UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Estructuras de Datos

Práctica 4: Iteradores

Profesora:
Amparo López Gaona
Ayudante: Adrián Aguilera Moreno
Ayudante de Laboratorio: Kevin Jair Torres Valencia

Objetivos

- Mejorar el razonamiento lógico sobre tipos de datos abstractos.
- Implementar la interfaz Iterator y sobreescribir sus métodos.

Introducción

Un iterador es un objeto que permite recorrer una colección de objetos. Para crear un iterador se debe implementar la interfaz `Iterator` del paquete `java.util` que tiene la siguiente forma:

```
1 public interface Iterator {
2     public boolean hasNext();
3     public Object next() throws NoSuchElementException;
4     public void remove() throws UnsupportedOperationException,
5                                 IllegalStateException;
6 }
```

Listing 1: Interfaz Iterator

Los métodos de la interfaz `Iterator` tienen el siguiente propósito:

- **hasNext** devuelve `true` si en la colección aún hay al menos un elemento. En caso de no haber más elementos devuelve `false`.
- **next** devuelve el siguiente elemento de la colección. La primera vez que se llama a este método devuelve el primer elemento de la colección. Si la colección no tiene elementos y se llama a este método, se dispara la excepción `NoSuchElementException`.
- **remove** permite eliminar de la colección el último elemento devuelto por el iterador. Este método puede ser llamado sólo una vez por cada llamada al método **next**. Esta operación es opcional; si se decide no permitir esta operación es necesario disparar la excepción `UnsupportedOperationException`.

Normalmente las clases que implementan o tienen colecciones incluyen un método que devuelve un iterador. En general, el iterador se implementa en una clase interna privada debido a que ésta tiene acceso, sin calificativo, a todos los métodos y estructura del objeto que la encierra, independientemente de la visibilidad de éstos afuera de la clase.

Desarrollo

Un contador tradicional incrementa valores dentro de un rango definido, pero cuando alcanza su límite, se detiene. Para ello desarrollarán el `TDA ContadorCircular`, donde permitirán recorrer un conjunto de valores de manera cíclica.

El `Contador Circular` tendrá un inicio y un fin predefinidos. Cada vez que alcance el valor final, deberá reiniciarse al inicio, repitiendo este proceso un número determinado de veces antes de detenerse por completo. Para lograrlo, deberán implementar la interfaz `Iterator<Integer>` para recorrer los valores de forma eficiente y controlada.

Además, el iterador debe permitir eliminar el último número retornado con el método `remove`, evitando modificar el comportamiento del recorrido circular.

Para la implementación, deben considerar lo siguiente:

- El contador debe recorrer números en un rango definido `[inicio, fin]`
- Debe repetir el ciclo hasta completar un número de vueltas (repeticiones).
- Si el contador alcanza el valor fin, debe volver a inicio y continuar.

Para el iterador, también deben considerar:

- Se debe implementar la interfaz `Iterator<Integer>`
- `hasNext`-Indica si quedan elementos disponibles para iterar.
- `next`-Devuelve el siguiente número válido en la secuencia.
- `remove`-Elimina el último número retornado, evitando que vuelva a aparecer.

Algunas restricciones son:

- No se debe usar List, Set ni otras estructuras dinámicas de almacenamiento.
- `remove` solo puede ser llamado después de `next`, de lo contrario, debe lanzar una excepción.
- Como sugerencia utilicen un arreglo booleano para marcar los números eliminados.

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. NO incluir los archivos .class dentro de la carpeta.
3. Los archivos de código fuente deben estar documentados.
4. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
5. La práctica se debe subir al Github Classroom correspondiente.
6. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
7. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.