Dylan Perkins

3/18/2023

Week 3: Research

Five methods from the String JavaDocs:

1. `boolean endsWith(String suffix)`:

- (a) `public boolean endsWith(String suffix)`

- (b) This method returns a boolean indicating whether the string ends with the specified suffix.

- (c) This method could be useful for checking if a string ends with a specific character or sequence of characters. For example, one could use this method to check if a given file name ends with a proper file extension, like `.txt` or `.csv`.

2. `String substring(int beginIndex, int endIndex)`:

- (a) `public String substring(int beginIndex, int endIndex)`

- (b) This method returns a new string that is a substring of the original string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`.

- (c) This method could be useful for extracting a portion of a string that falls between two indices. For example, one could use this method to extract the date from a string that contains both a date and a time.

3. `int indexOf(int ch)`:

- (a) `public int indexOf(int ch)`

- (b) This method returns the index of the first occurrence of the specified character in the string, or -1 if the character does not occur.

- (c) This method could be useful for finding the position of a specific character in a string. For example, one could use this method to determine if a string contains a specific character, and if so, at what position in the string.

4. `String replace(char oldChar, char newChar)`:

- (a) `public String replace(char oldChar, char newChar)`

- (b) This method returns a new string resulting from replacing all occurrences of `oldChar` in the original string with `newChar`.

- (c) This method could be useful for replacing all occurrences of a specific character in a string with a different character. For example, one could use this method to replace all occurrences of a whitespace character with an underscore character.

5. `String[] split(String regex)`:

- (a) `public String[] split(String regex)`

- (b) This method returns an array of strings consisting of the substrings in the original string that are separated by the specified regular expression.

- (c) This method could be useful for breaking up a string into smaller chunks based on a certain pattern or delimiter. For example, one could use this method to split a string of words into an array of individual words by using a space character as the regular expression.

Five methods from the String JavaDocs:

1. `Arrays.copyOf(T[] original, int newLength)`

(a) `public static <T> T[] copyOf(T[] original, int newLength)`

(b) This method copies the specified array, truncating or padding with nulls (if necessary) so the copy has the specified length. For all indices that are valid in both the original array and the copy, the two arrays will contain identical values. For any indices that are valid in the copy but not the original, the copy will contain null.

(c) This method would be useful when you want to create a new array with a specific length, that either truncates or pads with nulls (if necessary), based on the original array. This can be useful when you need to modify an existing array but want to keep a copy of the original array as well.

2. `Arrays.binarySearch(T[] a, T key)`

(a) `public static <T> int binarySearch(T[] a, T key)`

(b) This method searches the specified array of objects (the second argument) for the specified value (the first argument) using the binary search algorithm. The array must be sorted into ascending order

according to the natural ordering of its elements (as by the sort(T[]) method) prior to making this call. If it is not sorted, the results are undefined.

(c) This method would be useful when you need to search for a specific value in an array that has already been sorted. Binary search algorithm is an efficient way to perform search in a sorted array for a specific value.

3. `Arrays.equals(T[] a, T[] a2)`

(a) `public static <T> boolean equals(T[] a, T[] a2)`

(b) This method returns true if the two specified arrays of objects are equal to one another. The two arrays are considered equal if both arrays contain the same number of elements, and all corresponding pairs of elements in the two arrays are equal. Two elements e1 and e2 are considered equal if `(e1==null ? e2==null : e1.equals(e2))`.

(c) This method would be useful when you need to compare two arrays to check if they have the same elements. Since Java does not have a built-in equals method for arrays, this method can be used to compare two arrays where each element of the array is compared using the equals method.

4. `Arrays.asList(T... a)`

(a) `public static <T> List<T> asList(T... a)`

(b) This method returns a fixed-size list backed by the specified array. The list returned is serializable and implements List interface. The List can be modified if the array is modified.

(c) This method would be useful when you need to convert an array into a List. This can be useful when you have an existing array but need to use List methods or when you need to pass an array into a method that requires a List as an argument.

5. `Arrays.toString(T[] a)`

(a) `public static <T> String toString(T[] a)`

(b) This method returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets (`"[]"`). Adjacent elements are separated by the characters ", " (a comma followed by a space). Elements are converted to strings as by `String.valueOf(Object)`.

(c) This method would be useful when you need to display the contents of an array in a human-readable format. By calling this method and displaying the resulting string, you can easily display the elements of the array without having to manually iterate over the array and build the string representation yourself. This method can be useful when debugging or when displaying information to the user.