Object-Oriented

Internally, the library class `java.lang.String` stores an array of characters. It also provides a variety of useful methods for comparing, manipulating, and searching text in general.

Manager:	Recorder:
Presenter:	Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Describe how characters and strings are represented in memory.
- Explain the difference of calling static versus non-static methods.
- Recognize common mistakes when working with the String API.

Process Skill Goals

During the activity, students should make progress toward:

- Working with all team members to reach consensus on hard questions. (Teamwork)

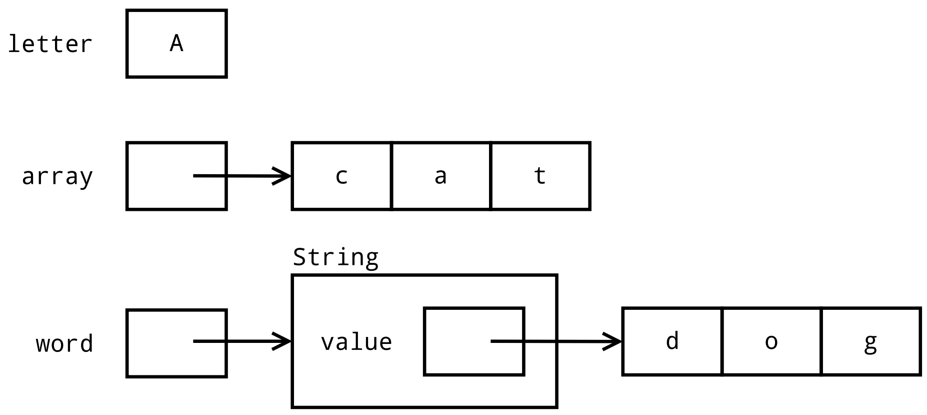
Model 1 Character Arrays

The primitive type `char` is used to store a single character, which can be a letter, a number, or a symbol. In contrast, the reference type `String` ***encapsulates*** an array of characters.

```
char letter;
letter = 'A';

char[] array;
array = new char[]
        {'c', 'a', 't'};

String word;
word = "dog";
```



Questions (15 min) **Start time:**

1. How is the syntax of character literals (like `'A'`) and string literals (like `"dog"`) different?

2. What is the index of `'d'` in the string above? What is the index of `'g'`?

3. What is the *value* of a `char` variable (i.e., stored in the variable's memory)? What is the *value* of an array variable? What is the *value* of a `String` variable?

4. Based on the diagram, what does it mean for an object to encapsulate data? How do you access the `char[]` inside of a `String` object?

5. Draw a memory diagram for the given code. (List the name of each variable next to a box containing its value.)

```
String str;
str = "Hi!";

char let;
let = 'X';

int num;
num = -1;

double foo;
foo = num;

String hmm;
hmm = str;
```

6. Recall that the == operator compares the *value* of two variables. What does it mean for two `char` variables to be ==? What does it mean for two `String` variables to be ==?

7. How could you determine whether two character arrays have the same contents? In other words, how does the `String.equals` method work internally?

Model 2 Non-Static Methods

Method	Returns	Description
`charAt(int)`	`char`	Returns the char value at the specified index of `this` string.
`indexOf(String)`	`int`	Returns the index within `this` string of the first occurrence of the specified substring.
`length()`	`int`	Returns the length of `this` string.
`substring(int, int)`	`String`	Returns a new string that is a substring of `this` string (from `beginIndex` to `endIndex - 1`).
`toUpperCase()`	`String`	Returns a copy of `this` string with all the characters converted to upper case.

Each method listed above is non-`static`. That is, they have an ***implicit parameter*** named `this` that is passed automatically. (Note: There are many other `String` methods not listed above.)

Questions (15 min) Start time:

8. If the variable `str` references the string `"hello world"`, then what is the return value of the following method calls?

a) `str.charAt(8)`

b) `str.indexOf("wo")`

c) `str.length()`

d) `str.substring(4, 7)`

e) `str.toUpperCase()`

9. Explain what precedes the dot operator (`.`) in the expressions above. What does it have to do with the keyword `this` in Model 2?

10. How many arguments does each method call in #8 have? (Hint: None of them have zero; don't forget to count the implicit argument.)

a)

b)

c)

d)

e)

To call a `static` method, you write *ClassName.methodName*, for example: `Math.abs(-5)`

To call a non-`static` method, you write *objectName.methodName*, for example: `str.charAt(8)`

Methods can be designed either way. Most `String` methods are non-`static`, because that makes the code easier to read.

Static (`str` passed explicitly)	**Non-Static (`str` passed implicitly)**
`String.charAt(str, 8) // wrong`	`str.charAt(8)`

11. Label each method below as `static` or non-`static` (based on how it is called).

a) `color.indexOf("RED")`

b) `String.format("%3d", x)`

c) `title.substring(0, 10)`

d) `String.valueOf(3.14)`

e) `name.charAt(0)`

12. Consider the following statement and compiler error. Why is it impossible to call `charAt` this way? How would you correct the error?

```
char c = String.charAt(0);
```

Error: non-static method charAt(int) cannot be referenced from a static context

13. For each method in #11, what object is referenced by `this`? (Write N/A if `this` is not passed to the method.)

a)

b)

c)

d)

e)

Model 3 Common Mistakes

Use *JShell* to run each of the code snippets. For Program B, do not provide any user input (just press Enter). Record the output of the last statement in the space below the table.

Program A	**Program B**

`String greeting = "hello world";`
`greeting.toUpperCase();`
`System.out.println(greeting);`
	`Scanner in = new Scanner(System.in);`
	`String line = in.nextLine();`
	`char letter = line.charAt(1);`

Questions (15 min) Start time:

14. What is the logic error you see when you run Program A?

15. In Program A, what is returned by the string method? What happens to the return value?

16. Describe two different ways you can fix the logic error in #15.

17. In what cases will Program B throw an exception? What is the error message displayed?

18. Describe two different ways you can fix the run-time error in #17.

19. To compare strings, you must use either the `String.equals` or `String.compareTo` method. Predict the output of the following code.

```java
String name1 = new String("Mark");
String name2 = new String("Mark");

// compare name1 and name2
if (name1 == name2) {
    System.out.println("name1 and name2 are identical");
} else {
    System.out.println("name1 and name2 are NOT identical");
}


// compare "Mark" and "Mark"
if (name1.equals(name2)) {
    System.out.println("name1 and name2 are equal");
} else {
    System.out.println("name1 and name2 are NOT equal");
}
```

20. What is the difference between ***identical*** and ***equal*** in the previous question?