Manejo de archivos de acceso directo en Java.

Alumno.java

```java
package uam.edoo.archivos.directo.clases;

public class Alumno {

	public static final int TAMANIO = 200;

	private String nombre;
	private String licenciatura;
	private int edad;
	private String matricula;



	public int getEdad() {
		return edad;
	}
	public void setEdad(int edad) {
		this.edad = edad;
	}
	public String getMatricula() {
		return matricula;
	}
	public void setMatricula(String matricula) {
		this.matricula = matricula;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getLicenciatura() {
		return licenciatura;
	}
	public void setLicenciatura(String licenciatura) {
		this.licenciatura = licenciatura;
	}

	public String toString(){
		String mensaje = "";
		mensaje = this.nombre+"\n"+this.licenciatura + "\n"+ this.matricula +
				"\n"+ this.edad;
		return mensaje;
	}


}
```

OperacionesArchivo.java

```
package uam.edoo.archivos.directo.operaciones;

import java.io.IOException;
import java.io.RandomAccessFile;

import uam.edoo.archivos.directo.clases.Alumno;

public class OperacionesArchivo {

      public RandomAccessFile archivo;

      /*
       * Se abre el archivo recibiendo un nombre
       */
       public void abrirArchivo( String nombreArchivo) throws IOException{
            archivo = new RandomAccessFile( nombreArchivo, "rw" );
       }

       /*
        * Se cierra el archivo
        */
       public void cerrarArchivo() throws IOException{
            if ( archivo != null )
               archivo.close();
         }


       /*
        * Se lee un registro para llenar un objeto alumno
        * a partir de una posición dada
        */
       public Alumno obtenerRegistro( int posicion )
                    throws IllegalArgumentException, NumberFormatException,
IOException{
             Alumno alumno = new Alumno();

                  if ( posicion < 1 || posicion > 100 )
                     throw new IllegalArgumentException( "Fuera de rango" );

                  // buscar registro apropiado en el archivo
                  archivo.seek( ( posicion  ) * Alumno.TAMANIO );

                    alumno.setEdad(archivo.readInt());
                  alumno.setNombre(leerCadena( archivo ));
                  alumno.setLicenciatura(leerCadena( archivo ));
                  alumno.setMatricula(leerCadena( archivo ));

                  return alumno;

               }
```

```java
/*
 * Almacena un alumno en la posición dada
 */
public void nuevoRegistro( int posicion, Alumno alumno)
           throws IllegalArgumentException, IOException
        {

      // buscar registro apropiado en el archivo
           archivo.seek( ( posicion  ) * Alumno.TAMANIO );

           archivo.writeInt( alumno.getEdad());
           escribirCadena(archivo, alumno.getNombre() );
           escribirCadena(archivo, alumno.getLicenciatura() );
           escribirCadena(archivo, alumno.getMatricula());

        }


/*
   *
   *asegurarse que la cadena sea de la longitud apropiada
   *remplazando los caracteres que falten de la longitud con
   *espacios en blanco
   */

  private String leerCadena( RandomAccessFile archivo ) throws IOException
  {
     char nombre[] = new char[ 50 ], temp;

     for ( int cuenta = 0; cuenta < nombre.length; cuenta++ ) {
        temp = archivo.readChar();
        nombre[ cuenta ] = temp;
     }

     return new String( nombre ).replace( '\0', ' ' );
  }


  /*
   * Elimina un registro del archivo
   */
  public void eliminarRegistro( int posicion )
                throws IllegalArgumentException, IOException {

                // crear un registro en blanco para escribir en el archivo
           Alumno alumno  = new Alumno();

                // buscar registro apropiado en el archivo
                archivo.seek( ( posicion ) * Alumno.TAMANIO );

                archivo.writeInt( alumno.getEdad());
                escribirCadena(archivo, alumno.getNombre() );
                escribirCadena(archivo, alumno.getLicenciatura() );
                escribirCadena(archivo, alumno.getMatricula());

             }
```

```
        /*
         * Actualiza un registro
         */

        public void actualizarRegistro( int posicion, Alumno alumno)
                        throws IllegalArgumentException, IOException {


                        // buscar registro apropiado en el archivo
                        archivo.seek( ( posicion  ) * Alumno.TAMANIO );

                        archivo.writeInt( alumno.getEdad());
                        escribirCadena(archivo, alumno.getNombre() );
                        escribirCadena(archivo, alumno.getLicenciatura() );
                        escribirCadena(archivo, alumno.getMatricula());

                    }


        /*
         * Escribe una cadena de caracteres de máximo 50 caracteres
         */
        private void escribirCadena( RandomAccessFile archivo, String cadena )
           throws IOException
        {
           StringBuffer bufer = null;

           if ( cadena != null )
              bufer = new StringBuffer( cadena );
           else
              bufer = new StringBuffer( 50 );

           bufer.setLength( 50 );
           archivo.writeChars( bufer.toString() );
        }



}
```

Principal.java

```java
package uam.edoo.archivos.directo.principal;

import java.io.IOException;

import uam.edoo.archivos.directo.clases.Alumno;
import uam.edoo.archivos.directo.operaciones.OperacionesArchivo;

public class Principal {

	/**
	 * @param args
	 * @throws IOException
	 */
	public static void main(String[] args) throws IOException {

		insertarAlumno(30);
		borrarAlumno(30);
		recuperarAlumno(30);
		Alumno nuevo = nuevoAlumno();
		actualizarAlumno(nuevo,30);
		recuperarAlumno(30);
	}

	/*
	 * Método que manda a insertar un alumno
	 */
	public static void insertarAlumno(int posicion) throws IOException{
		OperacionesArchivo operaciones = new OperacionesArchivo();

		Alumno alumno = leerAlumno();

		operaciones.abrirArchivo("alumnosDirecto.txt");
		operaciones.nuevoRegistro(posicion, alumno);
		operaciones.cerrarArchivo();

	}

	/*
	 * Método que manda a recuperar un alumno
	 */
	public static void recuperarAlumno(int posicion) throws IOException{
		OperacionesArchivo operaciones = new OperacionesArchivo();
		Alumno alumno;
		operaciones.abrirArchivo("alumnosDirecto.txt");
		alumno = operaciones.obtenerRegistro(posicion);
		operaciones.cerrarArchivo();
		System.out.println(alumno.toString());

	}
```

```
	/*
	 * Método que manda a actualizar un alumno
	 */
	public static void actualizarAlumno(Alumno nuevoAlumno, int posicion)throws
IOException{
		OperacionesArchivo operaciones = new OperacionesArchivo();
		operaciones.abrirArchivo("alumnosDirecto.txt");
		operaciones.actualizarRegistro(posicion,nuevoAlumno);
		operaciones.cerrarArchivo();

	}

	/*
	 * Método que manda a borrar un alumno dada una posición
	 */
	public static void borrarAlumno(int posicion)throws IOException{
		OperacionesArchivo operaciones = new OperacionesArchivo();
		operaciones.abrirArchivo("alumnosDirecto.txt");
		operaciones.eliminarRegistro(posicion);
		operaciones.cerrarArchivo();

	}

	/*
	 * Método que llena un alumno
	 */
	public static Alumno leerAlumno(){
		Alumno alumno = new Alumno();
		alumno.setNombre("Josue Figueroa González");
		alumno.setLicenciatura("Ing. en electrónica");
		alumno.setMatricula("matricula");
		alumno.setEdad(34);
		return alumno;
	}

	/*
	 * Método que llena un alumno para
	 * actualizarlo
	 */
	public static Alumno nuevoAlumno(){
		Alumno alumno = new Alumno();
		alumno.setNombre("Ivonne Figueroa González");
		alumno.setLicenciatura("Ing. bioquimica");
		alumno.setMatricula("matricula");
		alumno.setEdad(34);
		return alumno;
	}



}
```