Section 8: Arrays, Java inbuilt Lists, Autoboxing and Unboxing

15 July 2021 19:23

Section 8: Arrays, Java inbuilt Lists, Autoboxing and Unboxing

Arrays : A data structure that allows you to store multiple values of the same type into a single variable.

Int [] myIntArray;
myIntArray= new int[10];

Or

Int[] myIntArray= new int[10]

Or

Int[] myNumbers = {1, 2, 3, 4, 5};

Or

anotherArray = new int[] {1, 2, 3, 4, 5};

Saving value = myIntArray[5] = 50;

Double[] myDoubleArray = new double[10];

/r = move curser to next line.

Printing array : System.out.println("array= " + Arrays.toString(array));

Static is Important : That means we don't have to create a new object instance for this class

Array List : It is a resizable array. (Looks like vector (c++))

Defining a new Array List : (of String Type)
private ArrayList<String> myArrayList= new ArrayList<String>();

// This will not work (not for primitive add type)
private ArrayList<int> myArrayList= new ArrayList<int>();

Instead use this :
private ArrayList<Integer> myArrayList= new ArrayList<Integer>();

Iterator : Iterator<String> iterator = list.iterator();

```
while(iterator.hasNext())
{
        System.out.println(iterator.next());
}
```

Autoboxing :

Interger.valueOf(i);   // converting the primitive type Integer to the Object wrapper, to the object in other words, that's autoboxing

Unboxing :

myIntValue.intValue();   // here we are suing .intValue so we are actually unboxing. We are convering it from the object, the object wrapper back into the primitive top of this case the int which is value

// Use psvm, shorcut for creating main() method declaration

Linked List : (It's doubly linked list)

LinkedList<String> list = new LinkedList<String>();
List Iterator : ListIterator<String> listIterator = list.listIterator();

listIterator.hasPrevious() and listIterator.previous() doesn't work directly. Need you use another variable to maintain going next or previous.

```
if(forward)
        {
                 if(listIterator.hasPrevious())
                 {
                    listIterator.previous();
                 }
                 forward = false;
        }
if(listIterator.hasPrevious())
              {
                 System.out.println("Output: " + listIterator.previous().toString());
              }
              else
              {
                 System.out.println("Output Else ");
                 forward = true;
              }
If(!forward)      // same for hasNext()
```