Hardoy Juan Segundo – Harvey Benjamín Juan Alberto **DNI: 42994518 – 45526737**

Profesores: Medina Yanina – Airaldi Andrea

Grupo: 2 **TP: 4** **Fecha de entrega: 08/09/25**

En la realización del Trabajo Practico Numero 4, nos basamos y pudimos observar las ventajas de los siguientes conceptos teóricos para la resolución de los problemas que nos fueron dados.

COLECCIONES: Las colecciones son estructuras que contienen elementos con significado similar y que facilitan la manipulación de las mismas. También se los llaman contenedores porque contienen objetos. Los tipos de colecciones que utilizamos en este práctico fueron array (estático), ArrayList y HashMap.

- **Estáticas**: Son contenedores de tamaño fijo (lo definimos previamente y no puede variar durante su vida) y que además sus elementos son del mismo tipo. Su ventaja radica en que son muy eficientes en su almacenamiento y acceso.
- **Dinámica**: Son contenedores que no tienen las limitaciones de tamaño (permite aumentar su capacidad durante la ejecución del programa) y tipo al igual que el array. Podemos accederlos por medio del paquete Collection Framework.
- **Homogéneas**: Son colecciones que contienen elementos de la misma clase.
- **Heterogéneas**: Son colecciones que contienen elementos de distintas clases. En este práctico no las pudimos implementar.
- **Indexadas**: Son colecciones donde la secuencia tiene o no un orden. Un ejemplo de ordenada es un ArrayList y de no ordenada un Set.

```
Punto[] unPunto = new Punto[6];

for(int i = 0; i < unPunto.length; i++){
    System.out.println("Ingrese el valor de x: ");
    int x = teclado.nextInt();
    teclado.nextLine();
    System.out.println("Ingrese el valor de y: ");
    int y = teclado.nextInt();

    unPunto[i] = new Punto(x, y);
}
```

Definición y asignación de valores en un array del objeto unPunto de tamaño 6

```
public class Curso
{
    private String nombre;
    private HashMap<Integer, Alumno> alumnos;
```

Un objeto que tiene como atributo a un HashMap con genéricos que especifican su tipo de clave, valor

```java
public boolean agregarEmpleado(Empleado p_empleado){
    return this.getEmpleados().add(p_empleado);
}
public boolean quitarEmpleado(Empleado p_empleado){
    if(this.getEmpleados().size() > 1){
        return this.getEmpleados().remove(p_empleado);
    }
    else{
        return false;
    }
}

/**
 * Metodo para buscar un empleado por medio de su posicion
 *
 * @param   p_cuil  cuil del empleado
 * @return  devuelve el objeto Empleado
 */
public Empleado buscarEmpleado(int p_pos){
    return (Empleado)this.getEmpleados().get(new Integer(p_pos - 1));
}
```

En este ejemplo se puede apreciar cómo se agrega, elimina y recupera un elemento en un ArrayList

```java
/**
 * Metodo para ingresar un alumno a la coleccion de alumnos.
 *
 * @param   p_alumno    el alumno a ingresar
 */
public void agregarAlumno(Alumno p_alumno){
    this.getAlumnos().put(new Integer(p_alumno.getLU()), p_alumno);
}

/**
 * Metodo para quitar un alumno de la coleccion de alumnos.
 *
 * @param   p_lu    libreta universitaria del alumno
 * @return  devuelve el alumno que se quito
 */
public Alumno quitarAlumno(int p_lu){
    return (Alumno)this.getAlumnos().remove(new Integer(p_lu));
}

    /**
 * Metodo para buscar un alumno por medio de su LU.
 *
 * @param   p_lu    libreta universitaria del alumno a buscar
 * @return  devuelve un objeto Alumno a cual corresponde el LU
 */
public Alumno buscarAlumno(int p_lu){
    return (Alumno)this.getAlumnos().get(new Integer(p_lu));
}
```

En este ejemplo podemos observar cómo se agrega, elimina y recupera un elemento de un HashMap