5COSC001W - OBJECT ORIENTED PROGRAMMING
Lecture 5: Introduction to Collections (ArrayLists) and Arrays

Dr Dimitris C. Dracopoulos

Collections

The Collections is a set of classes found in the `java.util` package. Compared with arrays, collection classes:

- Can hold a number of objects as elements (arrays can store both primitives and objects).
- They can have an unlimited number of objects.
- Although primitives cannot be stored directly, wrapper classes can be used to store the value of a primitive in a wrapper object. Such an object can be stored in a collection class.

The ArrayList class

An example of a class belonging to Java Collections. An unlimited number of objects can be stored in an `ArrayList`.

Example:

```
import java.util.*;

public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> al = new ArrayList<String>();

        // Add three elements in the list
        al.add("aa");
        al.add("bb");
        al.add("ccc");


        for (int i=0; i < al.size(); i++) {
            String s = al.get(i);
            System.out.println(s);
        }
```

```
        // remove second element from the list
        al.remove(1);

        System.out.println("After remove(), al contains:");
        for (String e : al)
            System.out.println(e);
    }
}
```

When the above program is run, it displays:

```
aa
bb
ccc
After remove(), al contains:
aa
ccc
```

Wrappers and Storing Numbers in Collections

Primitives cannot be stored directly in a collection class. The Java library contains wrapper classes which correspond to primitives as they are capable to store a primitive value.

```
Double d1 = new Double(3.1);
Double d2 = 3.1;  //automatically creates a Double object from 3.1
```

Example:

```
import java.util.*;

public class WrapperExample {
    public static void main(String[] args) {
        // create an ArrayList object storing Double objects
        ArrayList<Double> a = new ArrayList<Double>();

        Double d1 = new Double(5.4);
        a.add(d1);
        a.add(11.2);    // autoboxing occurs
                        // (double -> Double conversion)

        //a.add(new Integer(2));  // Error!


        // get second element from arraylist
        Double d2 = a.get(1);

        // get 1st element - unboxing occurs
                                // (Double -> double conversion)
        double d3 = a.get(0);

        System.out.println("d2=" + d2 + ", d3=" + d3);
    }
}
```

When the above example is run, it displays:

```
d2=11.2, d3=5.4
```

Non-parameterised ArrayLists

Although non-recommended, an `ArrayList` object can be declared to store objects of any type. In such cases, explicit casting is required when obtaining an element from the list.

```
import java.util.*;

public class ArrayListExample2 {
    public static void main(String[] args) {
        ArrayList l1 = new ArrayList();

        l1.add(new Integer(11));
        l1.add(new Integer(3));
        l1.add(new Integer(55));


        for (int i=0; i < l1.size(); i++) {
            Integer k1 = (Integer) l1.get(i); // Cast is required!
            System.out.println(k1);
        }
    }
}
```

Arrays

A constant number of primitive types or objects can be stored in an array.

- Java arrays are objects (they are allocated in the heap).

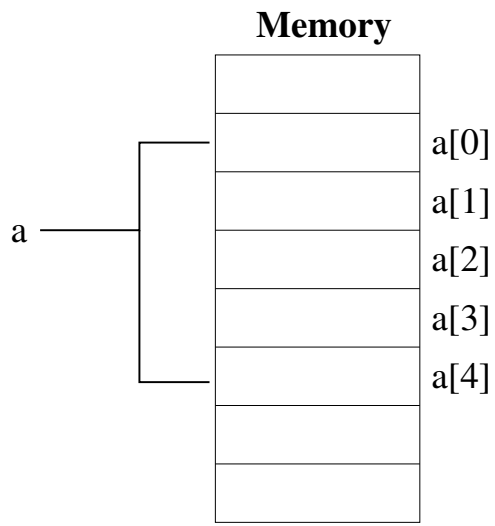
Declaring an array

```
int a[];
```

Creating an array object

```
a = new int[5];
```

The size of an array cannot be changed once it is created.




```
int a[] = new int[5];
```

Example:

```
/**
    A class to simulate the operation of a lottery
*/
public class Lottery {
    int results[];

    /**
        Constructs a lottery object with empty results
    */
    public Lottery() {
        results = new int[6];
    }
```

```
/**
    Simulates the lottery draw by filling in array results.
    The random generator should be called normally, but
    this class demonstrates arrays so for simplicity numbers
    are fixed.
*/
public void draw() {
    results[0] = 11;
    results[1] = 45;
    results[2] = 3;
    results[3] = 24;
    results[4] = 12;
    results[5] = 31;
}

/**
    Prints on the screen the latest draw results
*/
public void printResults() {
    System.out.println("The latest lottery results are:");
    for (int i=0; i < results.length; i++)
        System.out.print(results[i] + " ");

    System.out.println();
}
```

```
    public static void main(String[] args) {
        Lottery lot = new Lottery();
        lot.draw();
        lot.printResults();
    }
}
```

When the above program is run, it displays:

```
The latest lottery results are:
11 45 3 24 12 31
```

Initialising Arrays

There are two ways to initialise an array:

- Assign a value to each element individually.

```
double b[] = new double[10];
b[0] = 5.0;
b[1] = 1.2;
```

- Use an array initialiser at the point of declaration:

```
String weekdays[] = {"Mon", "Tue", "Wed", "Thu", "Fri"};
```

or

```
String weekdays[] = new String[] {new String("Mon"),
                                  new String("Tue"),
                                  new String("Wed"),
                                  new String("Thu"),
                                  new String("Fri")};
```

Two Dimensional Arrays (Arrays of Arrays)

A 2-dimensional array in Java is an array of an array.

```
Book mybooks [] [];
mybooks = new Book[10][12];  // an array[10] of array[12]
```

Because object declarations as the one above, do not create objects, an array of an array must create the elements in it, before using the array:

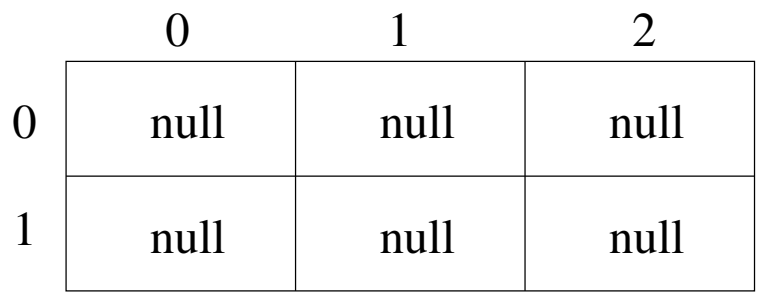
```
mybooks[i][j] = new Book();
```

This is also illustrated in the example below:

```
class Book {
    String colour;
}


public class ArrayExample {
    public static void main(String[] args) {
      Book mybooks [] [] = new Book[10][12];  // an array[10]
                                              // of array[12]
        System.out.println(mybooks[0][0]);

        mybooks[0][0] = new Book();
        System.out.println(mybooks[0][0]);
    }
```

When the program is run it prints:

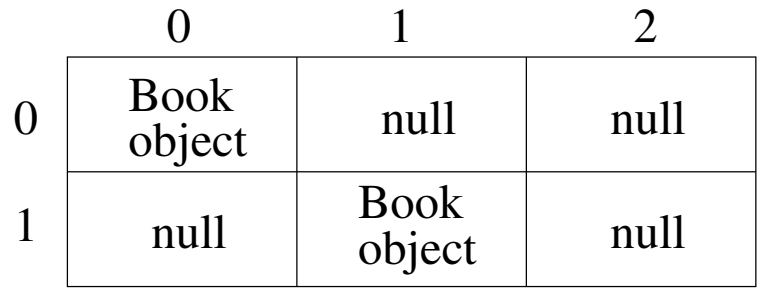
```
null
Book@f6a746
```



	0	1	2
0	null	null	null
1	null	null	null

(a) After:

```
Book m [] [] = new Book[2][3];
```



	0	1	2
0	Book object	null	null
1	null	Book object	null

(b) After:

```
m[0][0] = new Book();
m[1][1] = new Book();
```

Looping over Arrays - The for-each loop

```
public class ArrayExample2 {
    public static void main(String[] args) {
        String a[] = new String[3];
        a[0] = "aa";
        a[1] = "bb";
        a[2] = "cc";

        System.out.println("Array a contains:");
        for (String i : a)
            System.out.println(i);


        /* an array of an array containing different number
           of elements in each row */
        int myNumbers[][] = new int[][] {
                                    {0},
                                    {0,1},
                                    {0,1,2},
                                    {0,1,2,3}};
         System.out.println("\nArray myNumbers contains:");
```

```
        for (int[] r : myNumbers) {  // for each row
            for (int c : r) { // for each element in current row
                System.out.print(c + " ");
            }
            System.out.println();
        }
    }
}
```

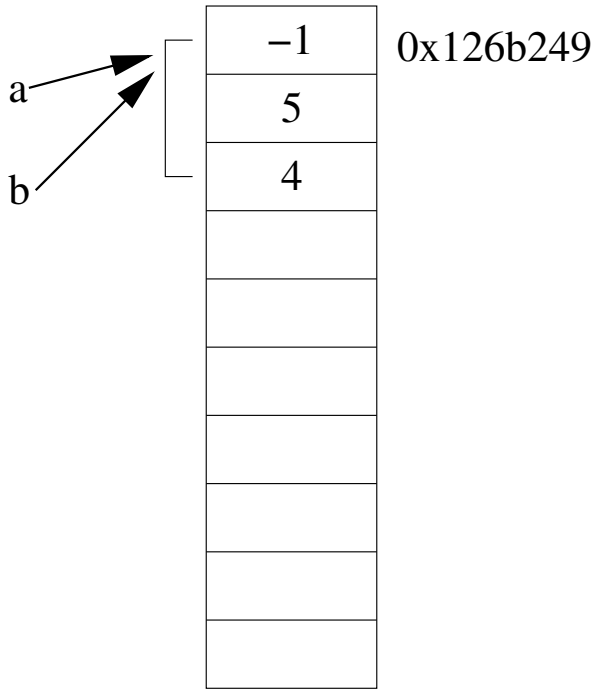
The above code displays:

```
Array a contains:
aa
bb
cc

Array myNumbers contains:
0
0 1
0 1 2
0 1 2 3
```

Another array example:

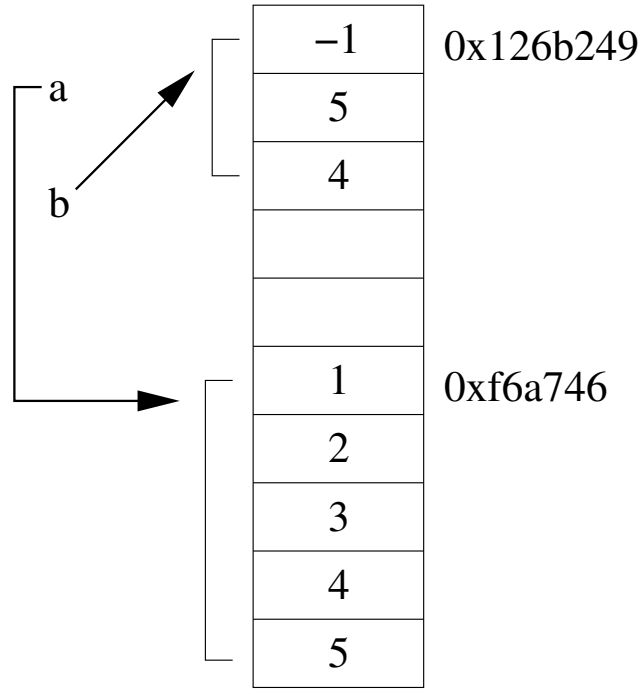
```
public class ArrayReferencesExample {
    public static void main(String[] args) {
        int a[] = new int[3];
        a[0] = -1;
        a[1] = 5;
        a[2] = 4;

        int b[];
        b = a;
        System.out.println("a is located at address " + a +
                           ", b is located at address " + b);

        a = new int[5];
        for (int i=0; i < 5; i++)
            a[i] = i+1;

        System.out.println("After a = new int[5]");
        System.out.println("a is located at address " + a +
                           ", b is located at address " + b);
```

```
        System.out.println("\n a contains: ");
        for (int n : a)
            System.out.print(n + " ");

        System.out.println("\n b contains: ");
        for (int n : b)
            System.out.print(n + " ");
        System.out.println();  // add a newline
    }
}
```



a) After:

```
int a[] = new int[3];
a[0] = -1;
a[1] = 5;
a[2] = 4;

int b[];
b = a;
```



b) After:

```
a = new int[5];
for (int i=0; i < 5; i++)
    a[i] = i+1;
```

The Arrays class

The `Arrays` library class located in package `java.util`, provides a number of useful utilities to manipulate arrays.
These include:

- Fill parts (or the whole) of the array with values.
- Compare arrays element by element
- Search.
- Sort.