```
/* Most of the time when you use an array it's to place integers inside of it, so why not
have a super fast integer only array? This file contains an implementation of an integer
only array which can outperform Java's ArrayList by about a factor of 10-15x! Enjoy!
  @author William Fiset, william.alexandre.fiset@gmail.com */
package com.williamfiset.datastructures.dynamicarray;

public class IntArray implements Iterable<Integer> {

  private static final int DEFAULT_CAP = 1 << 3;

  public int[] arr;
  public int len = 0;
  private int capacity = 0;

  // Initialize the array with a default capacity
  public IntArray() {    this(DEFAULT_CAP);
  }

  // Initialize the array with a certain capacity
  public IntArray(int capacity) {
    if (capacity < 0) throw new IllegalArgumentException("Illegal Capacity: " +
capacity);
    this.capacity = capacity;
    arr = new int[capacity];
  }

  // Given an array make it a dynamic array!
  public IntArray(int[] array) {
    if (array == null) throw new IllegalArgumentException("Array cannot be null");
    arr = java.util.Arrays.copyOf(array, array.length);
    capacity = len = array.length;
  }

  // Returns the size of the array
  public int size() {    return len;
  }
  // Returns true/false on whether the array is empty
  public boolean isEmpty() {    return len == 0;  }
```

```
// To get/set values without method call overhead you can do
// array_obj.arr[index] instead, you can gain about 10x the speed!
public int get(int index) {    return arr[index];  }

public void set(int index, int elem) {    arr[index] = elem;
}

// An an element to this dynamic array
public void add(int elem) {
  if (len + 1 >= capacity) {
    if (capacity == 0) capacity = 1;
    else capacity *= 2; // double the size
    arr = java.util.Arrays.copyOf(arr, capacity); // pads with extra 0/null elements
  }
  arr[len++] = elem;
}

// Removes the element at the specified index in this list.
// If possible, avoid calling this method as it take O(n) time
// to remove an element (since you have to reconstruct the array!)
public void removeAt(int rm_index) {
  System.arraycopy(arr, rm_index + 1, arr, rm_index, len - rm_index - 1);
  --len;
  --capacity;
}

// Search and remove an element if it is found in the array
// If possible, avoid calling this method as it take O(n) time
public boolean remove(int elem) {
  for (int i = 0; i < len; i++) {
    if (arr[i] == elem) {
      removeAt(i);
      return true;
    }
  }
  return false;
}
```

```
// Reverse the contents of this array
public void reverse() {
  for (int i = 0; i < len / 2; i++) {
    int tmp = arr[i];
    arr[i] = arr[len - i - 1];
    arr[len - i - 1] = tmp;
  }
}

// Perform a binary search on this array to find an element in O(log(n)) time
// Make sure this array is sorted! Returns a value < 0 if item is not found
public int binarySearch(int key) {
  int index = java.util.Arrays.binarySearch(arr, 0, len, key);
  // if (index < 0) index = -index - 1; // If not found this will tell you where to insert
  return index;
}

// Sort this array
public void sort() {    java.util.Arrays.sort(arr, 0, len);
}

// Iterator is still fast but not as fast as iterative for loop
@Override
public java.util.Iterator<Integer> iterator() {
  return new java.util.Iterator<Integer>() {
    int index = 0;

    public boolean hasNext() {        return index < len;
    }

    public Integer next() {        return arr[index++];
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  };
}
```

```
  @Override
  public String toString() {
    if (len == 0) return "[]";
    else {
      StringBuilder sb = new StringBuilder(len).append("[");
      for (int i = 0; i < len - 1; i++) sb.append(arr[i] + ", ");
      return sb.append(arr[len - 1] + "]").toString();
    }
  }

  // Example usage
  public static void main(String[] args) {

    IntArray ar = new IntArray(50);
    ar.add(3);
    ar.add(7);
    ar.add(6);
    ar.add(-2);

    ar.sort(); // [-2, 3, 6, 7]

    // Prints [-2, 3, 6, 7]
    for (int i = 0; i < ar.size(); i++) System.out.println(ar.get(i));

    // Prints [-2, 3, 6, 7]
    System.out.println(ar);
  }
}
```