ARRAYS OF OBJECTS

When the component type of an array is a class (rather than a primitive data type), the array is an array of references. To build such an array, you must build each object separately and store its reference into the array.

**Example**
Line 1 allocates the reference variable `name`, line 2 allocates the 3-element array and lines 3-5 allocate the `String` objects and store their references into the array.

```
1  String [] name;
2  name = new String[3];
3  name[0] = new String( "Tom" );
4  name[1] = new String( "Dick" );
5  name[2] = new String( "Harry" );
```

**Example**
```
1  Double [] v = new Double[4];
2  v[0] = new Double( 1.25 );
3  v[1] = new Double( 1.75 );
4  v[2] = new Double( 2.25 );
5  v[3] = new Double( 2.75 );
```
**Example**

Given the `Die` class shown below, the following code builds an array of 5 `Die` objects.

```java
Die [] dice = new Die[5];
dice[0] = new Die( );
dice[1] = new Die( );
dice[2] = new Die( );
dice[3] = new Die( );
dice[4] = new Die( );
```

In the `Die` class given below, I’ve added a `toString` method so that the jGRASP Viewer can display the contents of the array as shown to the right above.

```java
// A Die object models a 6-sided die used for dice games.
public class Die
{
    public int faceUp;   // number shown facing up
    public void roll( )  // roll the die
    {
        // randomly select a number from 1 to 6
        faceUp = (int)(Math.random( ) * 6 + 1);
    }
    public String toString( )
    {  // return the die value as a string
        return "faceUp | " + faceUp + " |";
    }
}
```
**Initializing an Array of Objects**
An array of objects can be declared and initialized using the brace ({} ) delimited syntax.

<table>
<thead>
<tr>
<th>Example</th>
<th>Array name, holding String objects, pictured in a previous example, can be built using this statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>String [ ] name = { new String(&quot;Tom&quot;), new String(&quot;Dick&quot;), new String(&quot;Harry&quot;) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Array v, holding Double objects, pictured in a previous example, can be built using this statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Double [ ] v = { new Double( 1.25 ), new Double( 1.75 ), new Double( 2.25 ), new Double( 2.75 ) }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Array dice, holding Die objects, pictured in a previous example, can be built using this statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Die [ ] dice = { new Die( ), new Die( ), new Die( ), new Die( ), new Die( ) }</td>
</tr>
</tbody>
</table>


String Arrays
Recall that the Java compiler automatically builds a **String** object when it encounters a **String** literal. This feature also works for arrays.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array <strong>name</strong>, holding <strong>String</strong> objects, pictured in a previous example, can be built using this statement.</td>
</tr>
</tbody>
</table>

```java
String [] name = { "Tom", "Dick", "Harry" }; 
```

Wrapper Class Arrays
Recall that Java automatically boxes primitive data inside an object of a wrapper class. This feature also works for arrays.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array <strong>v</strong>, holding <strong>Double</strong> objects, pictured in a previous example, can be built using this statement.</td>
</tr>
</tbody>
</table>

```java
Double [] v = { 1.25, 1.75, 2.25, 2.75 }; 
```

Programming Example

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let’s write an applet that new parents can use to help choose a baby name. For now, we’ll stick with boy’s names. We need a large array full of boy’s first and middle names:</td>
</tr>
</tbody>
</table>

```java
```

The program asks the user to enter his or her last name, selects first and middle names from the array at random and displays the full name for the user’s inspection.
Once the user clicks OK or Cancel, the program prints the next name selection and continues to do so until the browser window is closed.

The program and its accompanying HTML file are shown below.

```
public class BabyNames extends Applet {

    public String [] boys = {
        "JAMES", "JOHN", "ROBERT", "MICHAEL", 
        "WILLIAM", "DAVID", "RICHARD", "CHARLES", "JOSEPH", "THOMAS", 
        "CHRISTOPHER", "DANIEL", "PAUL", "MARK", "DONALD", "GEORGE", 
        "KENNETH", "STEVEN", "EDWARD", "BRIAN", "RONALD", "ANTHONY", 
        "KEVIN", "JASON", "MATTHEW", "GARY", "TIMOTHY", "JOSE", "LARRY", 
        "JEFFREY", "FRANK" 
    };

    public void init() {
        int f, m;  // random numbers for name selection
        String lastName, fullName; // input and output names
        lastName = showInputDialog( "What is your last name?" );
        lastName = lastName.toUpperCase( ); // capitalize it
        // show names until user closes applet
        while ( true ) {
            f = (int)(Math.random( ) * 31); // select first name
            m = (int)(Math.random( ) * 31); // select middle name
            fullName = boys[f] + " " + boys[m] + " " + lastName;
            showMessageDialog( null, fullName );
        }
    }
}
```
<html>
<body align="center" bgcolor="#00FFBF" style="color:#0B3861">

    BABY NAMES

<applet code="BabyNames.class" width="300" height="50">
</applet>

<p>Enter your last name to see a selection of baby names.</p>
<p>Browse each name by clicking OK.</p>
<p>To stop, close the browser window.</p>

</body>
</html>

Exercises

For each, draw a picture of the allocated memory.

1. String [] d = { "June", "July", "August" };

2. String [] suit = {"HEARTS", "SPADES", "DIAMONDS", "CLUBS"};

3. Double [] vtr = { 0.125, 0.250, 0.375, 0.500, 0.625 }; 

4. Integer [] even = { 0, 2, 4, 6, 8, 10 }; 

5. Character [] vowels = {'a', 'e', 'i', 'o', 'u'};
Given class **Point** below, for each of the following problems, draw a picture of the memory allocated.

```java
public class Point {
    public int x, y; // (x,y) coordinates

    // constructor
    public Point(int xa, int ya) {
        x = xa;
        y = ya;
    }

    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
```

6. `Point[] square = {new Point(1, 1), new Point(1, 6),
                        new Point(6, 6), new Point(6, 1)};`

7. `Point[] triangle = new Point[3];
       triangle[0] = new Point(20, 40);
       triangle[1] = new Point(10, 10);
       triangle[2] = new Point(30, 10);`
For each of the code fragments below, circle what’s wrong and explain. None of them is correct.

<table>
<thead>
<tr>
<th></th>
<th>Code Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.</td>
<td><code>Point line = new Point[2];</code></td>
</tr>
<tr>
<td>9.</td>
<td><code>Point [] line = new Point[];</code></td>
</tr>
</tbody>
</table>
| 10. | `Point [] line;`  
    | `line[0] = new Point( 1, 1 );`  
    | `line[1] = new Point( 6, 6 );` |
| 11. | `Point [] line = new Point[1];`  
    | `line[0] = new Point( 1, 1 );`  
    | `line[1] = new Point( 6, 6 );` |
| 12. | `Point [] line = new Point[2];`  
    | `line[0] = 1;`  
    | `line[1] = 6;` |
| 13. | `Point line = { new Point( 1, 1 ), new Point( 6, 6 ) };` |
| 14. | `Point [] line = { Point( 1, 1 ), Point( 6, 6 ) };` |
| 15. | `Point [] line = { 1, 1, 6, 6 };` |

For each, assume this array and give the output.

```java
String [] name = { "Richard", "Nixon" };```

<table>
<thead>
<tr>
<th></th>
<th>Code Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.</td>
<td><code>System.out.println( name[0] + &quot; Milhous &quot; + name[1] );</code></td>
</tr>
<tr>
<td>17.</td>
<td><code>System.out.println( name[0].length( ) );</code></td>
</tr>
<tr>
<td>18.</td>
<td><code>System.out.println( name.length );</code></td>
</tr>
<tr>
<td>19.</td>
<td><code>System.out.println( name[0].charAt(0) + &quot;. &quot; + name[1] );</code></td>
</tr>
</tbody>
</table>

21. Write the Java statement to declare and initialize an array of strings containing the names “Obama”, “Bush” and “Clinton”.

22. Write the Java statement to declare and initialize an array of `Point` objects containing the points (1, 5), (1, 7), (10, 7), (10, 5).
<table>
<thead>
<tr>
<th>Exercise</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.</td>
<td>This exercise requires an <code>if</code> statement. Extend the <strong>BabyNames</strong> applet so that it asks the user whether he or she wants boys or girls names. The program must respond appropriately to the user’s choice.</td>
</tr>
<tr>
<td>24.</td>
<td>This exercise requires a <code>while</code> statement. Extend the <strong>BabyNames</strong> applet so that it never displays the same name for both first and middle names.</td>
</tr>
</tbody>
</table>