Material descargable

egg

Java Collections Framework

¿Qué es una Colección del tipo Set?

Un Set es una colección que garantiza la exclusión de elementos duplicados. Esta característica se establece a través del comportamiento del método add() en un Set, el cual devuelve false si el elemento ya está presente en el conjunto y no se añade nuevamente.

A pesar de que los conjuntos evitan la duplicación de elementos, cada elemento individual puede ser único según su propio criterio de igualdad, el cual se define mediante los métodos equals() y hashCode() del objeto. Dos objetos diferentes pueden considerarse iguales si su implementación de equals() así lo define, incluso si contienen valores distintos en sus atributos.

¿Qué es HashSet?

HashSet es una implementación de la interfaz Set en el Java Collections Framework que se encuentra en el paquete java.util. Utiliza una estructura de datos basada en tablas hash para almacenar elementos.

A diferencia de otras implementaciones de Set que mantienen algún tipo de orden, como TreeSet, HashSet no garantiza ningún orden particular de los elementos y no permite duplicados. Cada elemento en un HashSet debe ser único.

Papel en el Framework de Colecciones de Java:

HashSet desempeña un papel crucial en el Framework de Colecciones de Java y se utiliza en una variedad de situaciones debido a sus características específicas:

- **Unicidad de Elementos:** HashSet garantiza que no puede haber elementos duplicados. Si intentas agregar un elemento que ya está presente, la operación de inserción no tiene efecto.
- **Eficiencia en Operaciones de Búsqueda:** La implementación basada en tablas hash permite a operaciones como agregar, eliminar y verificar la existencia de un

elemento se realicen en tiempo constante en promedio, lo que hace que HashSet sea eficiente para conjuntos de datos grandes.

- **No Garantiza Orden:** A diferencia de algunas otras implementaciones de Set, como TreeSet, HashSet no mantiene ningún orden específico de los elementos. Los elementos no se almacenan en el orden en que se insertan.
- **Adaptabilidad a Objetos Personalizados:**HashSet es capaz de manejar objetos personalizados siempre que estos objetos implementen correctamente los métodos hashCode y equals.
- **Escenarios de Uso Común:** HashSet se utiliza comúnmente en situaciones donde la verificación rápida de pertenencia a un conjunto y la eliminación de duplicados son requisitos clave.

Operaciones Básicas de HashSet:

- **Crear un HashSet de enteros:**

```
// Crear un HashSet de enteros
HashSet<Integer> conjuntoDeEnteros = new HashSet<>();
```

- **Agregar Elementos (add):**

```
// Agregar elementos al HashSet
conjuntoDeEnteros.add(e:10);
conjuntoDeEnteros.add(e:20);
conjuntoDeEnteros.add(e:30);
conjuntoDeEnteros.add(e:40);
```

- **Eliminar Elementos (remove):**

```
// Eliminar un elemento del HashSet
int elementoAEliminar = 20;
conjuntoDeEnteros.remove(elementoAEliminar);
```

- **Verificar la Existencia (contains):**

```java
// Validar si un elemento existe en el HashSet
int elementoBuscado = 20;

if (conjuntoDeEnteros.contains(elementoBuscado)) {
    System.out.println(elementoBuscado + " está presente en el HashSet.");
} else {
    System.out.println(elementoBuscado + " no está presente en el HashSet.");
}
```

- **Obtener el Tamaño (size):**

```java
// Obtener el tamaño del HashSet
int tamano = conjuntoDeEnteros.size();

// Mostrar el tamaño del HashSet
System.out.println("Tamaño del HashSet: " + tamano);
```

- **Iteración a través del HashSet:**

```java
// Iterar sobre el HashSet de enteros con un bucle for-each
System.out.println(x:"Iteración con bucle for-each:");
for (Integer numero : conjuntoDeEnteros) {
    System.out.println(numero);
}

// Iterar sobre el HashSet de enteros con un iterador
System.out.println(x:"Iteración con iterador:");
java.util.Iterator<Integer> iterator = conjuntoDeEnteros.iterator();
while (iterator.hasNext()) {
    Integer numero = iterator.next();
    System.out.println(numero);
}
```

HashCode y Equals en HashSet

Los métodos **`hashCode`** y **`equals`** son fundamentales en el funcionamiento interno de los **HashSet** en Java, ya que garantizan la unicidad y eficiencia en la gestión de elementos.

`hashCode()`

- El método **`hashCode()`** calcula un valor hash único para cada objeto. Este valor se utiliza para determinar la ubicación del objeto dentro de una tabla hash, la estructura subyacente de un `HashSet`.

- La calidad de la implementación del método **`hashCode()`** afecta directamente el rendimiento de las operaciones, ya que un mal diseño puede llevar a colisiones frecuentes.

`equals()`

- **`equals()`** se usa para comparar dos objetos y determinar si son "iguales" en un sentido lógico.
- Cuando se agrega un elemento al `HashSet`, primero se calcula su valor hash con **`hashCode()`**, y luego se compara con otros elementos en la misma ubicación mediante **`equals()`** para garantizar que no haya duplicados.

A continuación, se presenta un ejemplo de implementación de los métodos `hashCode` y `equals` en una clase personalizada:

```
import java.util.HashSet;

public class Libro {
    private String titulo;
    private String autor;
    private int isbn;

    public Libro(String titulo, String autor, int isbn) {
        this.titulo = titulo;
        this.autor = autor;
        this.isbn = isbn;
    }

    @Override
    public int hashCode() {
        return isbn;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) // Verifica si son la misma referencia
            return true;
        if (obj == null || getClass() != obj.getClass())
            return false; // Compara tipos
        Libro otroLibro = (Libro) obj;
          return isbn== otroLibro.isbn;// Compara ISBN para determinar
igua
    }
}

public class Main {
    public static void main(String[] args) {
```

```
            HashSet<Libro> catalogo = new HashSet<>(); // Catálogo de
libros.

        // Agregando libros al HashSet
            Libro libro1 = new Libro("Cien años de soledad", "Gabriel
García Márquez", 123456789);
              Libro libro2 = new Libro("El Principito", "Antoine de
Saint-Exupéry", 987654321);
        catalogo.add(libro1);
        catalogo.add(libro2);

        // Verificando si un libro está presente en el HashSet
            Libro libro3 = new Libro("Cien años de soledad", "Gabriel
García Márquez", 123456789);
         boolean existe= catalogo.contains(libro3); // Debería devolver
true
            System.out.println("¿El libro ya está en el catálogo? " +
existe);

        // Eliminar un libro del HashSet
        catalogo.remove(libro2);
  System.out.println("Libros en el catálogo: " + catalogo.size());
    }
}
```

En este ejemplo, la clase Libro representa un libro con atributos como título, autor y número de ISBN. Los métodos hashCode() y equals() se han sobreescrito para garantizar que dos libros sean considerados iguales si tienen el mismo número de ISBN.

Cuando se agregan libros al HashSet, el método hashCode() se utiliza para calcular un valor hash que se traduce en un índice en la tabla hash interna. Luego, el método equals() se utiliza para determinar si dos libros son iguales, evitando así duplicados en el conjunto.

💡Siempre es recomendable utilizar la **documentación oficial de Oracle** para obtener más detalles y ejemplos prácticos