Unidad 14

Colecciones de datos

Contenidos

*Parte 1: Tipos, Jerarquías, Operaciones y Uso de clases ****de colecciones*

Las **colecciones de datos** son estructuras que permiten almacenar y gestionar grupos de objetos de manera eficiente. Java proporciona una rica biblioteca de colecciones en el paquete `java.util`, que incluye listas, pilas, colas y tablas, entre otras. Este capítulo explorará los tipos, jerarquías y operaciones comunes con colecciones, además del uso de clases y métodos genéricos.

1. Tipos de Colecciones

1.1. Listas (`List`)

Una **lista** es una colección ordenada que permite elementos duplicados. Cada elemento tiene un índice que se puede utilizar para acceder a él.

Implementaciones Comunes

1. `ArrayList`: **Basado en un array dinámico. Es rápido para accesos aleatorios pero lento para inserciones y eliminaciones.**
2. `LinkedList`: **Basado en una lista enlazada. Es más eficiente para inserciones y eliminaciones.**

Ejemplo con `ArrayList`

```
import java.util.ArrayList;
public class Main {
    public static void main(String[] args) {
          ArrayList<String> lista = new ArrayList<>();
          lista.add("Manzana");
          lista.add("Banana");
          lista.add("Cereza");
          System.out.println(lista.get(1)); // Accede al segundo elemento
    }
}
```

Salida:

```
Banana
```

1.2. Pilas (`Stack`)

Una **pila** es una colección basada en el principio **LIFO** (Last In, First Out), donde el último elemento en entrar es el primero en salir.

Ejemplo con `Stack`

```
  import java.util.Stack;
  public class Main {
      public static void main(String[] args) {
            Stack<Integer> pila = new Stack<>();
            pila.push(10);
            pila.push(20);
            pila.push(30);
            System.out.println(pila.pop()); // Elimina y devuelve el último ele-
mento
      }
  }
```

Salida:

```
  30
```

1.3. Colas (`Queue`)

Una **cola** es una colección basada en el principio **FIFO** (First In, First Out), donde el primer elemento en entrar es el primero en salir.

Ejemplo con `LinkedList` como Cola

```
  import java.util.LinkedList;
  import java.util.Queue;
  public class Main {
      public static void main(String[] args) {
            Queue<String> cola = new LinkedList<>();
            cola.add("Uno");
            cola.add("Dos");
            cola.add("Tres");
            System.out.println(cola.poll()); // Elimina y devuelve el primer el-
emento
      }
  }
```

Salida:

```
Uno
```

1.4. Tablas (`Map`)

Un **mapa** es una colección que asocia claves únicas con valores. No permite claves duplicadas, pero sí valores duplicados.

Ejemplo con `HashMap`

```
import java.util.HashMap;
public class Main {
    public static void main(String[] args) {
          HashMap<Integer, String> mapa = new HashMap<>();
          mapa.put(1, "Enero");
          mapa.put(2, "Febrero");
          mapa.put(3, "Marzo");
          System.out.println(mapa.get(2)); // Devuelve el valor asociado a la clave 2
    }
}
```

Salida:

```
Febrero
```

2. Jerarquías de Colecciones

Java organiza las colecciones en una jerarquía bien estructurada dentro del paquete `java.util`. Todas las colecciones derivan de la interfaz `Collection`, excepto los mapas (`Map`).

```
Collection
|
|-- List
|   |-- ArrayList
|   |__ LinkedList
|
|-- Set
|   |-- HashSet
|   |__ TreeSet
|
|-- Queue
|   |── LinkedList
|   |── PriorityQueue
|
|-- Map (independiente de Collection)
    ├── HashMap
    └── TreeMap
```

3. Operaciones con Colecciones

3.1. Acceso a Elementos

- **Listas**: Acceso por índice usando `get`.
- **Mapas**: Acceso por clave usando `get`.

Ejemplo con Lista

```java
ArrayList<String> lista = new ArrayList<>();
lista.add("A");
lista.add("B");
System.out.println(lista.get(0)); // Devuelve "A"
```

Ejemplo con Mapa

```java
HashMap<String, Integer> mapa = new HashMap<>();
mapa.put("Enero", 31);
System.out.println(mapa.get("Enero")); // Devuelve 31
```

3.2. Recorridos

Las colecciones se pueden recorrer usando bucles tradicionales, iteradores o expresiones lambda.

Ejemplo: Recorrido de una Lista

```
ArrayList<String> lista = new ArrayList<>();
lista.add("A");
lista.add("B");
for (String elemento : lista) {
    System.out.println(elemento);
}
```

4. Uso de Clases y Métodos Genéricos

Las colecciones en Java son genéricas, lo que significa que se puede especificar el tipo de datos que almacenan.

Ejemplo de Uso de Genéricos con `ArrayList`

```
ArrayList<Integer> numeros = new ArrayList<>();
numeros.add(10);
numeros.add(20);
for (Integer numero : numeros) {
    System.out.println(numero);
}
```

Salida:

```
10
20
```

Ejemplo de Método Genérico

```
public class Utilidades {
    public static <T> void imprimirColeccion(ArrayList<T> coleccion) {
          for (T elemento : coleccion) {
                System.out.println(elemento);
          }
    }
}
public class Main {
    public static void main(String[] args) {
          ArrayList<String> lista = new ArrayList<>();
          lista.add("Hola");
          lista.add("Mundo");
          Utilidades.imprimirColeccion(lista);
    }
}
```

Salida:

```
Hola
Mundo
```

5. Ejercicio Guiado

Problema

1. Crea una lista genérica que almacene nombres de personas.

2. Añade nombres a la lista.

3. Recorre la lista usando un bucle `for` y un iterador.

4. Crea un mapa que asocie nombres con edades y recorre el mapa para imprimir la relación clave-valor.

6. Preguntas de Evaluación

1. ¿Qué principio sigue una pila en Java?

a) FIFO.
b) LIFO.
c) FILO.
d) Ninguna de las anteriores.

2. ¿Qué clase implementa la interfaz `Queue` en Java?

a) `ArrayList`.
b) `LinkedList`.
c) `HashSet`.
d) `TreeMap`.

3. ¿Qué método se usa para acceder a un elemento en un mapa por su clave?

a) `get`.
b) `put`.
c) `add`.
d) `remove`.