UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programacion Orientada a Objetos

Sesión 10: Colecciones: LinkedList, ArrayList, Hashmap

Recordando:

¿Que vimos la clase pasada?

Universidad Tecnológica del Perú

Logro de aprendizaje

Al finalizar la sesión, el estudiante comprende varias estructuras de datos comunes, como listas, conjuntos, mapas y colas en la solución de problemas usando el lenguaje Java.

Universidad Tecnológica del Perú

Utilidad

- Permite al estudiante tomar decisiones informadas sobre qué tipo de colección usar en función de los requisitos de tu aplicación y las operaciones que necesitas realizar.
- Les permite aprender a agregar, eliminar, buscar, ordenar y manipular datos de manera efectiva utilizando las operaciones proporcionadas por las colecciones.

Agenda

- Concepto de Colecciones.
- Principales Colecciones:
  - ArrayList

Colecciones

- En Java, las colecciones son estructuras de datos que se utilizan para almacenar, manipular y gestionar conjuntos de objetos.
- Las colecciones son una parte fundamental de la programación en Java, ya que proporcionan formas eficientes y flexibles de organizar y trabajar con datos.
- Java ofrece una variedad de interfaces y clases para manejar colecciones, que se encuentran en el paquete java.util.
- A continuación, te explicaré las principales colecciones en Java y sus características:

ArrayList:

Un ArrayList es una estructura de datos que implementa una lista dinámica en Java. Es parte de la biblioteca de colecciones de Java y se encuentra en el paquete java.util. Algunas características clave son:

- **Tamaño dinámico:** A diferencia de los arrays regulares, los ArrayLists pueden cambiar de tamaño automáticamente a medida que se agregan o eliminan elementos.
- **Índices:** Los elementos en un ArrayList se almacenan en un orden secuencial y se acceden mediante índices (posiciones).
- **Permite duplicados:** Puedes almacenar elementos duplicados en un ArrayList. Eficiencia en la lectura: La búsqueda y recuperación de elementos por índice es rápida y eficiente.

Ejemplo de uso de ArrayList en Java:

```
import java.util.ArrayList;

ArrayList<String> listaDeNombres =
new ArrayList<>();
listaDeNombres.add("Juan");
listaDeNombres.add("María");
listaDeNombres.add("Pedro");

String primerNombre =
listaDeNombres.get(0); //Obtiene el primer
nombre (Juan)
```

LinkedList:

Una LinkedList es otra estructura de datos que implementa una lista en Java. Al igual que el ArrayList, está en el paquete java.util, pero se basa en nodos enlazados.

Algunas características importantes son:

- **Doble enlace:** Cada elemento en una LinkedList tiene referencias tanto al elemento anterior como al siguiente, lo que permite una inserción y eliminación eficiente en cualquier posición.
- **Tamaño dinámico:** Al igual que el ArrayList, una LinkedList puede cambiar de tamaño automáticamente.
- **Uso de memoria:** Puede consumir más memoria que un ArrayList debido a las referencias adicionales.

Ejemplo de uso de LinkedList en Java:

```
import java.util.LinkedList;

LinkedList<Integer> listaDeNumeros = new
LinkedList<>();  listaDeNumeros.add(1);
listaDeNumeros.add(2);
listaDeNumeros.add(3);

int primerNumero = listaDeNumeros.getFirst(); //
Obtiene el
primer número (1)
```

HashMap:

Un HashMap es una estructura de datos que implementa una tabla de hash en Java. Se utiliza para almacenar pares clave-valor y permite un acceso rápido a los valores a través de sus claves. Algunas características clave son:

- **Clave-valor:** Almacena datos en forma de pares clave-valor, donde cada clave es única.
- **Búsqueda rápida:** Permite recuperar valores rápidamente a través de la clave, lo que lo hace eficiente en términos de tiempo de acceso.
- **No permite duplicados de claves:** Cada clave en un HashMap debe ser única, pero los valores pueden repetirse.
- **No garantiza un orden específico:** Los elementos no se almacenan en un orden específico en un HashMap.

Ejemplo de uso de HashMap en Java:

```
import java.util.HashMap;

HashMap<String, Integer> mapaDePuntajes = new HashMap<>();
mapaDePuntajes.put("Juan", 85);
mapaDePuntajes.put("María", 92);
mapaDePuntajes.put("Pedro", 78);

int puntajeMaria = mapaDePuntajes.get("María"); // Obtiene el
puntaje de María (92)
```

ArrayList

- Es una clase que se representa como una matriz dinámica que permite almacenar elementos.
- Hereda de la clase AbstractList, la cual implementa la interfaz List.
- Permite colecciones o elementos duplicados.
- El orden de los registros es el orden en el que fueron insertados.
- Permite acceso aleatorio (tiene índice)
- Manipulación lenta (recorrer tooodo el arraylist para hacer un cambio)

ArrayList

https://www.youtube.com/watch?v=qGIdpq98Tk4

Ejercicio de “ArrayList”

En este ejemplo, crearemos una lista de productos musicales que un cliente puede comprar en la tienda.
Cada producto tendrá un nombre y un precio. Los clientes podrán agregar productos a su carrito de compras y calcular el precio total de los productos seleccionados.

```
public class Producto {
   private String nombre;
   private double precio;
   public Producto(String nombre, double
precio) {
      this.nombre = nombre;
      this.precio = precio;
   }
   public String getNombre() {
      return nombre;
   }
   public double getPrecio() {
      return precio;
   }
}
```

Ejercicio de “ArrayList”

```
import java.util.ArrayList;
import java.util.Scanner;
public class TiendaMusical {
   public static void main(String[] args) {
      ArrayList<Producto> catalogo = new ArrayList<>();
      catalogo.add(new Producto("CD de música", 10.99));
      catalogo.add(new Producto("Vinilo", 29.99));
      catalogo.add(new Producto("Auriculares", 49.99));
      ArrayList<Producto> carrito = new ArrayList<>();
      double total = 0.0;
      Scanner scanner = new Scanner(System.in);
      System.out.println("Bienvenido a la Tienda Musical");
      while (true) {
         System.out.println("\nCatálogo de productos:");
         for (int i = 0; i < catalogo.size(); i++) {
            Producto producto = catalogo.get(i);
            System.out.println(i + 1 + ". " + producto.getNombre() + " - $" + producto.getPrecio());
         }
```

Ejercicio de “ArrayList”

```
System.out.print("Elija un producto (1-" + catalogo.size() + ") o 0 para salir: ");
            int opcion = scanner.nextInt();
            if (opcion == 0) {
                break;
            } else if (opcion >= 1 && opcion <= catalogo.size()) {
                Producto productoSeleccionado = catalogo.get(opcion - 1);
                carrito.add(productoSeleccionado);
                total += productoSeleccionado.getPrecio();
                System.out.println("Producto agregado al carrito: " +
productoSeleccionado.getNombre());
            } else {
                System.out.println("Opción no válida. Por favor, elija una opción
válida.");
            }
        }
        System.out.println("\nResumen de la compra:");
        for (Producto producto : carrito) {
            System.out.println(producto.getNombre() + " - $" + producto.getPrecio());
        }
        System.out.println("Total: $" + total);
        System.out.println("Gracias por comprar en la Tienda Musical.");
    }
```

Practica

Implementa el ejercicio visto.

Conclusiones

Las colecciones en Java son una parte esencial del desarrollo de software y pueden proporcionar numerosos beneficios en términos de organización de datos, eficiencia y legibilidad del código. Sin embargo, es importante utilizarlas de manera adecuada y consciente para aprovechar al máximo sus ventajas.

Cierre

¿Qué hemos aprendido hoy?

Bibliografía

- MORENO PÉREZ, J. "Programación orientada a objetos". RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. "Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java". Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú