Download

Day 7: StringBuilder + Arrays

Download

Agenda

- StringBuilder
- Stack v Heap
- Array
- Lab 4

Download

String Concatenation not always good

As we said on Monday, constantly concatenating to Strings isn't ideal. It ends up creating too many new objects in memory.

So how can we effectively add to a string?

Download

StringBuilder

Download

StringBuilder

Just like Scanner, this is another "class" we can use. However, unlike Scanner, StringBuilder doesn't have to be imported at the top.

This is because StringBuilder comes from `java.lang` system. This means that it's already automatically imported for us, just like how Math is automatically imported for us.

Download

What

StringBuilder allows us to construct a new String by adding values into a String buffer. Additionally, using a StringBuilder, we can also remove values from a String.

Download

Why

Well, it is more memory efficient as each addition doesn't create a new string in memory. Instead, it adds the value into a buffer, that continues to get updated until we call a function to give us the string.

How

We can create a new StringBuilder using the same format as our Scanner.

```
StringBuilder s = new StringBuilder();

s.append("hello");

s.append(" world");

String str = s.toString();
```

How

We can then call append on the variable containing the StringBuilder.

This will append a string into the current StringBuilder.

```
StringBuilder s = new
StringBuilder();

s.append("hello");

// s contains "hello"

s.append(" world");

String str = s.toString();
```

How

When we call append again, the StringBuilder will add the new value, but won't create a new String in memory. It'll continue adding until we stop.

```
StringBuilder s = new
StringBuilder();

s.append("hello");

// s contains "hello"

s.append(" world");

// s contains "hello world"

String str = s.toString();
```

How

In order to get the string out, we simply call `toString` on the variable containing the StringBuilder. This gives us the final resulting string!

```
StringBuilder s = new
StringBuilder();

s.append("hello");

// s contains "hello"

s.append(" world");

// s contains "hello world"

String str = s.toString();

// Gives us "hello world"
```

Removing values (deleteCharAt)

Let's say we want to remove a value from our StringBuilder.

We can call `deleteCharAt(index)` where index is the location of the value we want to remove.

```
StringBuilder s = new StringBuilder();

s.append("hello world!");

s.deleteCharAt(s.length() - 1);

s.toString(); // gives us "hello world"
```

Removing values (delete)

Alternatively, we can also call the `delete(startIndex, endIndex)` to delete a substring from our StringBuilder.

```
StringBuilder s = new
StringBuilder();

s.append("hello world!");

s.delete(5, 12); // remove "world"

s.toString(); // gives us "hello"
```

Getting the String

When we want to get the string from the StringBuilder, we can simply call the `toString()` function!

```
StringBuilder s = new
StringBuilder();

s.append("hello");

s.append(" world");

String str = s.toString(); // gives
us "hello world"
```

Download

StringBuffer?

If you look up StringBuilder, you may also find `StringBuffer`.

For all intensive purposes of this class, they are the same.

Download

Memory Access

Download

Memory?

You may have noticed that I've mentioned memory a lot in class...

I'm not just bringing it up to be annoying, it is an important topic to talk about when we think about our programs.

Download

The size of a data type matters to how we design our programs. Understanding that some types are bigger than others means we can create our programs to cater to these needs better.

Download

Stack and Heap

Download

Stack and Heap

Our memory is split into two main concepts called the **Stack** and **Heap**

Download

Differences

Stack

- Contains methods/function calls
- Local method primitive values/variables stored
- References to Objects used by methods/functions
- Deallocated/freed space when methods/functions are done
- Variables in Stack are freed once method is done

Heap

- Contains Objects
- References to location of Objects in Heap, sent to Stack
- Garbage Collected to free memory when objects are out of scope/no longer in use
- Objects created in Heap exists throughout the program globally until garbage collected

Download

Garbage Collection?

- Automatic process done by the JVM to clear memory from the Heap
- Common in some other languages such as Go, JavaScript, Python

Download

Array

Download

Array?

In Python, you should've learned about Lists. The data structure that uses [] and allows you to use indexes, add, and remove values from it.

In Java, we have a similar structure called an Array. This also uses [] and we can index it, but we cannot add to it nor remove values from it. We can only get and set values in it.

Download

Creating an array

We can create an array using the following notion.

Similar to any regular variable, we must give it a data type with square brackets after the type.

We will set it equal to a new array. The `<data type>` is the same in both cases. In this case, we must give it a size.

```
<data type>[] variable = new <data type>[size];
```

int array example

This will initialize an integer array with a size of 10 elements

```
int[] array = new int[10];
```

int array example (pre-set)

Alternatively, we can set the values in the array without setting the size.

The example to the right will initialize an array with the values of 1-6

```
int[] array = { 1, 2, 3, 4, 5, 6 };
```

Getting values

Alternatively, we can set the values in the array without setting the size.

The example to the right will initialize an array with the values of 1-6

```
int[] array = { 1, 2, 3, 4, 5, 6 };
```

Download

Getting values

We can get values using square brackets and indexes, just like we can in Python.

```
int[] arr = { 1, 4, 9, 16, 25 };

System.out.println(arr[0]); //
gives us 1

System.out.println(arr[4]); //
gives us 25
```

Setting values

Similar to how we can do it in Python, we can set values using the index.

```
int[] arr = { 1, 4, 9, 16, 25 };

arr[3] = 4;

System.out.println(arr[3]); //
gives us 4
```

Types?

One of the major differences between a Python list and a Java array is that a java array must contain a single type. Python lists can contain multiple types and values, but a Java array must keep the same type.

```
int arr[] = new int[10];

arr[0] = 1;

arr[1] = "3"; // not allowed!
```

Length

We can also get the length of an array using the length property.

This one doesn't use parentheses since it isn't a function, but rather a property of arrays.

```
int[] arr = { 1, 2, 3, 4, 5, 6 };

System.out.println(arr.length); //
Gives us 6
```

Iterating through an array

We can use a `for loop` to iterate through an array using the indexes of the array and the length of the array.

```
int[] arr = new int[11];

for(int i = 0; i < arr.length; i++)
{

  arr[i] = i * i;

}

// Stores the squares into the
array

// 0 1 4 9 16 25 36 49 64 81 100
```

For each/enhanced for loop

We have another way of looping through a collection using a `for loop`. It looks a little different, but it's called a `for-each` or a `enhanced for loop`.

Different notation and uses a colon.

The left is the type of a single value in the collection. The right of the colon is the collection itself.

```
int[] arr = { 1, 4, 9, 16, 25, 36
};

for(int num : arr) {

  System.out.print(num + " ");

}

// Gives 1 4 9 16 25 36
```

For each/enhanced for loop

This one actually uses colons. The regular for loop uses semi-colons.

be careful/mindful about the difference!

```
int[] arr = { 1, 4, 9, 16, 25, 36
};

for(int num : arr) {

  System.out.print(num + " ");

}

// Gives 1 4 9 16 25 36
```

When?

Regular for loop:

- Index is necessary
- Changing array with index

Enhanced for loop

- Index not necessary, just need the value
- Only need to read values from a collection

Download

Lab 4