7.- Datos permanentes en archivos secuenciales.

7.1.- Memoria volátil.

Todas las variables declaradas en un programa se localizan en memoria volátil, se le llama así porque al detener la ejecución de un programa toda la información almacenada en esta memoria desaparece.
Normalmente es necesario conservar información de una ejecución de un programa a otra, la información que deseamos conservar de una corrida a otra de un programa debe ser almacenada en memoria permanente.

7.2.- La necesidad de archivos para conservar el valor de las variables.

La memoria permanente está constituida por medios de almacenamiento tales como discos rígidos, memorias USB y otros dispositivos. Para guardar algo en éste tipo de dispositivos es necesario almacenarlos en archivos o ficheros.
Debido a lo anterior, si un programa requiere guardar sus variables puede hacerlo almacenándolas en un archivo ya que posteriormente podrá utilizar dicho archivo para leer de ahí la información que previamente almaceno el mismo programa u otro.

7.3.- Selección de archivos y directorios.

Para poder manipular archivos es necesario poder seleccionarlos, el paquete inovaProg permite con facilidad hacer esta tarea.

Cuando se va a leer un archivo es posible seleccionarlo utilizando la siguiente instrucción:

```
nombreArchivoSeleccionado=Ip.seleccionaArchivo(new File("."));
```

La expresión `new File(".")` indica iniciar la selección en el directorio en el que esta corriendo el programa, pero el punto "." Puede ser cambiado por una ruta por ejemplo "C\ejemplo\"

Cuándo se va a crear un archivo es necesario especificar el directorio en el cual lo vamos a crear y para ello hay que seleccionarlo con la siguiente instrucción:

```
nombreDirectorioSeleccionado=Ip.seleccionaDirectorio(new File("."));
```

El ejemplo adjunto 700OperacionesDeSeleccionDeArchivosYDirectorios.java muestra todas las variantes que inovaProg proporciona para estas tareas.

7.4.- Creación, lectura y actualización de un archivo.

Un archivo se crea escribiendo información en él, posteriormente puede ser leído e incluso cambiado o actualizado.
El siguiente programa crea un archivo

```
import java.io.*;
```

```
class CreacionDeArchivos {
    public static void main(String[] args) {
    {
        boolean r;
        String nombreArchivo="";
        String rutaParaGuardar="";
        int numero=0;
        String nombre="";
        float peso=0;
        nombreArchivo=Ip.wlee(
            "Teclear el nombre del archivo ",nombreArchivo);
        nombreArchivo += ".txt"; //Se asume archivo de texto
        rutaParaGuardar=Ip.seleccionaDirectorio(new File("."),
        "Favor de seleccionar un directorio para guardar el archivo");
        nombreArchivo = rutaParaGuardar + nombreArchivo;
        if (Ip.existeArchivo(nombreArchivo))
          {r=Ip.wleeRespuesta(
            "El archivo ya existe desea reemplazarlo?");
             if(!r)
                  {Ip.wescribeMensaje("Operacion abortada");
                    return;}
            }
        Ip.abreArchSalidaNuevo(nombreArchivo);
        r=true;
        numero=0;
        while(r)
        {
            nombre=Ip.wlee("Nombre ",nombre);
            peso=Ip.wlee("Peso   ",peso);
            Ip.escribeEnArch(numero);
            Ip.escribeEnArch(nombre);
            Ip.escribeEnArch(peso);
            numero++;
            r=Ip.wleeRespuesta("Desea agregar otro registro?");
        }
        Ip.cierraArchSalida();
        Ip.wescribeMensaje("Termina la creacion del archivo");
    }
}}
```

Este código se encuentra en el ejemplo 702CreacionDeArchivos.java.
La primera parte del programa solicita el nombre del archivo y selecciona el directorio dónde se va a crear.
Posteriormente el archivo se abre de salida y se escribe en el las variables tecleadas por el usuario, finalmente el archivo se cierra. Podemos mediante el explorador del sistema operativo comprobar la existencia del archivo.

El siguiente programa contenido en el ejemplo 703LecturaDeArchivos.java lee el archivo anterior:

```
import java.io.*;
class LecturaDeArchivos {
    public static void main(String[] args) {
    {  boolean r;
        String nombreArchivo="";
        int numero=0;
        String nombre="";
        float peso=0;
```

```
        String datoLeido="";
        String Mensaje="Lista de Pesos \nGuardada en el archivo \n";
        nombreArchivo=Ip.seleccionaArchivo(new File("."),
            "Favor de seleccionar un archivo para su lectura","txt");
        Mensaje += nombreArchivo + "\n";
        Ip.abreArchEntrada(nombreArchivo);
        datoLeido = Ip.leeDeArch();
        while(!datoLeido.equals(""))
        {   numero = Ip.aEntero(datoLeido);
            nombre = Ip.leeDeArch();
            peso   = Ip.aDecimal(Ip.leeDeArch());
            Mensaje +=  "\n" + numero + " " +
                              nombre + " " +
                              peso   + " ";
            datoLeido = Ip.leeDeArch();
        }
        Ip.cierraArchEntrada();
        Ip.wescribeMensaje(Mensaje);
        }
}}
```

En la primera parte se selecciónale archivo y se abre de entrada, en la segunda parte se leen las variables y se convierten al tipo interno que el programa utiliza para ellas ya que cunado una variable se escribe en un archivo automáticamente se convierte en una variable de cadena de caracteres de tal manera que al recuperarla es necesario convertirla al tipo original. Después del ciclo de lectura el archivo se cierra, note que las variables se leen exactamente en el mismo orden en el que se escribieron.

Finalmente tenemos la actualización del archivo que se encuentra en el ejemplo 704ActualizacionDeArchivos.java

```
import java.io.*;
class ActualizacionDeArchivos {
    public static void main(String[] args) {
    {   boolean r;
        String nombreArchivo="";
        int numero=0;
        String nombre="";
        float peso=0;
        nombreArchivo=Ip.seleccionaArchivo(new File("."),
            "Favor de seleccionar un archivo para su lectura","txt");
        Ip.abreArchSalidaYaExistente(nombreArchivo);
```

```
        r=true;
        numero=Ip.wlee(
        "Folio de inicio para agregar los registros nuevos: ",numero);
        while(r)
        {   nombre=Ip.wlee("Nombre ",nombre);
            peso=Ip.wlee("Peso   ",peso);
            Ip.escribeEnArch(numero);
            Ip.escribeEnArch(nombre);
            Ip.escribeEnArch(peso);
            numero++;
            r=Ip.wleeRespuesta("Desea agregar otro registro?");
        }
        Ip.cierraArchSalida();
        Ip.wescribeMensaje("Termina la actualizacion del archivo");
    }
}}
```

En la primera parte se selecciona el archivo, luego se abre de salida ya existente, esto permite que la información que se le añada al archivo se agregue al final. El ciclo permite agregar datos al final del archivo y finalmente se cierra el archivo.

Note que la instrucción `import java.io.*;` al principio de los programas que utilizan archivos es obligatoria.

7.5.- Selección, comprobación de existencia y eliminación de un archivo.

Para borrar un archivo es necesario seleccionarlo y posteriormente eliminarlo, el ejemplo 701OperacionesParaBorrarArchivos.java que discutiremos a continuación muestra en detalle las instrucciones.

```
import java.io.*;
class OperacionesParaBorrarArchivos {
    public static void main(String[] args) {
        String nombreArchivoSeleccionado;
        boolean r=false;
        nombreArchivoSeleccionado=Ip.seleccionaArchivo(new File("."));
        Ip.wescribeMensaje(
            "Archivo seleccionado ("+nombreArchivoSeleccionado+")");
        r=Ip.wleeRespuesta(
            "Desea eliminar el archivo :\n"+nombreArchivoSeleccionado +
            "\ndefinitivamente?");
        if (r)
              {Ip.borraArchivo(nombreArchivoSeleccionado);}
          if (!Ip.existeArchivo(nombreArchivoSeleccionado))
            {Ip.wescribeMensaje(
            "El archivo :\n"+nombreArchivoSeleccionado +
             "\nHa sido eliminado ");}
          else
                  {Ip.wescribeMensaje(
```

```
                    "El archivo :\n"+nombreArchivoSeleccionado +
                     "\nNo ha sido eliminado ");}
        }
}
```

En la primera parte se selecciona el archivo a eliminar, se pide una confirmación al usuario antes de eliminarlo y se comprueba la operación.

7.6.- Actividades

Compile y ejecute los ejemplos adjuntos agregando datos a los archivos.

7.7.- Ejercicio propuesto:

Escriba un programa que solicite una lista de artículos y que para cada artículo lea su código, nombre, existencia y precio, esta información la almacenará en un archvio de salida, posteriormente escriba otro programa que lea y muestre el archivo capturado en el primer programa.