Arrays

- Arrays are a simple data structure
- Arrays store a set of values of the same type
  - Built-in types (int, double, etc.) or
  - Objects (Students, Engines, etc.)
- Arrays are part of the Java language
  - Arrays are objects, not primitives like int or double.
  - They are declared in the same way as other objects
    ```java
    int[] intArray= new int[20];  //Irregular verb'
    ```
  - The array object has an int data member, length, that gives the number of elements in the array:
    ```java
    int aSize= intArray.length;  // aSize= 20
    ```
- Each value is accessed through an index
  ```java
  intArray[0]= 4; intArray[1]= 77;
  ```
Arrays, p.2

- Array index always starts at 0, not 1
  - An array with N slots has indices 0 through N-1
  - intArray has elements intArray[0] through intArray[19]
- Array lengths cannot be changed once they are declared
- Arrays can be initialized when declared
  int[] intArray= {5, 77, 4, 9, 28, 0, -9};
  // 'new' is implicit (not needed) in this case
- Arrays of numerical values are zero when constructed

Copying arrays

- To copy an array, use arraycopy( ) method of System class:
  int[] intArray= new int[20];  // Same as first slide
  int[] newArray= new int[intArray.length]

  // arraycopy(fromArray, fromIndex, toArray, toIndex, count)
  System.arraycopy(intArray, 0, newArray, 0, intArray.length);

  // Now intArray and newArray have separate copies of data

  // Arrays don't have to be same length as long as segment
  // copied fits into destination array
Copying entire array

```java
int[ ] intArray= { 5, 77, 4, 9, 28, 0, 9 };  
int[ ] newArray = new int[ intArray.length ];  
System.arraycopy(intArray, 0, newArray, 0, intArray.length)
```

Copying an array reference

```java
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};  
int[ ] newArray = intArray;
```
Looping Over Arrays

- If `doubleArray` is a reference to an array of doubles, there are two ways to iterate over it.
  - This way gives more control—you can loop over just part of it, and you know what element (i) is being computed:

    ```java
    double sum = 0.0;
    for (int i = 0; i < doubleArray.length; i++)
        sum += doubleArray[i];
    ```

  - This way is simpler but you can only iterate over the entire array, and you don’t have a loop counter (e.g., i):

    ```java
    double sum = 0.0;
    // “for each” double d in array doubleArray
    for (double d : doubleArray)
        sum += d;
    ```

Test Your Knowledge

1. Which of the following expressions does not declare and construct an array?
   a. `int[] arr = new int[4];`
   b. `int[] arr;`  
      `arr = new int[4];`
   c. `int[] arr = {1,2,3,4};`
   d. `int[] arr;`

2. Given this code fragment:
   ```java
   int j = ?;
   int[] data = new int[10];
   System.out.print(data[ j ]);  
   ```
   Which of the following is a legal value of j?
   a. -1
   b. 0
   c. 1.5
   d. 10
3. Given this code fragment:

```java
int[] arrayA = new int[4];
int[] arrayB;
arrayB = arrayA;
arrayA[0]=arrayB[2];
```

What are the values of the elements in array A?

- a. unknown
- b. 0,0,0,0
- c. 4,0,4,0
- d. 4,0,0,0

4. How many objects are present after the following code fragment has executed?

```java
double[] arrayA=new double[10];
double[] arrayB;
arrayB = arrayA;
```

- a. 1
- b. 2
- c. 10
- d. 20

5. For which of these applications an array is NOT suitable?

- a. Holding the scores on 4 quarters of a Basketball game
- b. Holding the name, account balance and account number of an individual
- c. Holding temperature readings taken every hour through a day
- d. Holding monthly expenses through a year

6. Given the following code fragment:

```java
int[] data = {1,3,5,7,11};
for(__________) System.out.println(data[index]);
```

Fill in the blanks so that the program prints out every element in the array in order

- a. int index = 4; index>0; index--
- b. int index=0; index<4; index++
- c. int index=0; index<data.length(); index++
- d. int index=0; index<data.length; index++
Test Your Knowledge

7. What is the output of the following program?

```java
public class Test{
    public static void main ( String[] args ){
        int value = 10;
        int[] arr = {10,11,12,13};
        System.out.println("value before:"+value);
        alterValue( value );
        System.out.println("value after:"+value);
        System.out.println("arr[0]after:"+arr[0]);
        alterArray( arr );
        System.out.println("arr[0] after:"+arr[0]);
    }
    public static void alterValue (int x ){
        x = 0;  }
    public static void alterArray (int[] a){
        a[0] = 0;  }
}
```

d. value before:10
   value after:0
   arr[0] before:10
   arr[0] after: 0

c. value before:10
   value after:10
   arr[0] before:10
   arr[0] after: 10

b. value before:10
   value after:10
   arr[0] before:10
   arr[0] after: 10

a. value before:10
   value after:10
   arr[0] before:10
   arr[0] after: 10

Exercise

- Create a TemperatureTest class
- Write a main() method to:
  - Declare and construct an array of doubles, called `dailyTemp` holding daily temperature data
    - Use an initializer list with curly braces
      - Mon Tue Wed Thu Fri Sat Sun
      - 70  61  64  71  66  68  62
  - Using a for loop, print every element of the `dailyTemp` array in order
    - Use the array length, not the constant 7, to control the loop if you use the “full control” version
    - Or use the simpler style of for loop
Exercise, p.2

• In class TemperatureTest, write a static method to find average weekly temperature:
  public static double average(double[] aDouble) {
    // Declare a total variable, initialize it to 0
    // Loop thru aDouble and add each element to the total
    // Use the simple \textit{for (double d : aDouble)} for loop
    // Divide by the number of elements, return the answer
  }

• In the main() method, call the average method you just wrote
  – Pass the dailyTemp array as the argument
  – Print the average temperature in main() as:
    • Average weekly temperature: 66

ArrayList Class

• The ArrayList class IS COMPLETELY DIFFERENT THAN an array.
  – It’s a more flexible way to store data
  – ArrayList can grow automatically as needed
    • Has capacity that is increased when needed
    • Has \texttt{size()} method that returns actual number of elements in the ArrayList
  – ArrayList can hold elements of different types
    • As long as each is an Object (reference),
    • Technically an ArrayList can’t hold a basic type (int, double, etc.)
    • But, conversion of primitive to an object happens automatically. This is called “auto-boxing”.
    • Wrapper classes allow objects (e.g., Boolean or Double) that hold basic types (e.g. boolean or double)
  – Think airplane vs bicycle as two ways to get from A to B
    • Bicycle is simple, airplane is complex, though both get you there
    • Squeeze hand brake doesn’t apply to plane, adjust flaps to bicycle
    • So it is with arrays and ArrayLists: similar but quite different
ArrayLists

• ArrayList class is not in the core Java language
  – It is in package java.util, which you must import:
    import java.util.*;  // At top of program
• ArrayLists are slightly slower than arrays
  – This matters only in large numerical applications
• ArrayList class has many methods that provide functionality beyond what arrays provide
• You can declare an ArrayList as containing objects of a particular type. Example:
  ArrayList<Point> pList = new ArrayList<Point>( );

Some Methods of ArrayList

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean add (Object o)</td>
<td>Adds object to end, increases size by one. Always returns true</td>
</tr>
<tr>
<td>void add(int i, Object o)</td>
<td>Inserts o at index i moving subsequent elements to right</td>
</tr>
<tr>
<td>Object get(int i)</td>
<td>Returns object at index i</td>
</tr>
<tr>
<td>int indexOf(Object o)</td>
<td>Finds first occurrence of object; -1 if not found</td>
</tr>
<tr>
<td>boolean isEmpty( )</td>
<td>Returns true if ArrayList has no objects, false otherwise</td>
</tr>
<tr>
<td>void remove (int i)</td>
<td>Deletes obj at index i moving subsequent elements leftward</td>
</tr>
<tr>
<td>void remove (Object o)</td>
<td>Deletes first occurrence of o moving subsequent elements leftward</td>
</tr>
<tr>
<td>void set(int i, Object o)</td>
<td>Sets element at index i to be the specified object</td>
</tr>
<tr>
<td>int size( )</td>
<td>Returns size of ArrayList</td>
</tr>
</tbody>
</table>
ArrayList Example

```java
import java.awt.*;      // to use Point class
import java.util.*;     // to use ArrayList class

public class ArrayListTest {
    static final int M = 100;   // Max coordinate

    public static void main(String args[]) {
        Random r = new Random();
        int numPoints = r.nextInt(20);  // Max 20 points
        ArrayList<Point> points = new ArrayList<Point>();

        for (int i = 0; i < numPoints; i++) {
            Point p = new Point(r.nextInt(M), r.nextInt(M));
            points.add(p);
        }

        System.out.println("ArrayList size: " + points.size());
        for (Point pt : points)
            System.out.println(pt);
    }
}
```

Automatic conversion of primitives to objects

- Java has “boxing” and “unboxing”:
  - When necessary, the compiler converts a primitive (e.g., int or double) to the corresponding object type (e.g., Integer or Double)
- This lets us add primitive types to an ArrayList:
  ```java
  ArrayList<Integer> myAList = new ArrayList<Integer>();
  myAList.add(1);   // 1 is an int; it's boxed
  myAList.add(3);   // same as myAList.add(new Integer(3));
  myAList.add(7);
  // retrieves Integer, unboxes to int
  int iValue = myAList.get(1);
  ```
Test Your Knowledge

1. Which of the following statements is NOT true about ArrayLists?
   a. ArrayLists are slightly faster than arrays.
   b. ArrayLists can store elements of different types.
   c. ArrayLists can increase in size to store more elements.
   d. ArrayLists have methods to manage their content.

Test Your Knowledge

2. Given the following code fragment:
   
   ```java
   ArrayList<String> myArrayList = new ArrayList<String>();
   myArrayList.add("One");
   myArrayList.add("Two");
   myArrayList.add("Three");
   myArrayList.add("Four");
   ```

   Which of the following expressions will modify myArrayList so it looks like:
   One; Two; Four
   a. myArrayList.remove (myArrayList.get(3));
   b. myArrayList.remove (myArrayList.indexOf("Three"));
   c. myArrayList.remove ("Three");
   d. myArrayList.remove (myArrayList.get(2));
Test Your Knowledge

3. Given the following code fragment (same as question 2):
   ```java
   ArrayList<String> myArrayList = new ArrayList<String>();
   myArrayList.add("One");
   myArrayList.add("Two");
   myArrayList.add("Three");
   myArrayList.add("Four");
   ```

Which of the following expressions will modify myArrayList so it looks like:
One; Two; Three; Five

a. myArrayList[3] = "Five"
b. myArrayList[4] = "Five"
c. myArrayList.set(myArrayList.indexOf("Four"), "Five");
d. myArrayList.set(myArrayList.indexOf("Five"), "Four");

Test Your Knowledge

4. Given the following code fragment:
   ```java
   ArrayList<Integer> myArrayList = new ArrayList<Integer>();
   myArrayList.add(1);
   myArrayList.add(3);
   myArrayList.add(7);
   ```

Which of the following expressions will modify myArrayList so it looks like:
1 3 5 7

a. myArrayList.add (5);
b. myArrayList.add (2, 5);
c. myArrayList.add (4, 5);
d. myArrayList.add (3, 5);
Arrays and ArrayLists

**Array**
- Capacity fixed at creation
- Accessed with z[i]
- Constructor: new double[30]
- One data member: z.length
- No methods

**ArrayList**
- Capacity increases as data is added
- Accessed with z.get(i)
- Constructor: new ArrayList<Bus>();
- No data members

- Many methods – z.size(), z.add(), z.get()...
- More flexible

---

**Exercise**

- Create class CourseTest:
  - import java.util.*; // 1st line in CourseTest
  - In main():
    - Create an ArrayList<String> students
    - Add 4 students to the ArrayList:
      - “Amy”, “Bob”, “Cindy” and “David”
      - Add them to the ArrayList directly:
        students.add("Amy");
    - Write method to print elements in the ArrayList and its size
      public static void printOutArrayList(// Argument) {
        // Code goes here
      }
    - Call printOutArrayList() method from main()
      - Pass the ArrayList as the argument
    - Your output should be:
      Amy
      Bob
      Cindy
      David
      Size: 4