| Investigate the **String** documentation. This can be found by typing in “Java API String” into Google and clicking on one of the first couple links that refers to Oracle or by using this link: <https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>Note that there is a Field Summary, a Constructor Summary, and a Method Summary. I don’t expect you to understand everything, but note that you can see which methods you have access to when using the String class. |  |
| --- | --- |
| **Exercise 5.3** Look up the **startsWith** method in the documentation for **String**. If you have a difficult time finding something, you can always type “control-F” (“Command-F” for mac) and search quickly for the word **startsWith**. There are two versions. Describe in your own words what they do and the differences between them. |  |
| **Exercise 5.4** Is there a method in the **String** class that tests whether a string ends with a given suffix? If so, what is it called and what are its parameters and return type?Name of method:Parameter(s): Return type: |  |
| **Exercise 5.5** Is there a method in the **String** class that returns the number of characters in the string? If so, what is it called and what are its parameters?Name of method:Parameter(s): |  |
| **Exercise 5.7** Find the **trim** method in the **String** class’s documentation. Write down the signature of that method. Write down an example external method call to that method on a **String** variable called **text**. |  |
| **Open and download the “tech-support1” code. Try running the code to see how it works. Right now, it is not a very “smart” tech support system, but we will make it better!****Exercise 5.8** Use trim to get rid of the extra trailing spaces in the input your version of the *tech-support1* project. You will do this in the SupportSystem class in the start() method, this is where you will be working for the next few exercises. Test it to confirm that it is tolerant of extra space around the word ‘bye’. |  |
| **Exercise 5.9** Improve the code of the **SupportSystem** class in the *tech-support1* project so that case in the input is ignored. Use the **String** class’s **toLowerCase** method to do this. Remember that this method will not actually change the **String** it is called on, but result in the creation of a new one being created with slightly different contents. |  |
| **Exercise 5.10** Find the **equals** method in the documentation for class **String**. What is the return type of this method? |  |
| **Exercise 5.11** Change your implementation to use the **equals** method instead of **startsWith**. |  |
| Find the class **Random** in the Java class library documentation online. Skim through it quickly to see the methods.Note that you will probably not understand everything that is stated in the documentation. Just try to find out what you need to know. |  |
| **Exercise 5.13** Write a small code fragment (on paper) that generates a random integer number using this class. |  |
| **Exercise 5.14** Create a new class called RandomTester. Write a method called **printMultiRandom(int howMany)**, which prints out multiple random numbers; so if the parameter was 10, it would print out 10 random numbers – don’t worry, the numbers can be large values.  |  |
| **Exercise 5.15** Find the **nextInt** method in class **Random** library documentation that allows the target range of random numbers to be specified. In other words, a method that allows you to specify an upper limit. What are the possible random numbers that are generated when you call this method with 100 as its parameter?Method Name:Possible values with a parameter of 100: |  |

| **Exercise 5.18** Write a method called **getResponse** so that it uses an **ArrayList** to store an arbitrary number of responses (such as “yes”, “no”, or “maybe”) and randomly returns one of them. |  |
| --- | --- |
| **Exercise 5.19** Add a method to your **RandomTester** class that takes a parameter **max** and generates a random number in the range 1 to **max** (inclusive). |  |