Nombre de los alumnos: Antoniow Agustín, Alegre Ariel santiago.
Nombre del profesor: Mónica Inés Ringa, Daniela Demchum.
Grupo Laboratorio: Grupo 1 – lunes 15hs.
Trabajo Práctico: 4.
DNI: 46.380.591, 44.467.862.
Fecha de entrega: 22/09/25

En este informe analizamos el uso de los conceptos principalmente del principio de reutilización de código y colecciones, pero también se trabaja con encapsulamiento, conocimiento entre clases e instanciamiento. Todos estos conceptos fueron ejemplificados con clases implementadas en los distintos ejercicios del práctico.

La reutilización de código se trabajó principalmente mediante el uso de clases predefinidas de java como ArrayList, HashMap y HashSet. Estas colecciones permiten modelar situaciones reales sin necesidad de crear estructuras desde cero.
Por ejemplo, en la clase “Pedido” se reutiliza la clase “Producto” para armar una lista de productos solicitados por el cliente, manteniendo el orden de ingreso y permitiendo cálculos de precios de lista y contado.

```
private ArrayList<Producto> productos;
```

El encapsulamiento se aplicó al definir atributos privados y métodos públicos para acceder o modificar los datos. En la clase “Empleado”, por ejemplo, se utilizan getters y setters para manejar el CUIL, nombre y sueldo. También se trabajó el concepto de doble encapsulamiento, donde la propia clase accede a sus atributos a través de sus propios métodos, lo que mejora la seguridad e integridad del software.

```
private void setCuil(long p_cuil){
    this.cuil = p_cuil;
}
```

```
public long getCuil(){
    return this.cuil;
}
```

```
this.getCuil()
```

También el uso de colecciones fue el eje central del práctico. trabajamos con:
. Estáticas, como arrays de tipo Punto, utilizados en el ejercicio 1.

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

. Dinámicas, como arrayList, para manejar listas de empleados, productos o cuentas bancarias.

```
private ArrayList empleados;
private ArrayList cuentasBancarias;
```

. Homogéneas, donde todos los elementos son del mismo tipo, por ejemplo una lista de objetos Producto.

```
private ArrayList<Producto> productos;
```

. Indexadas, como HashMap, que permiten acceder a elementos mediante una clave, como el LU de un alumno o el CUIL de un empleado.

```
private HashMap alumnos;
```

También trabajamos con el tipo de colección hashSet, en el ejercicio relacionado con la gestión de cuentas bancarias en la clase Banco. Usamos HashSet<Persona> para evitar elementos duplicados al listar los titulares.

```
public HashSet<Persona> listaDeTitulares(){
    HashSet<Persona> titulares = new HashSet<>();
    for(Object unaCuenta: this.getCuentasBancarias()){
        titulares.add(((CuentaBancaria)unaCuenta).getTitular());
    }
    return titulares;
}
```

Además, también se vio el concepto de recuperación de elementos desde colecciones, especialmente en aquellas que implementan la interfaz Map, como HashMap. En los ejercicios de Curso y Comercio, se utilizaron métodos como

```
public Alumno buscarAlumno(int p_lu){
    return (Alumno)this.getAlumnos().get(new Integer(p_lu));
}
```

o

```
public Empleado buscarEmpleado(Long p_cuil){
    //No estoy seguro si es necesario castear acá, ya que el HashMap ya espera recibir un valor tipo Empleado.
    return (Empleado)this.getEmpleados().get(new Long(p_cuil));
}
```

Para recuperar objetos directamente mediante su clave, lo que mejora la eficiencia y la claridad del código.

Se implementaron clases ejecutables para cada ejercicio, como ArraysDePuntos, AplicacionBanco, Carrera y gestionDeComercio. Estas clases permiten instanciar objetos, asignarles valores, mostrar sus datos y ejecutar métodos específicos como listar sueldos, calcular totales, buscar elementos o emitir resúmenes. Ejemplo de una clase Ejecutable con un Array estático.

```java
/**
 * Clase ejecutable que instancia 6 elementos de tipo Punto
 */
import java.util.Scanner;
public class ArrayDePuntos{
    public static void main(String args[]){
        Punto arraysPuntos[] = new Punto[6];
        Scanner teclado = new Scanner(System.in);
        for(int i = 0; i < 6; i++){
            System.out.println("Ingrese el x" + (i+1) + ": ");
            double x = teclado.nextDouble();
            System.out.println("Ingrese el y" + (i+1) + ": ");
            double y = teclado.nextDouble();
            arraysPuntos[i] = new Punto(x, y);
        }
        for(Punto unPunto: arraysPuntos){
            unPunto.coordenadas();
        }
        for(int i = 0; i < 5; i++){
            double distancia = arraysPuntos[i].distanciaA(arraysPuntos[i+1]);
            System.out.println("Distancia de " + arraysPuntos[i].coordenadas() + " y " + arraysPuntos[i+1].coordenadas() +
                               " es: " + distancia);
            System.out.println("*-----*");
        }
    }
}
```

Por último, también se aplicó la documentación correspondiente javadoc, y se incluyó ejercicios con menú para interactuar. Este TP nos ayudó a reforzar los conocimientos sobre colecciones en java, reutilización de clases, encapsulamiento y el diseño orientado a objetos (por ej, comprender mejor los diseños de uml), intentando siempre aplicar los principios fundamentales del paradigma OO en situaciones concretas y funcionales.