**Introducción**

**El objetivo del Trabajo Práctico 4 (TP4) fue profundizar en el uso de las colecciones predefinidas en Java, aplicando conceptos fundamentales de Programación Orientada a Objetos (POO) como la reutilización de código, polimorfismo y encapsulamiento. Estas colecciones son clases del lenguaje que nos permiten manejar grupos de objetos de manera eficiente, permitiendo su almacenamiento, búsqueda y manipulación de una forma más flexible que los arrays tradicionales. Además, se explorarán distintos tipos de colecciones para comprender sus propiedades y cuándo resulta más conveniente utilizar cada una.**

**Desarrollo**

**El desarrollo del trabajo consistió en la implementación de un sistema que gestiona distintos tipos de objetos, aprovechando la variedad de colecciones disponibles en Java. Para la resolución de este trabajo práctico, se utiliza principalmente ArrayList, HashSety HashMap, cada una elegida en función de las características requeridas.**

**Colecciones Estáticas y Dinámicas**

**En el desarrollo del TP4, fue crucial la diferencia entre colecciones estáticas (como los arrays) y dinámicas (como las colecciones mencionadas anteriormente). Las colecciones dinámicas, como ArrayListy HashSet, tienen la ventaja de ajustarse automáticamente en tamaño conforme se agregan o eliminan elementos. Esto no solo mejora la flexibilidad del sistema, sino que también reduce la necesidad de asignar más memoria de la necesaria para enfrentar problemas de falta de espacio.**

**Por otro lado, las colecciones estáticas, como los arrays, tienen la limitación de tener un tamaño fijo, lo que puede llevar a un uso ineficiente de la memoria oa la necesidad de redimensionar manualmente. En este TP, trabajar con colecciones dinámicas fue clave para poder implementar soluciones que se adaptan a la variabilidad en el número de objetos gestionados.**

**Teoría Aplicada**

**El TP4 permitió poner en práctica varios conceptos fundamentales de la Programación Orientada a Objetos:**

1. **Encapsulamiento : En POO, el encapsulamiento es la técnica que se utiliza para proteger los datos de una clase, ocultando su estado interno y proporcionando métodos controlados para su manipulación. En este TP, utilicé el encapsulamiento para gestionar los atributos de las clases de manera segura, garantizando que solo se podrán modificar a través de los métodos públicos definidos, como los getters y setters. Este principio mejora la modularidad del código y facilita su mantenimiento a largo plazo.**
2. **Polimorfismo : El polimorfismo es uno de los pilares de la POO y se refiere a la capacidad de tratar objetos de distintas clases de manera uniforme a través de una interfaz común. En este TP, utilicé el polimorfismo para manejar colecciones heterogéneas, donde diferentes tipos de objetos podían ser almacenados y manipulados bajo una misma estructura. Esto permitió aplicar métodos comunes a distintos objetos sin necesidad de conocer su tipo específico, haciendo el código más flexible y escalable.**
3. **Reutilización de código : El uso de colecciones predefinidas es un claro ejemplo de reutilización de código, ya que en lugar de implementar nuestras propias estructuras de datos, aprovechamos las clases provistas por Java que ya están optimizadas y probadas. Este principio es crucial para escribir software eficiente, ya que no solo reduce el tiempo de desarrollo, sino que también disminuye el riesgo de errores en la implementación de estructuras básicas.**
4. **Homogeneidad vs Heterogeneidad : Otro aspecto relevante fue la distinción entre colecciones homogéneas y heterogéneas. Las colecciones homogéneas, como listas de un solo tipo de objeto (por ejemplo, ArrayList<Producto>), son más fáciles de manejar porque se puede asegurar que todos los elementos son del mismo tipo. Sin embargo, las colecciones heterogéneas permiten mayor flexibilidad al almacenar diferentes tipos de objetos, lo cual es útil en situaciones donde los elementos comparten comportamientos comunes a través de la herencia o interfaces, pero difieren en su implementación.**

**Conclusión**

**El TP4 me permitió consolidar los conceptos teóricos de colecciones en Java y su aplicación en un contexto práctico. El uso de estructuras de datos dinámicas como ArrayList, HashSety HashMapfue esencial para gestionar de manera eficiente los diferentes objetos del sistema. A su vez, la aplicación de principios como el encapsulamiento, polimorfismo y reutilización de código garantizó un diseño modular y mantenible.**

**En resumen, este trabajo refuerza la importancia de utilizar las colecciones de Java de manera adecuada y de comprender los conceptos de POO para desarrollar software escalable y eficiente. La capacidad de manejar conjuntos de datos de manera dinámica y flexible, sin preocuparse por las limitaciones de tamaño o tipo, es una de las grandes ventajas de las colecciones en Java, y su uso adecuado resulta clave para el desarrollo de aplicaciones robustas.**