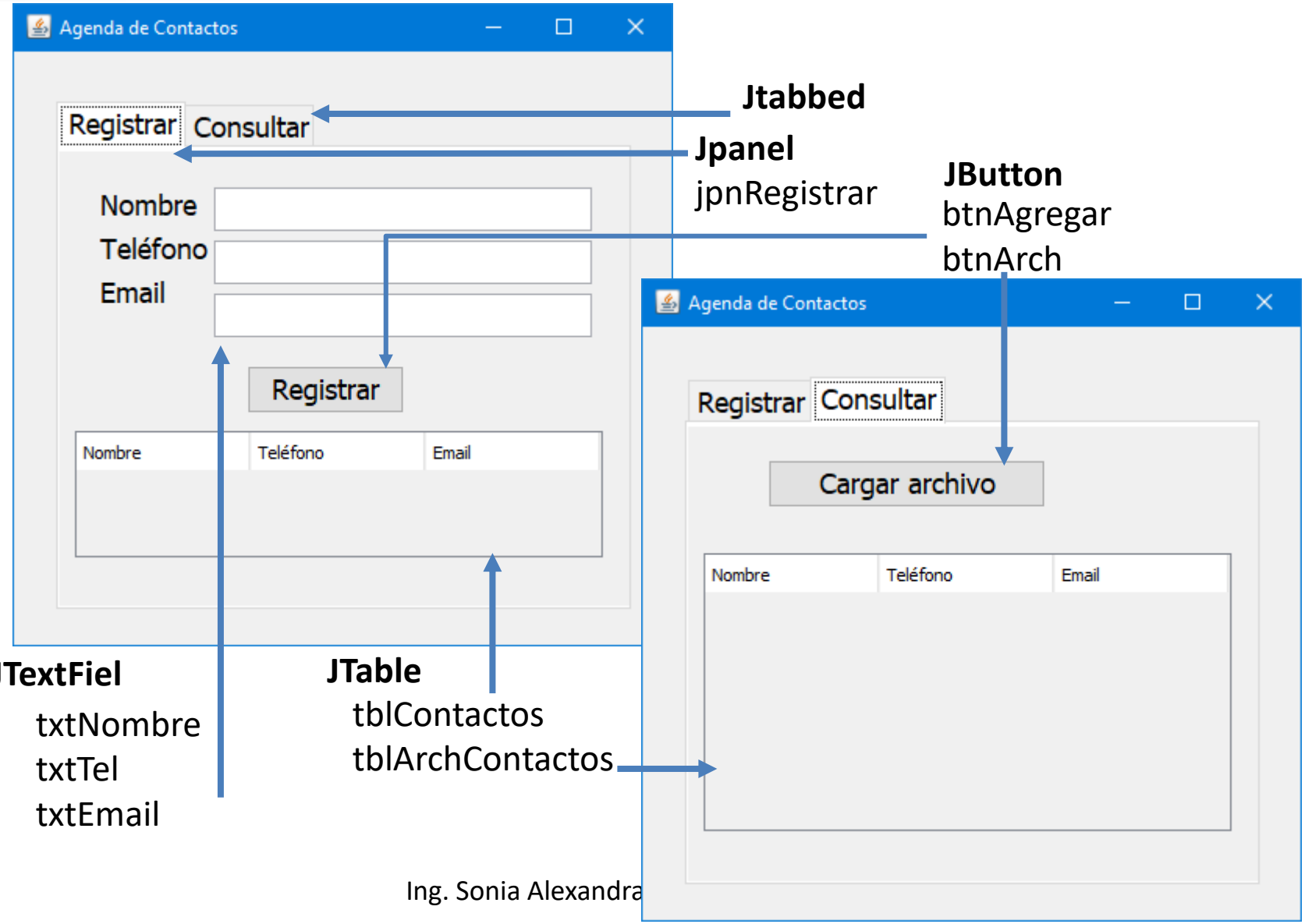
Archivo agenda.txt



Cada campo está sepado por punto y coma (;)



Clase Ventana - vista



Clase Controlador I

Se sugiere agregar la clausula trhows al método

```
import javax.swing.table.DefaultTableModel;
import modelo.Agenda;
import modelo.Conexion;
import modelo.Persona;
import vista.Ventana;
public class Controlador implements ActionListener{
  Agenda agenda;
  Ventana frmV;
  Conexion con;
    public Controlador(Agenda agenda, Ventana objV) throws IOException  {
        this.agenda = agenda;
        this.frmV = objV;
        this.con= new Conexion();
        frmV.getBtnAgregar().addActionListener(this);
        frmV.getBtnArch().addActionListener(this);
    }
    public Controlador() throws IOException {
        this.agenda = new Agenda();
        this.frmV = new Ventana();
        this.con= new Conexion();
        frmV.getBtnAgregar().addActionListener(this);
        frmV.getBtnArch().addActionListener(this);
    }
    public void iniciar(){
     frmV.setTitle("Agenda Contactos con Archivos");
     frmV.setVisible(true);
    }
```

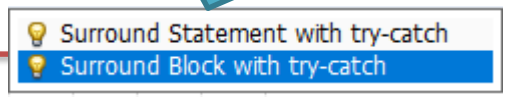
Add throws clause for java.io.IOException
Surround Statement with try-catch
Surround Block with try-catch

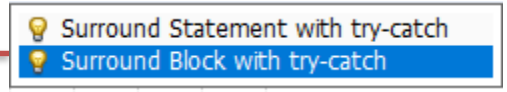
Add throws clause for java.io.IOException
Surround Statement with try-catch
Surround Block with try-catch

Clase Controlador II

```
public void actionPerformed(ActionEvent ae) {
    if(ae.getSource().equals(frmV.getBtnAgregar())){
        try {
            Persona contacto= new Persona(
                    frmV.getTxtNom().getText(),
                    frmV.getTxtTel().getText(),
                    frmV.getTxtCorreo().getText());
            agenda.getListaC().add(contacto);
            con.EscribeDatos(contacto.datos());
            JOptionPane.showMessageDialog(frmV,
                    "Datos registrados..\n"+contacto.toString());
            iniciarControles(frmV.getJpnRegistro().getComponents());
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(frmV,
                    "Error al acceder al archivo...\n");
        }
    }
    if(ae.getSource().equals(frmV.getBtnMostrarArch())){
        try {
            iniciarTabla(frmV.getTblContactosArchivo());
            String Agenda[]=con.leerDatos().split("\n");
            for (String registro : Agenda) {
             String[] datos= registro.split(";");
             Persona contacto= new Persona(datos[0],datos[1],datos[2]);
             agregarContactoTabla(contacto,frmV.getTblContactosArchivo());
            }
        } catch (IOException ex) {
             JOptionPane.showMessageDialog(frmV,
                    "Error al acceder al archivo...\n");
        }
    }
}
```

Se sugiere agregar dentro de un bloque try-catch





Clase Controlador III

```
public void agregarContactos(Persona cont,JTable tabla){
 Object datos[]={cont.getNom(), cont.getTel(),cont.getEmail()};
  DefaultTableModel plantilla =(DefaultTableModel) tabla.getModel();
  plantilla.addRow(datos);
}
public void llenarLista(String contactos, JTable tabla){
    String Agenda[]=contactos.split("\n");
     for(int i=0;i<Agenda.length;i++){
     String datos[]= Agenda[i].split(";");
     System.out.println("registro "+Agenda[i].toString());
      Persona contacto= new Persona(
                              datos[0],
                              datos[1],
                              datos[2]);
     agregarContactoTabla(contacto,tabla);
     }
}
public void iniciarControles( Component[] controles){
  for (Object control : controles) {
      if(control instanceof JPanel){
          iniciarControles(((JPanel)control).getComponents());
        }else if(control instanceof JTextField){
        ((JTextField) control).setText("");
      }
  }
}
```

agenda.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
Pedro perez;315151515;pedro@mail.com
Maria Martinez;311020304;maria.m@mail.com
Ana Franco;3212121;anaF@mail.com
```

Agenda

Pedro Perez;3152151515;pedro@gmail.com
Maria Martinez;3112565656;maria@gmail.com
Ana Franco;3212121;anaF@yahoo.com

datos

Pedro Perez	3152151515	pedro@gmail.com
0	1	2

contacto

nom="Pedro Perez"
tel="3152151515"
email= "pedro@gmail.com"

tabla

Nombre	Teléfono	Correo
Pedro Perez	3152151515	pp@gmail.com
Maria Martinez	3112565656	mari@gmail.com
Ana Franco	3212121	anaF@mail.com

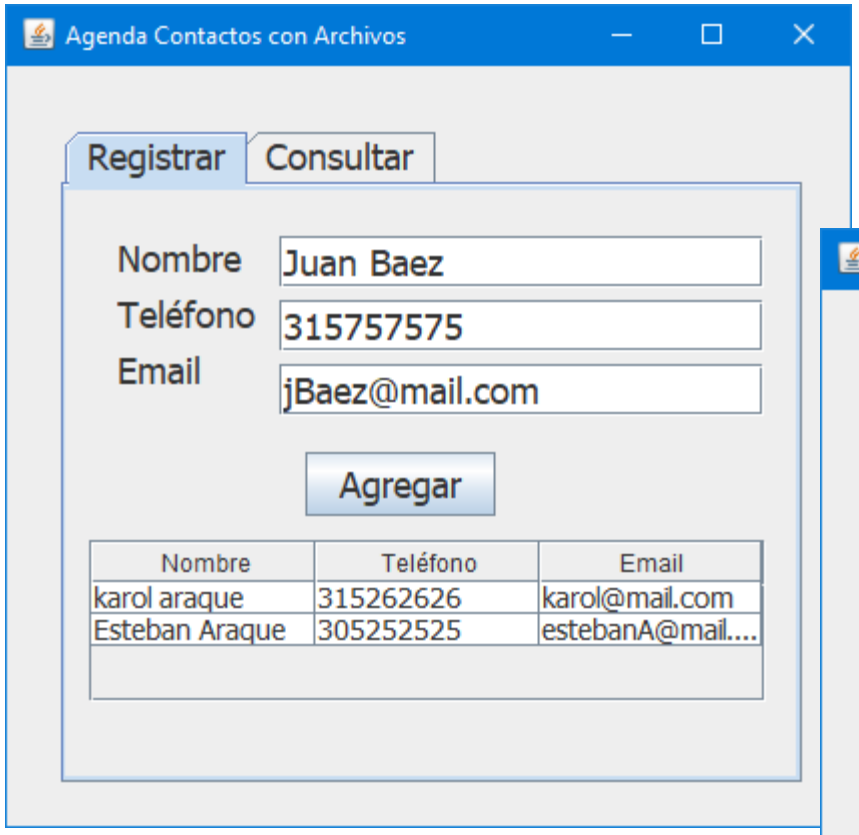
Clase progPincipal

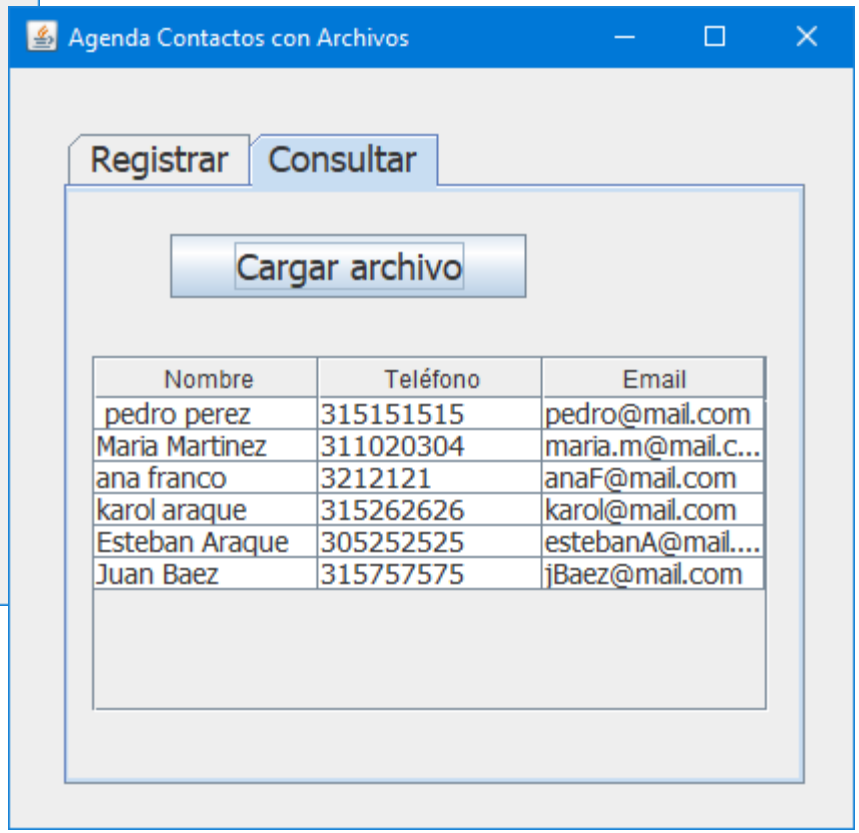
```
package proypersistenciaagenda;
import control.Controlador;
/**...4 lines */
public class ProyPersistenciaAgenda {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Controlador objC= new Controlador();
```

Add throws clause for java.io.IOException
Surround Statement with try-catch
Surround Block with try-catch

```
}
```

Ejecución


Agenda Contactos con Archivos
Registrar
Consultar
Nombre
Juan Baez
Teléfono
315757575
Email
jBaez@mail.com
Agregar
Nombre
Teléfono
Email
karol araque
315262626
karol@mail.com
Esteban Araque
305252525
estebanA@mail....


Agenda Contactos con Archivos
Registrar
Consultar
Cargar archivo
Nombre
Teléfono
Email
pedro perez
315151515
pedro@mail.com
Maria Martinez
311020304
maria.m@mail.c...
ana franco
3212121
anaF@mail.com
karol araque
315262626
karol@mail.com
Esteban Araque
305252525
estebanA@mail....
Juan Baez
315757575
jBaez@mail.com

Bibliografía

- Deitel y Deitel. Programación Java. Editorial Mc Graw Hill.
- Sonia Pinzón .Curso Programación Multinivel Moodle.