CSCI 2120:
Software Design & Development II

UNIT2: Data management

Collections Framework & Generics

ArrayDeque

Overview

1. Introduction
2. Hierarchy of ArrayDeque class
3. ArrayDeque class declaration
4. Features of ArrayDeque
5. ArrayDeque Constructors
6. ArrayDeque Methods
7. ArrayDeque Examples

Introduction

ArrayDeque in Java is a concrete class that creates a dynamic array (resizable array) to store its elements from both sides of the queue.

It provides a resizable-array implementation of deque interface. Therefore, it is also known as array double-ended queue or simply array deck.

Since ArrayDeque has no capacity restrictions, ArrayDeque is especially useful for implementing stacks and queues whose sizes are not known in advance.

ArrayDeque class was added by Java 6 version that makes it a recent addition to the Java Collections Framework. It is present in java.util.ArrayDeque package.

Hierarchy of ArrayDeque class

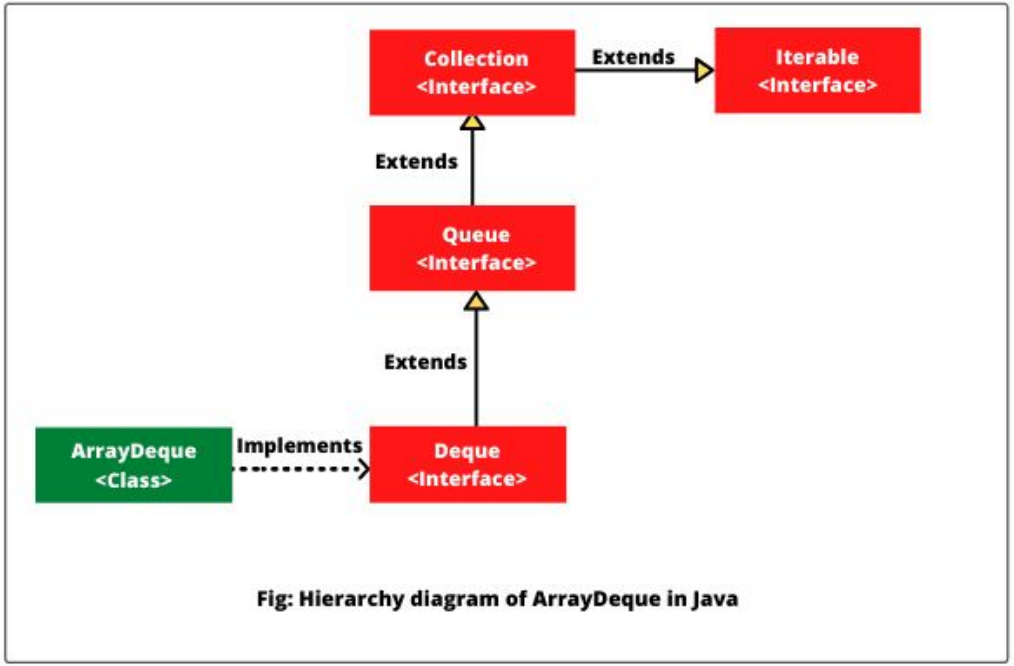

Fig: Hierarchy diagram of ArrayDeque in Java

`ArrayDeque` class implements double-ended queue interface to support the addition of elements from both sides of queue. It also implements queue interface to support the first in first out data structure.

All implemented interfaces by ArrayDeque class in the hierarchy are `Serializable, Cloneable, Iterable<E>, Collection<E>, Deque<E>, Queue<E>`.

ArrayDeque class declaration

ArrayDeque is a generic class that can be declared as follows:

```
public class ArrayDeque<E>
       extends AbstractCollection<E>
       implements Deque<E>, Cloneable, Serializable
```

Here, E represents the type of objects stored in the collection.

Features of ArrayDeque

There are several important features of array deck that should be kept in mind. They are as:

1. Java ArrayDeque class provides a resizable array implementation of Deque interface.

2. It has no capacity restrictions. So, it can grow according to the need to handle elements added to the collection.

3. Array deque is not synchronized. That means it is not thread-safe. Multiple threads can access the same ArrayDeque object at the same time.

4. Null elements are restricted in the ArrayDeque.

5. ArrayDeque class performs faster operations than Stack when used as a stack.

6. ArrayDeque class performs faster operations than LinkedList when used as a queue.

7. The iterators returned by array deque class are fail-fast: If the deque is modified during iteration, the iterator will generally throw an exception named `ConcurrentModificationException`.

ArrayDeque Constructors

1. ArrayDeque(): This constructor creates an empty array deque with starting capacity of 16 elements. The general syntax to create array deque object is as follows:

```
ArrayDeque<E> dq = new ArrayDeque<E>();
```

ArrayDeque Constructors

2. ArrayDeque(int numElements): This constructor creates an empty array deque with the specified initial capacity sufficient to hold elements. If the argument passed to numElements is less than or equal to zero, no exception will be thrown.

The general syntax to create ArrayDeque object with the specified initial capacity is as follows:

```
ArrayDeque<E> dq = new ArrayDeque<E>(int numElements);

//For example:
ArrayDeque<String> dq = new ArrayDeque<String>(5);
```

ArrayDeque Constructors

3. ArrayDeque(Collection c): This constructor creates an array deque that is initialized with elements of collection c. If c contains null reference then NullPointerException will be thrown.

The general syntax to create ArrayDeque instance with the specified collection is given below:

```
ArrayDeque<E> dq = new ArrayDeque<E>(Collection c);
```

In all cases, the capacity can grow as per the requirement to handle elements inserted into the collection.

ArrayDeque Methods

ArrayDeque in Java adds no methods of its own. All methods are inherited by Deque, Queue, and Collection interface. The important methods are given below:

ArrayDeque Methods

Method	Description
boolean add(Object o)	It inserts the specified element to the end of deque.
void addFirst(Object o)	It inserts the specified element to the front of the deque.
void addLast(Object o)	It inserts the specified element to the last of deque.
void clear()	This method is used to remove all elements from deque.
Object clone()	It is used to get a copy of ArrayDeque instance.

ArrayDeque Methods

Method	Description
boolean contains(Object o)	It returns true if the deque contains the specified element.
Object element()	This method is used to retrieve an element from the head of deque. It does not remove element.
Object getFirst()	This method is used to retrieve the first element of deque. It does not remove element.
Object getLast()	This method is used to retrieve the last element of deque. It does not remove element.
boolean isEmpty()	The isEmpty() method returns true if deque has no elements. 11. boolean offer(Object element): It inserts the specified element at the end of deque.

ArrayDeque Methods

Method	Description
boolean offerFirst(Object o)	It inserts the specified element at the front of deque.
boolean offerLast(Object o)	It inserts the specified element at the end of deque.
Object peek()	The peek() method retrieves element from the head of deque but does not remove. If the deque is empty, it returns null element.
Object peekFirst()	The peekFirst() method retrieves the first element from the head of deque but does not remove it. It returns null element if the deque is empty.
Object peekLast()	The peekLast() method retrieves the last element from deque but does not remove it. It returns null element if the deque is empty.

ArrayDeque Methods

Method	Description
Object poll()	The poll() method retrieves and removes the element from the head of deque. It returns null element if the deque is empty.
Object pollFirst()	The pollFirst() method retrieves and removes the first element of deque. It returns null element if the deque is empty.
Object pollLast()	The pollLast() method retrieves and removes the last element of deque. It returns null element if the deque is empty.
Object pop()	The pop() method is used to pop an element from the stack represented by the deque.
Object push(Object o)	The push() method is used to push an element from the stack represented by the deque.

ArrayDeque Methods

Method	Description
Object remove()	The remove() method is used to retrieve and remove an element from the deque.
boolean removeFirst()	This method is used to retrieve and remove the first element from the deque.
boolean removeFirstOccurrence(Object o)	This method removes the last occurrence of the specified element from the deque.
int size()	It returns the number of elements in the deque.
Object toArray()	It returns an array that contains all elements in the deque in the proper sequence from first to last element.

ArrayDeque Methods

Method	Description
Iterator iterator()	It returns an iterator over elements from the deque.
Iterator descendingIterator()	It returns an iterator over elements in reverse sequential order from the deque.

ArrayDeque Examples

Let’s take some useful example programs to perform the various operations based on the above methods.

Example 1: Adding elements

1. Adding elements: Let’s create a program where we will add elements to the ArrayDeque using methods such as add(), addFirst(), addLast(), offer(), offerFirst(), offerLast().

Example 1: Adding elements

```
import java.util.ArrayDeque;
public class ArrayDequeTester1 {
   public static void main(String[] args) {
       // Create object of ArrayDeque class of String type.
       ArrayDeque<Integer> dq = new ArrayDeque<Integer>();

       // Adding elements to deque using add() method.
       dq.add(10);
       dq.addFirst(20);
       dq.addLast(05);

       // Adding elements to deque using offer() method.
       dq.offer(30);
       dq.offerFirst(50);
       dq.offerLast(40);

       System.out.println("Elements in ArrayDeque : " + dq);
   }
}
```

Example 1: Adding elements

Output:

```
Elements in ArrayDeque : [50, 20, 10, 5, 30, 40]
```

Example 2: Accessing elements

2. Accessing Elements: Let’s create a program where we will access elements using methods like getFirst(), getLast(), etc.

Example 2: Accessing elements

```
import java.util.ArrayDeque;
public class ArrayDequeTester2 {
   public static void main(String[] args) {
       // Creating an empty ArrayDeque instance.
       ArrayDeque<Integer> dq= new ArrayDeque<Integer>();

       dq.add(25);
       dq.add(50);
       dq.add(75);
       dq.add(100);
       dq.add(125);

       // Displaying elements of ArrayDeque.
       System.out.println("ArrayDeque: " + dq);
       System.out.println("First element is: " + dq.getFirst());
       System.out.println("Last element is: " + dq.getLast());
   }
}
```

Example 2: Accessing elements

Output:

```
ArrayDeque: [25, 50, 75, 100, 125]
First element is: 25
Last element is: 125
```

Example 3: Removing elements

3. Removing Elements: Let’s make a program where we will remove an element from a deque using various methods available such as removeFirst(), removeLast(), poll(), pop(), pollFirst(), pollLast().

Example 3: Removing elements

```
import java.util.ArrayDeque;
public class ArrayDequeTester3 {
   public static void main(String[] args) {
       ArrayDeque<String> dq= new ArrayDeque<String>();

       dq.add("One");
       dq.addFirst("Two");
       dq.addLast("Three");

       // Displaying elements of ArrayDeque.
       System.out.println("Elements in ArrayDeque : " + dq);

       // Remove element as a stack from top/front end.
       System.out.println(dq.pop());

       // Remove element as a queue from front
       System.out.println(dq.poll());

       // Remove element from front end.
       System.out.println(dq.pollFirst());

       // Remove element from back end.
       System.out.println(dq.pollLast());
   }
}
```

Example 3: Removing elements

Output:

```
Elements in ArrayDeque : [Two, One, Three]
Two
One
Three
null
```

Example 4: Iterating over elements of Deque

4. Iterating over elements of Deque: ArrayDeque can be iterated over elements from both ends of deque using iterator() and descendingIterator() methods. Look at the following source code.

Example 4: Iterating over elements of Deque

```
import java.util.ArrayDeque;
import java.util.Iterator;
public class ArrayDequeTester4 {
   public static void main(String[] args) {
       ArrayDeque<String> dq= new ArrayDeque<String>();

       dq.add("One");
       dq.addFirst("Two");
       dq.addLast("Three");
       dq.add("Four");

       // Iterating over elements from the front end of the queue using iterator() method.
       System.out.println("Front End Iteration:");
       Iterator<String> itr = dq.iterator();
       while(itr.hasNext()){
           System.out.println(itr.next());
       }

       System.out.println();

       // Iterating over elements in reverse order.
       System.out.println("Back End Iteration:");
       Iterator<String> itr2 = dq.descendingIterator();
       while(itr2.hasNext()){
           System.out.println(itr2.next());
       }
   }
}
```

Example 4: Iterating over elements of Deque

Output:

```
Front End Iteration:
Two
One
Three
Four

Back End Iteration:
Four
Three
One
Two
```

Example 5: ArrayDeque as a Stack

5. ArrayDeque as a Stack: Let’s take an example program where we will use ArrayDeque to create a Stack. Look at the following source code to understand better.

Example 5: ArrayDeque as a Stack

```
import java.util.ArrayDeque;
public class ArrayDequeTester5 {
   public static void main(String[] args) {
       ArrayDeque<String> dq= new ArrayDeque<String>();

       String[] weekdays = {"Sunday", "Monday", "Tuesday", "Thursday", "Friday", "Saturday"};
       System.out.println("Poping Stack: ");

       for(String weekday: weekdays)
           dq.push(weekday);
       while(dq.peek() != null){
           System.out.println(dq.pop());
       }
   }
}
```

Example 5: ArrayDeque as a Stack

Output:

```
Poping Stack:
Saturday
Friday
Thursday
Tuesday
Monday
Sunday
```

END