**Part I. Representation of a String**

The *primitive type* **char** is used to store a *single* *character*, whether it is a letter, number or special character (for example: space, tab, or new line). More often, we want to store a *string of characters* in a single variable. We can do this by using a variable of the class type **String**. A String variable can be initialized to a ***String literal*** (which is similar to how a char variable can be initialized to a ***character literal***):

|  |
| --- |
| char letter;  letter = 'a';  String word;  word = **"**food**"**; |

We can visualize the multiple characters in a String as a list of characters, where each character can be indexed by its position in the String:

|  |  |  |  |
| --- | --- | --- | --- |
| f | o | o | d |
| 0 | 1 | 2 | 3 |

index numbers

1. What is the index for the character ‘d’ in the String “food”?

3

1. How many characters long is the String “food”?

4

1. Sketch a list (with indices numbering each character) for the String “hello world”.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| h | e | l | l | o |  | w | o | r | l | D |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

1. If the length of the String “hello world” is 11, what is the index number of ‘d’, the last letter in the String?

10

1. What is the index for the space in “hello world”? 5

**Part II. String Methods**

Once you have a String object, you can apply ***methods*** to the String.

Here are some commonly used String methods. More methods can be found in your textbook and in the Java API documentation.

|  |  |  |
| --- | --- | --- |
| **Method (Parameters)** | **Return Value Type** | **Description**  (“this String” refers to the original String - used in front of the . and the method name) |
| charAt(int) | char | Returns the character at the specified index in this String. |
| indexOf(String) | int | Returns the index of the first occurrence of the substring in this String. Returns -1 if the substring is not found. |
| length() | int | Returns the length of this String. |
| substring(int, int) | String | Returns a new String that is a substring of this String. |
| toUpperCase() | String | Returns a new String where all of the characters in this String have been converted to upper case. |

Assume you have “hello world” stored in String object called greeting. Here are two examples of ***method calls*** that you should type into Dr. Java’s Interactions Pane (one line at a time):

String greeting = **"**hello world**"**;

System.out.println(greeting.length());

System.out.println(greeting.charAt(0));

1. Most String methods have a return value. Examine the above table and the output in Dr. Java, and then give both the return value and its type for:
   1. the length method?

11, int

* 1. the charAt method?

h, string

1. Give the line of code that would calculate the length of String word (the variable from Model I).

System.out.println(“word”.length())

1. In your own words, explain what precedes the **.** (period) in a method call. Your answer should be more than a one-word answer.

Object of the class

1. The information the method requires inside the ()’s is known as the **parameters** for the method.
   1. What parameters (number of parameters and their types) does the length method require?

none

* 1. What parameters (number of parameters and their types) does the charAt method require?

Location inside the string, one integer

* 1. What parameters (number of parameters and their types) does the subtring method require?

Starting and ending location, two integers

* 1. Where does the above table provide information about the parameters?

In the description

1. Now examine the indexOf() method. We are going to call it on the String greeting.
   1. What parameters (number of parameters and their types) does the indexOf method require?

One string, location on the string

* 1. Give a line of code that uses indexOf to find the index where “world” appears in “hello world”. Be sure to test your code out in Dr. Java’s Interactions Pane.

System.out.println(ex.indexOf(“World”))

* 1. How would you represent a space as the parameter for the indexOf method?

System.out.println(ex.indexOf(“ ”))

* 1. What is the return type of the indexOf method? Write a line of Java code that declares a new variable of this type called position.

Integer

Int position = ex.indexOf(“ “);

If(position >= 0)

System.out.println(“there is a space at position: “ + position);

Else System.out.println(“there is no space”);

* 1. Give a line of code that uses indexOf to find the index for the space in “hello world” and saves the return value into position.

Int position = ex.indexOf(“ “)

1. Based on your answers to question 12, what are the three parts of the method call that you need to be thinking about when you call a method on a String object? Define them in your own words.
   1. first part:

return type

* 1. second part:

Parameter

* 1. third part:

main method

1. Using another method from the table, write a line of code that would isolate **"**hello**"** from the String greeting, using the value stored in position from Question 12.

greeting.indexOf(“ “);

substring(0,5);

**Part III. Common String Errors**

The following two example programs demonstrate common errors that often occur when programming with Strings.

|  |
| --- |
| public class StringErrors1 {  public static void main(String [] args) {  String greeting = **"**hello world**"**;  greeting.toUpperCase();  System.out.println(greeting);  }  } |

1. Type above code in Dr. Java or Eclipse. Compile and run the program. (The program should run.)
   1. What is the *logic error* you see when you run the program? In other words, what unexpected behavior do you see when you run the code? (You do not need to fix the error yet.) it doesn’t covert the string to uppercase

* 1. What String method is being called in the above program?

toUpperCase();

1. Look up the String method in the table from the previous model.
   1. What is the return type of the String method called in StringErrors1?

The return type is a string

* 1. What is the return value of the String method?

All characters are converted to uppercase

* 1. What happens to the original String, ***the calling object***?

Nothing happens to the original string it stays at it is

1. Usually a Java method will either return a value, or it will modify the calling object, not both. Review the String methods and their descriptions in the table in the previous Model.
   1. Which of the listed methods *return*a value based on the calling object?

indexOf();

charAt(int);

length();

substring(int, int)

toUpperCase();

* 1. Which of the listed methods modify the calling object?

None of them

1. There are at least two different ways of fixing the error in StringErrors1. Give at least one fix.

//public class StringErrors1 {

// public static void main(String [] args) {

String greeting = "hello world";

greeting = greeting.toUpperCase();

//comment out greeting if you want to temporarily uppercase

//System.out.println(greeting.toUpperCase());

System.out.println(greeting);

|  |
| --- |
| import java.util.Scanner;  public class StringErrors2 {  public static void main(String [] args) {  Scanner keyboard = new Scanner(System.in);  String input = keyboard.nextLine();  char letter = input.charAt(1);  System.out.println(letter);  }  *}* |

1. Type above code in Dr. Java or Eclipse. Compile and run the program a few times.
   1. What does this program do?

Whatever string you type, it prints whatever is at index 1

* 1. How might you make the program more user-friendly?

Create a prompt for the user

1. Now run the program two more times. The first time, type “a b c” and then the <enter> key when you run the program. The second time, just type the <enter> key.
   1. What *runtime* error message do you see?

String index out of range: 1

* 1. What line of code caused the error message?

Char letter = input.charAt(1);

* 1. How did you know which line of code caused the error message?

Because if there is no string, this line causes the problem because its looking for a character in a string.