

> Unit Summary and Outcomes

In this unit, students will learn the fundamentals of arrays: declaration, population, and usage. To illustrate arrays, this unit will employ a “Madlibs” example, whereby arrays of nouns, verbs, and adjectives will be compiled and used randomly to create stories.

> Student Assessment

How well do students perform each of the following tasks:

- Code and compile the `Madlibs` class
- Add an array of colorful adjectives for program use
- Add a new `getAdj()` method to supply random adjectives
- Modify `tellStory()` to enhance the story by increasing the nouns, verbs, and adjectives used in the story

> Prerequisite Knowledge and Skills

BlueJ:

- Creating projects
- Adding new classes
- Instantiating objects
- Testing methods from instantiated objects

Java:

- Understanding class structure and methods
- Understanding arrays
- Understanding randomization

> Subject Areas(s): This Lesson Targets the Following

Pre-Java	Java Programming
<input type="checkbox"/> Hardware Basics	<input type="checkbox"/> Applet Programming
<input checked="" type="checkbox"/> Software Basics	<input type="checkbox"/> Subroutine Programming
<input type="checkbox"/> Networks and Servers	<input checked="" type="checkbox"/> Full Scale Programming
<input type="checkbox"/> HTML	
<input type="checkbox"/> Action Scripting	
<input type="checkbox"/> Java Scripting	

> Curriculum-Framing Questions

Essential Question	<ul style="list-style-type: none"> • What is an array? • What is the advantage of using arrays over individual variables?
Unit Questions	<ul style="list-style-type: none"> • How are arrays declared and populated in this exercise? • How else can arrays be populated?
Sample Content Questions	<ul style="list-style-type: none"> • What is the purpose of the call to <code>Math.random()</code>? • Can the arrays in this exercise be different lengths? • What could be added to the story to make it more dynamic?

> Targeted Content Standards, Benchmarks, or State Frameworks

Colorado State Standards:

Mathematics:

- **Standard 2** – Students use algebraic methods to explore, model, and describe patterns and functions involving numbers, shapes, data, and graphs in problem-solving situations and communicate the reasoning used in solving these problems.
- Grades 9–12 – Analyzing and explaining the behaviors, transformations, and general properties of types of equations and functions.
- **Standard 6** – Students link concepts and procedures as they develop computational techniques, including estimation, mental arithmetic, paper and pencil, calculators, and computers, in problem-solving situations and communicate the reasoning used in solving these problems. In order to meet this standard, a student will develop, use, and analyze algorithms.

International Society of Technology Education (ISTE):

Information Literacy Standards:

- **Standard 6** – Mathematics instructional programs should focus on solving problems as part of understanding mathematics so that all students
 - Build new mathematical knowledge through their work with problems
 - Develop a disposition to formulate, represent, abstract, and generalize in situations within and outside mathematics
 - Apply a wide variety of strategies to solve problems and adapt the strategies to new situations
 - Monitor and reflect on their mathematical thinking in solving problems

Technology Foundation Standards for Students:

- **Standard 3** – Students use technology tools to enhance learning, increase productivity, and promote creativity. Students use productivity tools to collaborate in constructing technology-enhanced models, prepare publications, and produce other creative works.

- Standard 6 – Students use technology resources for solving problems and making informed decisions. Students employ technology in the development of strategies for solving problems in the real world.

> Materials and Resources Required for Lesson

Equipment	<ul style="list-style-type: none"> • Computer for each student with BlueJ and Java SDK installed
Consumable Supplies	<ul style="list-style-type: none"> • Diskettes • Printed output for prototype
Textbooks/Lesson Guides	<ul style="list-style-type: none"> • <i>Objects First With Java: A Practical Introduction to BlueJ</i> by David J. Barnes and Michael Kölling. Prentice Hall, 2003 • Handouts of lessons
Technology	<ul style="list-style-type: none"> • Windows-based machines running Windows 98 SE or higher
Internet Resources	<ul style="list-style-type: none"> • http://java.sun.com/j2se/1.5.0/docs/api/ (Look at <code>Math</code> class for explanation of <code>Random</code>)

> Lesson Plan Outline

Overview: A variable contains a single instance of a data type, while an array, a structure with a pre-defined number of elements in it, allows for a collection of related values. Each element is of the same type (for example, an `int`, `double`, or `String`), and it has a specific position in the array known as the index. An array is basically a one-dimensional grid. As with any other variable types, an array must be declared before it can be used. You can do so using any of the following formats.

```
type[] arrayName; // only declares the array

type[] arrayName = new type[size]; // declares and constructs all at
once

type[] arrayName = { val1, val2, ..., lastval }; // this fills in
values for arrayName
```

To illustrate, an array of days of the week could be declared in the following manner:

```
String daysOfWeek;

String[] daysOfWeek = new String[7];

String[] daysOfWeek = { "Sunday", "Monday", "Tuesday",
"Wednesday",
    "Thursday", "Friday", "Saturday" };
```

In each of the previous examples, an array of seven `Strings` is created. In the second case, the array is populated as shown in the following table.

index	value
0	"Sunday"
1	"Monday"
2	"Tuesday"
3	"Wednesday"
4	"Thursday"
5	"Friday"
6	"Saturday"

Each day of the week is represented as an element in the array, and it can be accessed or referred to by index. To set or use the value of one of the elements, you refer to the name of array variable and the element's index in square brackets. In the following code, the first line sets the value of the array's first element (index 0). The second line writes the value of the array's fourth element (index 3).

```

daysOfWeek[0] = "Sunday";

System.out.println(daysOfWeek[3]); // displays "Wednesday"

```

The Madlibs example will use similar collections of Strings.

To test the `Madlibs` example, first create a new project in BlueJ named `Madlib` and create a class also called `Madlib`. Enter the following code:

```

/**
 * Madlib.java creates a silly, random story from arrays of words.
 * @author (your name)

```

```
* @version (a version number or a date)
*
*/
public class Madlib
{
    // instance variables
    // array of nouns

    private String[] nouns = {"banana", "Ferrari", "hammer",
"guacamole",
                                "bug-eyed monster"};

    // array of verbs
    private String[] verbs = {"eat", "drive", "whack",
"transmogrify"};

    /**
     * Pick a noun at random from the array with a random index.
     * Remember indexes start at zero.
     */

    public String getNoun()
    {
        int size = nouns.length; // get the size of the array
        int index = (int)(Math.random() * size); // random # must be
< size

        return nouns[index]; // send back the noun at that index

    } // end of getNoun() method
}
```

```
/**
 * Pick a verb at random from the array with a random index.
 */

public String getVerb()
{
    int size = verbs.length;
    int index = (int)(Math.random() * size);
    return verbs[index];
} // end of getVerb() method

/**
 * Generate and display the madlib.
 * This is just a series of println's.
 */

public void tellStory()
{
    System.out.println("Once upon a time there was a " +
getNoun() +
        ".");
    System.out.println("All it could do was " + getVerb() + "
the " +
        getNoun());
    System.out.println("but they all " + getVerb() +
        " happily ever after. The end.");
}
```

```
} // end of tellStory() method

/**
 * Create a story from random nouns, verbs, adjectives
 */

public static void main(String[] args)
{

    Madlib story1 = new Madlib(); // create a madlib object
    System.out.println("Here is a story just for you\n");
    story1.tellStory(); // generate and display the story
} // end of main()
} // end of Madlib class
```

Next, compile and run the program, noting the resulting story.

Finally, add an array of adjectives and create a `getAdv()` method to randomly access them. Modify the `tellStory()` method to include adjectives.

Tips, Hints, and Tricks: Make the story more elaborate, using at least four each of nouns, verbs, and adjectives. Modify the program so that it runs for an unlimited number of messages, quitting only when the user opts to quit.

> Pacing/Timeline

- Code and debug basic `MadLib` class: 1 day
- Add an array of adjectives and a `getAdj()` method: 1 day
- Modify `tellStory()` method using all three components (noun, verb, adjective): 1 day
- Modify `main()` to allow the program to loop, continuing until the user opts to quit: 1 day

> Teacher Reflection (For example, what worked well in this lesson? What would you change if you were to teach it again?)

- Were students comfortable with the concept and implementation of arrays?
- Were they able to add the `getAdj()` method?
- Were students able to make the program loop?
- Did they have suggestions for making the program more interesting?

> Student Handout: Java Programming Arrays—A Madlibs Example

A variable contains a single instance of a data type, while an array, a structure with a pre-defined number of elements in it, allows for a collection of related values. Each element is of the same type (for example, an `int`, `double`, or `String`), and it has a specific position in the array known as the index. An array is basically a one-dimensional grid. As with any other variable types, an array must be declared before it can be used. You can do so using any of the following formats.

```
type[] arrayName; // only declares the array

type[] arrayName = new type[size]; // declares and constructs
all at once

type[] arrayName = { val1, val2, ..., lastval }; // this
fills in values for arrayName
```

To illustrate, an array of days of the week could be declared in the following manner:

```
String daysOfWeek;

String[] daysOfWeek = new String[7];

String[] daysOfWeek = {"Sunday", "Monday", "Tuesday",
"Wednesday",
"Thursday", "Friday", "Saturday"};
```

In each of the previous examples, an array of seven `Strings` is created. In the second case, the array is populated as shown in the following table.

index	value
0	"Sunday"
1	"Monday"
2	"Tuesday"
3	"Wednesday"
4	"Thursday"
5	"Friday"
6	"Saturday"

Each day of the week is represented as an element in the array, and it can be accessed or referred to by index. To set or use the value of one of the elements, you refer to the name of array variable and the element's index in square brackets. In the following code, the first line sets the value of the array's first element (index 0). The second line writes the value of the array's fourth element (index 3).

```

daysOfWeek[0] = "Sunday";
System.out.println(daysOfWeek[3]); // displays
"Wednesday"

```

Once you define an array's size, you cannot change it. You can modify the value of any element in the array, but the size is always fixed. Note this declaration:

```
String[] daysOfWeek = new String[7];
```

The array `daysOfWeek` has seven elements, each of which is a `String`. The indexes start at zero and go to six. Note that there are only null (empty) values in the array at this point; the elements' values must be set by the program. To set or use the value of one of the elements, you refer to the name of array variable and the element's index in square brackets. In the following code, the first line sets the value of the array's first element (index 0). The second line writes the value of the array's fourth element (index 3).

```

daysOfWeek[0] = "Sunday";
System.out.println(daysOfWeek[3]); // displays
"Wednesday"

```

To experiment with arrays in a program, create a project called Madlibs and create

within it a class named Madlib. Enter the following code into the Madlib class.

```
/**
 * Madlib.java creates a silly, random story from arrays of words.
 * @author (your name)
 * @version (a version number or a date)
 */
public class Madlib
{
    // instance variables
    // array of nouns

    private String[] nouns = {"banana", "Ferrari", "hammer",
"guacamole", "bug-eyed monster"};
    // array of verbs
    private String[] verbs = {"eat", "drive", "whack",
"transmogrify"};

    /**
     * Pick a noun at random from the array with a random index.
     * Remember indexes start at zero.
     */

    public String getNoun()
    {
        int size = nouns.length; // get the size of the array
        int index = (int)(Math.random() * size); // random # must
be < size
        return nouns[index]; // send back the noun at that index
    } // end of getNoun() method

    /**
     * Pick a verb at random from the array with a random index.
     */

    public String getVerb()
    {
        int size = verbs.length;
        int index = (int)(Math.random() * size);
        return verbs[index];
    } // end of getVerb() method

    /**
     * Generate and display the madlib.
     * This is just a series of println's.
     */

    public void tellStory()
```

```
{
    System.out.println("Once upon a time there was a " +
getNoun() + ".");
    System.out.println("All it could do was " + getVerb() + "
the " +
                                getNoun());
    System.out.println("but they all " + getVerb() +
                                " happily ever after. The end.");
} // end of tellStory() method

/**
 * Create a story from random nouns, verbs, adjectives
 */

public static void main(String[] args)
{
    Madlib story1 = new Madlib(); // create a madlib object
    System.out.println("Here is a story just for you\n");
    story1.tellStory(); // generate and display the story
} // end of main()
} // end of Madlib class
```

> YOUR JOB:

- Add a third `String` array containing at least five adjectives
- Add a `getAdj()` method to return a random adjective
- Return a random value from the corresponding array in each method, just as in `getAdj()`
- Modify `tellStory()` to include adjectives
- Add a loop so that the user can continue generating stories

> EXTRAS YOU CAN ADD:

- Get creative and make the program better. Add more arrays and methods to allow for more variation in the story. For example, experiment with adverbs, different verb tenses, numbers, and sound effects.
- Challenging: Allow the user to enter the word lists manually.

