Problem 1

Write a class **SimpleArrayList**, objects of which represent vectors, i.e., "extensible arrays" of **int**s (similar to **ArrayList** in Java or **vector** in C++).
The class declares the following fields

- size — current number of elements in the vector;
- cap — current capacity (size) of the array which holds the elements; size of them are "true" elements which have been put there by the user and the remaining provide room for elements which will be put later (in this way we avoid allocating a new array every time we want to add an element);
- arr — reference to a ordinary array of size cap where the elements of the vector are held;
- INITIAL_CAPACITY — static constant determining the initial capacity of the vector (initial size is, of course, zero).

Define also member functions

- Default constructor, creating a vector of size (size) zero and the capacity (cap) equal to INITIAL_CAPACITY.
- Constructor taking one value of type **int** and creating a vector with this single element (size will be 1).
- Constructor taking an array of **int**s; its elements will become elements of the vector, size will be the number of these elements and the capacity must be appropriately chosen (see below).
- Constructor taking another vector of type **SimpleArrayList**: its elements will be elements of the vector being created.
- Method **size** returning the size (size) of the vector.
- Method **clear** „clearing" the vector; after this operation the state of the vector becomes identical to the state of an object created by the default constructor.
- Method **trim**: after calling this method the array arr has capacity equal to the current size.
- Method **insert** taking an index (say, ind) and an array of **int**s (say, other). The method:
  - throws an **IndexOutOfBoundsException** if the value of ind is larger than size or negative;
  - inserts elements from the array other (which is of size, say, sz) into the vector starting at position indicated by the index ind. If the current capacity is sufficient, existing elements of the vector from position ind are shifted to the right to make room for new elements from the array other. If the capacity is too small to accommodate old and new elements, a new array arr is allocated, with capacity *twice as big* as necessary, i.e., 2*(size+sz). Old and new elements are copied into this array at correct positions.

The method returns the reference to the object that it was invoked on (**this**).

- Methods

```
1    public SimpleArrayList insert(int ind, int e)
2    public SimpleArrayList append(int e)
3    public SimpleArrayList append(int[] a)
4    public SimpleArrayList append(SimpleArrayList a)
```

  which:

  - the first inserts a single element e at position ind;
  - the second appends a single element e at the end of the vector;
  - the third appends to the vector elements from the array passed as an argument;
  - the fourth appends to the vector elements from another vector passed as an argument.

  All these methods should be very short; they should just use the first version of **insert**, described above.

- Methods

```
public int get(int ind)
public SimpleArrayList set(int ind, int val)
```

  The first of them returns ind-th element of the vector, while the second modifies the element under index ind assigning the value val to it. Both throw **IndexOutOfBoundsException** for invalid values of the index ind.

- Method

```
public String toString()
```

  which returns a string representation of the vector (it will be called automatically when the reference to an object of the class is passed to method **println**).

Note: function **System.arraycopy** should be used for copying arrays or parts of arrays.

For example, the program

download *SimpleArrayList.java*

```
public class SimpleArrayList {
    private final static int INITIAL_CAPACITY = 5;
    private int cap   = INITIAL_CAPACITY;
    private int size  = 0;
    private int[] arr = new int[cap];

    // ... constructors and methods

    public static void main(String[] args) {
        SimpleArrayList a =
            new SimpleArrayList()
                .append(new int[]{1,3}).insert(1,2)
                .append(6).insert(3,new int[]{4,5});
```

```
            SimpleArrayList b = new SimpleArrayList(a);
            for (int i = 0; i < a.size(); ++i)
                a.set(i,a.get(i)+6);
            b.append(a).append(13).trim();
            System.out.println("a -> " + a);
            System.out.println("b -> " + b);
        }
    }
```

should print

```
a -> Cap=12, size=6: [ 7 8 9 10 11 12 ]
b -> Cap=13, size=13: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 ]
```

Do not use any classes except those in the package ***java.lang***.
