**Nombre Alumno: Sena Nicolás Ariel**

**LU:**

**Nombre Profesor:** Burghardt – Mettini

**Grupo Laboratorio:** Grupo 3

**TP:** 5

**Fecha:** 10/17

En esta guía analizamos la implementación de arreglos, conjuntos de elementos de un mismo tipo, con una estructura de datos estatica, también la utilización de estructuras dinámicas como las listas enlazadas desde el paradigma de la Programacion Orientada a Objetos.

La aplicación de estas estructuras de datos la realizamos mediante la reutilización de clases predefinidas que nos brinda la biblioteca de utilidades de java (java.util), en donde aprendimos a sacar provecho de estas clases que ya poseen implementadas las funciones básicas para el manejo de listas, como agregar un elemento, eliminarlo, buscarlo, etc, evitando el tedioso trabajo de actualizar punteros que otros lenguajes poseen. Elementos mejor conocidos como objetos, los cuales son guardados en contenedores de diferentes tipos:

**Estructura Estatica (Array):** Pueden contener datos primitivos (enteros, dobles, cadenas, etc) o identificadores que referencian a un objeto. Es la forma mas fácil de guardar “objetos”, y tipos de datos primitivos, en la primera guarda la referencia, en el segundo, el dato directamente. Su manipulación es sencilla tanto para recorrerla como acceder, usando los corchetes [ i ] y un índice de posición. La gran desventaja, además de contener datos homogéneos es que su capacidad es predefinida y no puede ser incrementada.

**Estructura Dinamicas (Colecciones**): Mediante el Collection Framework, conjunto de clases e interfaces, java nos permite manipular de manera sencilla una lista de referencias a objetos. Set, List y Map (Conjunto, lista y mapa) las estructuras mas comunes. Son similares, cada una presenta pequeñas diferencias y todas permiten aumentar la capacidad durante el tiempo de ejecución. Las **Colecciones** contienen elementos individuales con reglas aplicadas sobre ellos. List los contiene en el orden que fueron insertados y Set no permite la duplicidad de elementos. El **mapa** es un par definido por (Key, Value) en donde Key contiene la clave por la cual un objeto se diferencia del resto (un DNI, un numero de LU) y un valor asociado a esa clave. Un valor “Alumno” seria el objeto a ser localizado mediante su clave asociada (LU).

Como hemos dicho, las colecciones solo almacenan objetos, por lo tanto no es aceptado tipos primitivos, para utilizar estos datos es necesario realizar una envolutura de estos datos a su clase correspondiente. Ejemplo: int a Integer, long a Long.

**ArrayList:** guarda los elementos como una lista. Su acceso a ellos es simple pero presenta desventaja a la hora de insertar o eliminar elementos en una k-esima posición.

**LinkedList:** a diferencia del ArrayList es poco eficiente al acceder a un elemento cualquiera y muy eficiente para insertar o remover elementos en medio de la lista.

**HashMap:** presenta una serie de métodos para manipular los elementos utilizando lo dicho anteriormente, un par definido por (Clave, Valor) en donde una clave nos permite localizar un valor almacenado en el mapa.