**Nombre Alumno: Sena Nicolás Ariel**

**LU:**

**Nombre Profesor:** Burghardt – Mettini

**Grupo Laboratorio:** Grupo 3

**TP:** 4

**Fecha:** 10/17

En esta unidad estudiamos el manejo de archivos, su aplicación en este lenguaje de programación orientado a objetos, que comparte algunas similitudes con el manejo en otros lenguajes, y acentuadas diferencias como el tratamiento de datos mediante un flujo de bytes, es decir la lectura y escritura se realiza byte por byte.

Existe una variedad de clases para persistir la información en una unidad de almacenamiento secundario o bien leerla. Nombrando las que usamos con frecuencia en esta guía según el tipo de lectura/escritura que hayamos empleado.

**Archivos Secuenciales:** este tipo de archivos no requiere asignación de un tamaño fijo de registro por lo tanto brinda un mejor aprovechamiento del espacio libre, aunque al no fijar un tamaño de registro será imposible utilizarlo con manejo de archivos de tipo directo. La mayor desventaja es que, al ser su lectura secuencial, si el archivo es muy grande habrá que esperar la lectura de todos los registros necesarios hasta llegar a la posición inicial del registro que necesitamos, algo poco eficiente. La escritura y lectura de este tipo de archivos es posible gracias a las clases FileOutputStream (Escritura) y FileInputStream (Lectura) respectivamente, las mismas son auxiliadas por DataOutputStream y DataInputStream.

**Archivos Directos:** Los registros son de tamaño fijo. El manejo de String se realiza mediante una iteración de caracteres en donde hay que determinar en qué momento truncar, o rellenar esa cadena para que cumpla con el tamaño fijo pautado. La principal ventaja de este tipo de archivo es la posibilidad de acceder directamente a un registro específico, mediante el método seek que nos brinda la clase nos posicionamos en el registro deseado, calculando su posición ya que los registros en este lenguaje no son mas que una sucesión de bytes. La manipulación de estos archivos es posible gracias a la clase RandomAccessFile

Gestión de Excepciones: es un mecanismo que nos ofrece el lenguaje para manejar situaciones que al presentarse podrían ocasionar la detención de la ejecución del programa. Al prever las situaciones en las que un módulo pudiera generar un error es posible programar una tarea que se ejecute cuando se detecte dicho error y de esta manera no se interrumpe el programa. Para utilizar esta funcionalidad se debe encerrar un bloque de código entre dos palabras reservadas que son try y catch. try indicará el punto desde donde se empezará a realizar una tarea que en algún momento podría generar alguno de los errores del tipo especificado en un o más catch. En este trabajo se puede apreciar este concepto en todos los procesos de lectura y escritura de archivos. Las excepciones utilizadas fueron FileNotFoundException que se lanza cuando el archivo especificado no existe y IOException que se lanza cuando hay un error en la escritura o lectura del archivo.