SEMANA 06

HERENCIA

Herencia con archivos de texto, sobrecarga de métodos

CAPACIDAD EN PROCESO:
▪ Aplica el mecanismo de herencia utilizando archivos de texto como medio de almacenamiento. ▪ Aplica sobrecarga de métodos en una jerarquía de clases.

Herencia con archivos de texto

Hasta ahora, las clases **ArrayList** y **LinkedList** nos ha dado 2 ventajas inobjetables: tamaño ilimitado y funcionalidad que facilitan la administración de colecciones de objetos con código simplificado. Sin embargo, todo lo que se guarda en los objetos **ArrayList y LinkedList** permanece en memoria sólo mientras se esté ejecutando el programa, si lo ejecutamos nuevamente, tendremos que ingresar todos los datos nuevamente!.

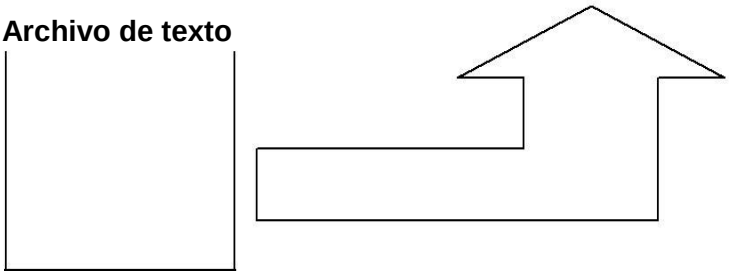
Afortunadamente éste problema se resuelve utilizando archivos de texto para guardar permanentemente los datos en el disco.

Vamos a tomar como clase padre a una clase administradora, heredamos toda su funcionalidad y a la clase hija le agregamos la funcionalidad que permita grabar en el archivo y leer desde el archivo.

Proceso gráfico de cómo se realiza la **lectura** de los datos dese el archivo de texto hacia la lista.

ArrayList o LinkedList

Archivo de texto



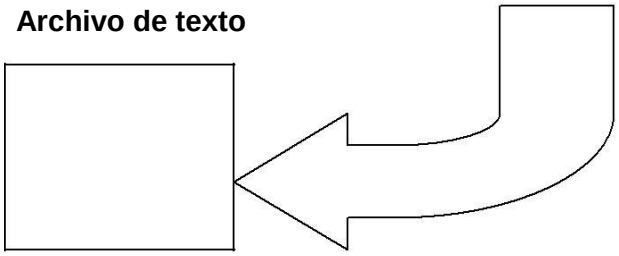
- Lee una línea del archivo
- Descompone los datos
- Crea nuevo objeto
- Lo guarda en el ArrayList o LinkedList

Para leer los datos del archivo hacia el objeto **ArrayList o LinkedList** utilizaremos las siguientes clases: **FileReader**, **BufferedReader**, **StringTokenizer**.

Proceso gráfico de cómo se realiza la **grabación** de los datos dese la lista hacia el archivo de texto.

ArrayList o LinkedList

Archivo de texto



- Obtiene un objeto del ArrayList o LinkedList
- Concatena los datos en una cadena
- Lo guarda en el archivo

Para grabar los datos en el archivo desde el objeto **ArrayList o LinkedList** utilizaremos las siguientes clases: **FileWriter**, **PrintWriter**.

Ejemplo 1

Considere la existencia de una clase administradora de nombre **ListaPilaTVH** desarrollada anteriormente. Aplique herencia y desarrolle la clase hija **ArchivoTVH** donde se consideren métodos para leer y para grabar en un archivo de texto.

```
public class ArchivoTVH extends ListaPilaTVH {
    // atributos protegidos
    protected String nombre;

    // constructor
    public ArchivoTVH(String nombre){
      super();
      this.nombre=nombre;
      lee();

    }
```

```
    // métodos que operan un archivo de
    texto public void lee(){
      try{
            FileReader fr = new
            FileReader(nombre); BufferedReader br
            = new BufferedReader(fr); String
            linea=br.readLine();
            while(linea != null){
                  StringTokenizer st = new
                  StringTokenizer(linea,"/");
                  String serie=st.nextToken();
                  int marca=Integer.parseInt(st.nextToken());
                  int tamaño=Integer.parseInt(st.nextToken());
                  double
                  precio=Double.parseDouble(st.nextToken());
                  int origen=Integer.parseInt(st.nextToken());
                  int
                  tecnologia=Integer.parseInt(st.nextToken());
                  agrega(serie,marca,tamaño,precio,origen,tecnolo
                  gia); linea= br.readLine();
            }
            br.close();
      }catch(Exception ex){ }
}
public void
      graba(){
      try{
            FileWriter fw = new FileWriter(nombre);
            PrintWriter pw= new
            PrintWriter(fw); for(TVH t :
            pila){
                  pw.println(t.getSerie()+"
                  /"+
                              t.getMarca()+"/"+
                              t.getTamaño()+"/"+
                              t.getPrecio()+"/"+
                              t.getOrigen()+"/"+
                              t.getTecnologia());
            }
            pw.close();
```

```
            }catch(Exception ex){}
      }
      }// fin de la clase
```

Considere el mismo diseño de GUI

- Declare y cree un objeto de la clase ArchivoTVH en la clase de la GUI
- La grabación de datos debe hacerse en cada botón que lo requiera.
- Programe la acción de los botones.

Ejemplo 2

Considere la existencia de una clase administradora de nombre **ListaColaPersonasH** desarrollada anteriormente. Aplique herencia y desarrolle la clase hija **ArchivoPersonasH** donde se consideren métodos para leer y para grabar en un archivo de texto.

Considere el mismo diseño de GUI:

- Declare y cree un objeto de la clase **ArchivoTVH** en la clase de la GUI
- La grabación de datos debe hacerse en cada botón que lo requiera.
- Programe la acción de los botones.

Sobrecarga de métodos

El concepto de sobrecarga va orientado a que un método puede repetirse con el mismo nombre pero con la condición de que se diferencien por los parámetros: en número y/o en cantidad.

Por ejemplo, si existiera un método que suma dos números enteros que recibe como parámetros, tendría la siguiente declaración:

```
public int suma(int a, int b)
```

Ahora, si quisiéramos sumar dos números reales, no tenemos porqué cambiar el nombre al método, porque su tarea va a ser la misma: sumar. Entonces aplicamos sobrecarga para declararlo así:

```
public double suma(double a, double b)
```

Nuevamente, si ahora quisiéramos sumar 3 números enteros, aplicamos sobrecarga así:

```
public int suma(int a, int b, int c)
```

Este concepto de sobrecarga también es aplicable en los **constructores** de las clases.

La ventaja de éste concepto de sobrecarga es que flexibiliza la funcionalidad de las clases, ofreciendo varias alternativas de uso de una misma actividad: los métodos y los constructores.

Por ejemplo, en una clase administradora se puede considerar métodos **sobrecargados** para agregar objetos de diferente manera:

```
public void agrega(TVH t){
   a.add(t);
}
public void agrega(String serie, int marca, int tamaño,
              double precio, int origen, int tecnologia){
a.add(new TVH(serie,marca,tamaño,precio,origen,tecnologia));
}
```

Asimismo, se puede considerer métodos sobrecargados para eliminar objetos de diferente manera:

```
public void elimina(int i){
    a.remove(i);
 }
 public void elimina(String serie){
    if(busca(serie)!=null)
        a.remove(busca(serie));
 }
 public void elimina(TVH t){
    a.remove(t);
 }
```

- Incorpore estos métodos sobrecargados en las clases administradoras.

GUIA DE LABORATORIO 6

Herencia con archivos de texto

Ejercicio 1

Cree un proyecto nuevo de nombre **P06E01**. Cree un paquete nuevo de nombre **controlador**. Considere la existencia de una clase administradora de nombre **ListaPilaTVH** desarrollada anteriormente. Aplique herencia y desarrolle la clase hija **ArchivoTVH** donde se consideren métodos para leer y para grabar en un archivo de texto.

```
public class ArchivoTVH extends ListaPilaTVH {
    // atributos protegidos
    protected String nombre;

    // constructor
    public ArchivoTVH(String nombre){
      super();
      this.nombre=nombre;
      lee();

    }

    // métodos que operan un archivo de
    texto public void lee(){
      try{
            FileReader fr = new
            FileReader(nombre); BufferedReader br
            = new BufferedReader(fr); String
            linea=br.readLine();
            while(linea != null){
                  StringTokenizer st = new
                  StringTokenizer(linea,"/");
                  String serie=st.nextToken();
                  int marca=Integer.parseInt(st.nextToken());
                  int tamaño=Integer.parseInt(st.nextToken());
                  double
                  precio=Double.parseDouble(st.nextToken());
                  int origen=Integer.parseInt(st.nextToken());
```

```
                    int
                    tecnologia=Integer.parseInt(st.nextToken());
                    agrega(serie,marca,tamaño,precio,origen,tecnolo
                    gia); linea= br.readLine();
              }
              br.close();
        }catch(Exception ex){ }
}
public void
        graba(){
        try{
              FileWriter fw = new FileWriter(nombre);
              PrintWriter pw= new
              PrintWriter(fw); for(TVH t :
              pila){
                    pw.println(t.getSerie()+"
                    /"+
                                t.getMarca()+"/"+
                                t.getTamaño()+"/"+
                                t.getPrecio()+"/"+
                                t.getOrigen()+"/"+
                                t.getTecnologia());
              }
              pw.close();
        }catch(Exception ex){}
}
}// fin de la clase
```

Cree un paquete nuevo de nombre **vista** y considere el mismo diseño de GUI e implemente, lo necesario para la programación de los botones.

En el paquete **vista** diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

Ejercicio 2

Cree un proyecto nuevo de nombre **P06E02**. Cree un paquete nuevo de nombre **controlador**. Considere la existencia de una clase administradora de nombre **ListaColaPersonasH** desarrollada anteriormente. Aplique herencia y desarrolle la clase hija **ArchivoPersonasH** donde se consideren métodos para leer y para grabar en un archivo de texto.

Considere el mismo diseño de GUI

Cree un paquete nuevo de nombre **vista** e implemente, en la interfaz gráfica de usuario (GUI), lo necesario para la programación de los botones.

En el paquete **vista** diseñe la clase **Principal** (Frame) y haga funcionar su aplicación.

- Incorpore métodos sobrecargados en las clases administradoras.