

> Lesson Summary and Outcomes

In this lesson, students will learn the key methods used to manipulate `Strings`. Because `Strings` are so useful and so necessary to successful programming, it is imperative that students develop familiarity with these methods. To illustrate the use of `Strings`, this lesson will use a class called `Translate` to translate a simple message into shorthand by substituting symbols for some commonly used words.

> Student Assessment

How well do students perform each of the following tasks:

- Code and compile the `Translate` class
- Complete the `shorthand()` method with additional substitutions
- Modify `main()` to allow the program to loop until the user opts to quit
- Comment as needed, ensuring that each method has a description of what it does

> Prerequisite Knowledge and Skills

BlueJ:

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

Java:

- Understanding class structure and methods

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

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

- What is a `String`?
- Why is the `String` class so useful?

Activity Questions

- Which `String` class methods are used in this exercise?
- Which other methods could be used for this application?

Sample Content Questions

- What is the purpose of the `toLowerCase()` method?
- What does the `charAt()` method do?

> 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 or CDs • Graph paper
Textbooks/Lesson Guides	<ul style="list-style-type: none"> • <i>Objects First With Java: A Practical Introduction to BlueJ</i>, Second Edition by David J. Barnes and Michael Kölling Prentice Hall, 2004. • Handouts of lessons
Technology	<ul style="list-style-type: none"> • Windows-based machines running Windows 98 SE or higher

Internet Resources

- <http://java.sun.com/j2se/1.5.0/docs/api/>
- <http://cs.colgate.edu/APCS/Java/JavaUtilities/JavaUtilities.htm>

> Activity Outline

In this unit, students will learn about `String` class methods, focusing on `String` manipulation. To illustrate these methods, this unit will employ a `Translate` class to convert words into symbols. The most commonly used `String` methods are listed in the table at the top of the accompanying handout.

Begin by creating a BlueJ project called *Translate Strings* and within that project create a class called `Translate`. The code for that class is illustrated as follows:

```
/**
 * Translate.java
 * translates a phrase into something else
 * @author (Your name here)
 * @version (Today's date)
 */
import chn.util.*;
public class Translate
{
    /*
     * Method translates the string to "shorthand" form.
     * Replaces instances of "and"
     * by "&", "to" by "2", "you" by "U", "at" by "@",
     * and "for" by "4". Removes all vowels
     * ('a', 'A', 'e', 'E', etc.)
     */
    public String shorthand(String phrase)
    {
        String shorthandMsg = ""; // holds translated phrase
        phrase = phrase.replaceAll("and", "&"); // change
            // 'and' to '&'
        phrase = phrase.replaceAll("to", "2"); // change 'to'
            // to '2'
        int len = phrase.length(); // get current length of
            // the phrase
        for (int i = 0; i < len; i++) // loop through the
            // phrase one char at
            // a time
        {
            // check each char to see if it is not a vowel
            if (phrase.toLowerCase().charAt(i) != 'a' &&
                phrase.toLowerCase().charAt(i) != 'e')
            {
```

```

        shorthandMsg += phrase.charAt(i); // char
                                   // is consonant,
                                   // add to translated
    }
}
return shorthandMsg; // send back the translated
                    // phrase
} // end of shorthand() method
/**
 * Main method gets string to be translated
 */
public static void main(String[] args)
{
    String input = ""; // create an empty string for
                      // user input
    Translate output = new Translate(); // Translate
                                   // object for output
    ConsoleIO console = new ConsoleIO(); // enable
                                   // keyboard input
    System.out.println("Welcome to the UT9000 Universal
        Translator");
    System.out.println();
    System.out.print("Enter a message (q to quit): ");
    input = console.readLine(); // get the phrase
    // translate and show
    // translated message
    System.out.println(output.shorthand(input));
} // end of main() method
} // end of Translate class

```

Add the external package `chn.util` using the following instructions:

1. Create a directory specifically for packages or classes. For example, you might have a directory hierarchy that looks like `\java\utilities`.
2. Extend that hierarchy to reflect your new package. For instance, if your package is named `files.readwrite`, your hierarchy should be extended to read `\java\utilities\files\readwrite`.
3. Store your new packages in that directory, making sure your package is named `files.readwrite`.
4. Set your `CLASSPATH` environmental variable to include the path `\java\utilities`.
5. Make sure that all the classes in your package begin with the designation
6. package `files.readwrite`. Nothing should appear in the file above that line.
7. To include the new package, add into your program the import directive

8. `import files.readwrite.*; .`

Here are the steps for using packages in BlueJ:

9. Follow Steps 1–6 above.
10. From the BlueJ menu, select Tools→Preferences.
11. Select the Libraries Tab.
12. Click on the Add button on the right-hand side of the dialog box.
13. Navigate to the `.jar` or `.zip` file containing your package and select that file.
14. Click OK to confirm that addition and return to BlueJ.
15. Save your project and exit BlueJ so that the new `library / package` can load.
16. Return to BlueJ.
17. Select Tools→Preferences followed by the Libraries tab to verify that your `package / library` has been loaded.

Next, run the program, enter a message, and note the results.

Finally, modify the `shorthand()` method to translate all vowels.

Tips, Hints, and Tricks: Add some shorthand translations of your own and test them. Modify the program so that it can run for an unlimited number of messages, quitting only when the user opts to quit.

> Pacing/Timeline

- Code and debug basic `Translate` class: 1 day
- Modify `shorthand()` method to add new translations:
- Modify `main()` to allow the program to loop and continue until the user opts to quit:

> **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 many `String` methods?
- Were they able to complete the `shorthand()` method to translate other vowels?
- Were students able to make the program loop?

The String Method	Returns this value	And looks like this
<code>int length();</code>	number of characters	<code>int len = name.length();</code>
<code>int indexOf(String str);</code> <code>int indexOf(char ch);</code>	the position of the first occurrence of String str or char ch in the String	<code>int pos = name.indexOf("Rob");</code> <code>int pos = name.indexOf('o');</code>
<code>char charAt(int index);</code>	the individual char at the specified position	<code>char ch = name.charAt(7);</code>
<code>String toLowerCase();</code>	same String with all characters lower case	<code>String lower = name.toLowerCase();</code>
<code>String toUpperCase();</code>	same String with all characters upper case	<code>String upper = name.toUpperCase();</code>
<code>String substring(int begin);</code>	part of String beginning at position begin	<code>String part = name.substring(5);</code>
<code>String substring(int begin, int end);</code>	part of String from position begin to end-1	<code>String part = name.substring(1,4);</code>
<code>boolean equals(String str);</code>	true if the two Strings are exactly the same, else false	<code>Boolean same = name.equals(otherName);</code>
<code>boolean equalsIgnoreCase(String str);</code>	true if the two Strings match when ignoring upper and lower case	<code>Boolean same = name.equalsIgnoreCase(otherName);</code>
<code>String concat(String str);</code>	appends str to the current String	<code>String complete = first.concat(last);</code>
<code>String replaceFirst(String str1, String str2);</code>	replaces first occurrence of str1 with str2	<code>String replaced = name.replaceFirst(str1, str1);</code>
<code>String replaceAll(String str1, String str2);</code>	replaces all occurrences of str1 with str2	<code>String replaced = name.replaceAll(str1, str1);</code>
<code>String replace(char ch1, char ch2);</code>	replaces all occurrences of ch1 with ch2	<code>String replaced = name.replace(ch1, ch2);</code>



> Create a new project in BlueJ named Translate and within that project create a class called Translate. Enter the following code:

```
/**
 * Translate.java
 * translates a phrase into something else
 * @author (Your name here)
 * @version (Today's date)
 */
import chn.util.*;
public class Translate
{
    /**
     * Method translates the string to "shorthand" form.
     * Replaces instances of "and"
     * by "&", "to" by "2", "you" by "U", "at" by "@",
     * and "for" by "4". Removes all vowels
     * ('a', 'A', 'e', 'E', etc.)
     */
    public String shorthand(String phrase)
    {
        String shorthandMsg = ""; // holds translated phrase
        phrase = phrase.replaceAll("and", "&"); // change
            // 'and' to '&'
        phrase = phrase.replaceAll("to", "2"); // change 'to'
            // to '2'
        int len = phrase.length(); // get current length of
            // the phrase
        for (int i = 0; i < len; i++) // loop through the
            // phrase one char at
            // a time
        {
            // check each char to see if it is not a vowel
            if (phrase.toLowerCase().charAt(i) != 'a' &&
                phrase.toLowerCase().charAt(i) != 'e')
            {
                shorthandMsg += phrase.charAt(i); // char
                    // is consonant,
                    // add to translated
            }
        }
        return shorthandMsg; // send back the translated
            // phrase
    } // end of shorthand() method
} /**
 * Main method gets string to be translated
 */
public static void main(String[] args)
{
```

```

String input = "";    // create an empty string for
                    // user input
Translate output = new Translate(); // Translate
                    // object for output
ConsoleIO console = new ConsoleIO(); // enable
                    // keyboard input
System.out.println("Welcome to the UT9000 Universal
    Translator");
System.out.println();
System.out.print("Enter a message (q to quit): ");
input = console.readLine(); // get the phrase
// translate and show
// translated message
System.out.println(output.shorthand(input));
    } // end of main() method
} // end of Translate class

```

Create an instance of `Translate`, run the `main()` method, and run some test translations.

> YOUR JOB:

- Complete `shorthand()` with the remaining replacements (notice that the comment suggest replacing “you” with “U”, “at” with “@” and “for” with “4”) and vowel removals (notice that the code removes all occurrences of ‘a’ and ‘e’ – students should also remove ‘i’, ‘o’ and ‘u’)..
- Modify `main()` to allow the user to continue entering phrases or to enter a single `q` to quit.
- Comment where needed, and save, compile, and test OFTEN!

> EXTRAS YOU CAN ADD:

- Get creative and make the program better. Perhaps there are other shorthand translations that could be added. Students could also modify the program or create a second program to incorporate the shorthand used in instant messaging.