Using Classes

String Class

Lecture Contents

- Declaring a `String`
- **`String`** Class Methods
  - **`length()`**
  - **`charAt(int index)`** ← **<u>not</u> part of the AP Subset!**
  - **`indexOf(String s)`**
  - **`substring(int start)`**
  - **`substring(int start, int end)`**
  - `equals(String s)`
  - `compareTo(String s)`

} These methods in part 2

`String` Class – Calling Methods

- Common `String` class methods:

```
int length()
int indexOf(String str)
String substring(int start)
String substring(int start, int end)
boolean equals(String other)
int compareTo(String other)

String toUpperCase()
String toLowerCase()
char charAt(int index)
boolean equalsIgnoreCase(String other)
```

`String` Class – Calling Methods

- We can call methods from the String class:

```
String s1 = "Hello World!";
String s2 = s1.toUpperCase();

System.out.println(s2);
```

`String` Class – Calling Methods

- We can call methods from the String class:

```
String s1 = "Hello World!";
String s2 = s1.toUpperCase();

System.out.println(s2);
```

  - Notice the difference between using methods of `System` and `Math` and methods of `String` …?

```
HELLO WORLD!
```

`String` Class – Calling Methods

- We can call methods from the String class:

```
String s1 = "Hello World!";
String s2 = s1.toUpperCase();

System.out.println(s2);
```

  - Notice the difference between using methods of `System` and `Math` and methods of `String` …?
    - Math.abs(-5) ← called using the ***class*** name
    - s1.toUpperCase() ← called using the ***variable*** name

```
HELLO WORLD!
```

`String` Class – Calling Methods

- We can call methods from the String class:

```
public static void tryToUpperCase2() {
   String s1 = "Hello World!";
   String s2 = s1.toUpperCase();
   String s3 = "Goodbye Cruel World...";
   String s4 = s3.toLowerCase();

   System.out.println(s2);
   System.out.println(s4);
}
```

```
HELLO WORLD!
goodbye cruel world...
```

`String` Class – `length()`

- We can call methods from the `String` class:

```
public static void tryStringLength() {
   String s1 = "Hello World!";
   int i = s1.length();

   System.out.println(i);
}
```

`String` Class – `length()`

- We can call methods from the `String` class:

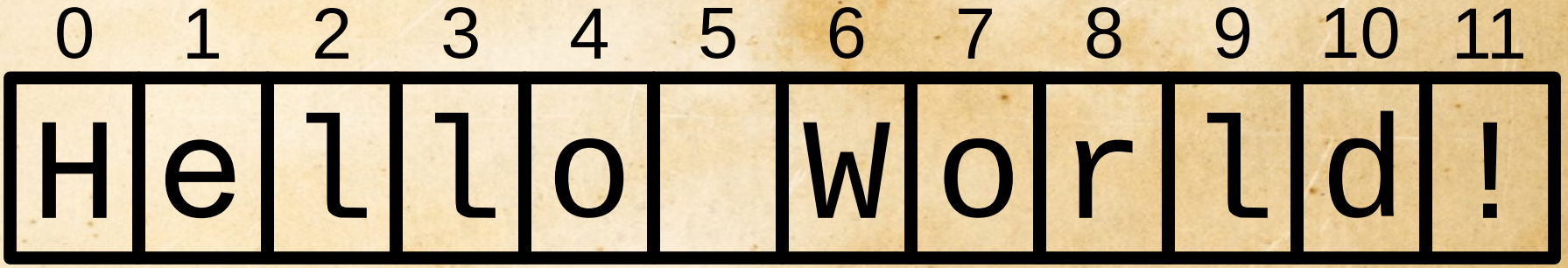
```
public static void tryStringLength() {
   String s1 = "Hello World!";
   int i = s1.length();

   System.out.println(i);
}
```

```
12
```

`String` Class – indices

- Each letter of a string has a number assigned called the ***index***.
- The numbering is sequential, starting from zero.



What is the length of the String above?

```
System.out.println("Hello World!".length());
```

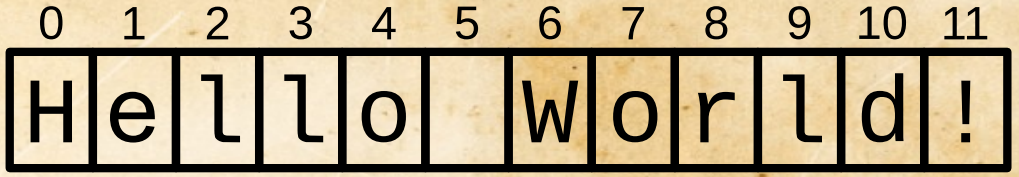
`String` Class – `charAt()`

- Each letter of a string has a number assigned called the ***index***.
- The numbering is sequential, starting from zero.


```
public static void playWithCharAt() {
   String s = "Hello World!";
   char c = s.charAt(8);
   System.out.println(c);
}
```

`String` Class – `charAt()`

- Each letter of a string has a number assigned called the ***index***.
- The numbering is sequential, starting from zero.

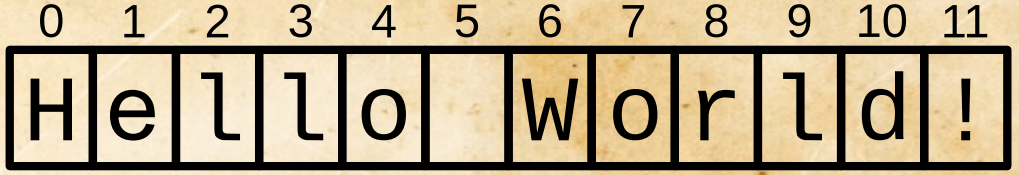



```
public static void playWithCharAt() {
   String s = "Hello World!";
   char c = s.charAt(8);
   System.out.println(c);
}
```

```
r
```

`String` Class – `charAt()`

- Each letter of a string has a number assigned called the ***index***.
- The numbering is sequential, starting from zero.




```
public static void playWithCharAt() {
    String s = "Hello World!";
    char c = s.charAt(8);
    System.out.println(c);
}
```

However, the primitive type **char** is not included in the AP subset!

```
W
```

`String` Class – Calling Methods

- Common `String` class methods:

```
int length()
int indexOf(String str)
String substring(int start)
String substring(int start, int end)
boolean equals(String other)
int compareTo(String other)

String toUpperCase()
String toLowerCase()
char charAt(int index)
boolean equalsIgnoreCase(String other)
```

`String` Class – `indexOf()`

- To find the location of a substring within a string, we use:

```
<String>.substring(String sub)
```

- It will find the first occurrence of the substring:

```
public static void playWithIndexOf1() {
   String s = "Hello World!";
   int i = s.indexOf("World");
   System.out.println(i);
}
```

```
6
```

`String` Class – `indexOf()`

- To find the location of a substring within a string, we use:

```
<String>.substring(String sub)
```

- It will find the first occurrence of the substring:

```
public static void playWithIndexOf() {
   String s = "We started painting in art class.";
   int i = s.indexOf("art");
   System.out.println(i);
}
```

`String` Class – `indexOf()`

- To find the location of a substring within a string, we use:

```
<String>.substring(String sub)
```

- It will find the first occurrence of the substring:

```
public static void playWithIndexOf() {
   String s = "We started painting in art class.";
   int i = s.indexOf("art");
   System.out.println(i);
}
```

```
5
```

`String` Class – `indexOf()`

- To find the location of a substring within a string, we use:

```
<String>.substring(String sub)
<String>.substring(String sub, int start)
```

- It will find the first occurrence of the substring:

```
public static void playWithIndexOf2() {
   String s = "We started painting in art class.";
   int i = s.indexOf("art");
   System.out.println(i);
   i = s.indexOf("art",i+1);
   System.out.println(i);
}
```

Note: to find subsequent occurrences, we can give a starting index… but this is **not** part of the Java AP Subset!

```
5
23
```

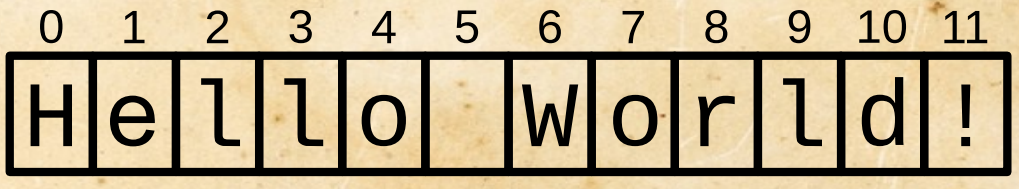
`String` Class – Calling Methods

- Common `String` class methods:

```
int length()
int indexOf(String str)
String substring(int start)
String substring(int start, int end)
boolean equals(String other)
int compareTo(String other)

String toUpperCase()
String toLowerCase()
char charAt(int index)
boolean equalsIgnoreCase(String other)
```

`String` Class – `substring()`

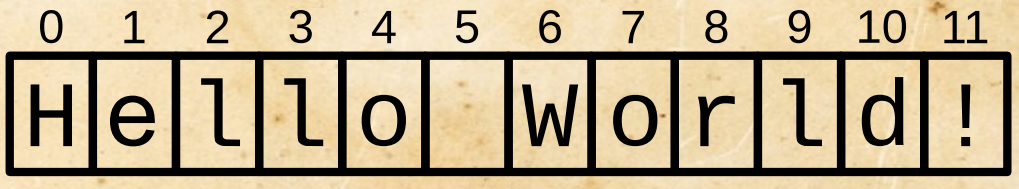


- Since we don't cover **`char`** in the AP Subset…
- The substring method returns a `String` within the range given by a:
  - Start index (then end is set to the end of the string)

    `<String>.substring(int start)`
  - A start and an end index (it includes the start, excludes the end character)

    `<String>.substring(int start, int end)`

`String` Class – `substring()`



- Since we don't cover **`char`** in the AP Subset…
- The substring method returns a String within the range given by a:
  - Start index (then end is set to the end of the string)

    ```
    <String>.substring(int start)
    ```

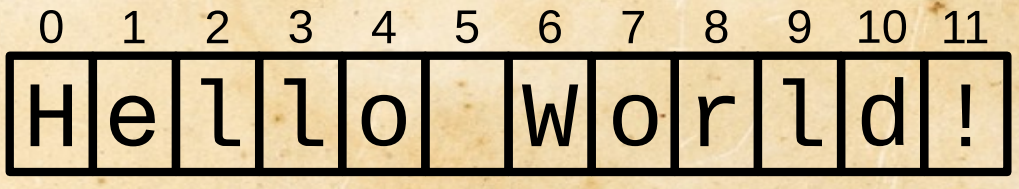
  - A start and an end index (it includes the start, excludes the end character)

    ```
    <String>.substring(int start, int end)
    ```

  - These are equivalent:

    ```
    s.substring(6, s.length())
    s.substring(6)
    ```

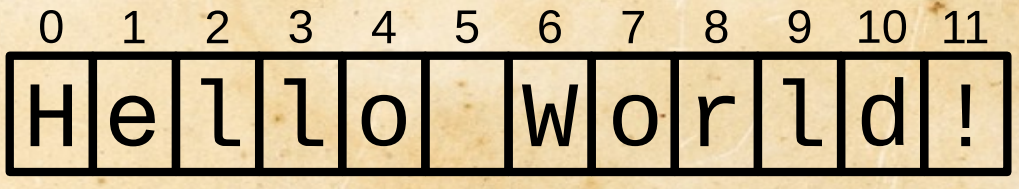
`String` Class – `substring()`



- Since we don't cover **`char`** in the AP Subset…
- The substring method returns a String within the range given by a start and end index:

```
public static void playWithSubstring() {
    String s = "Hello World!";
    String s1 = s.substring(6,7);
    System.out.println(s1);
}
```

`String` Class – `substring()`

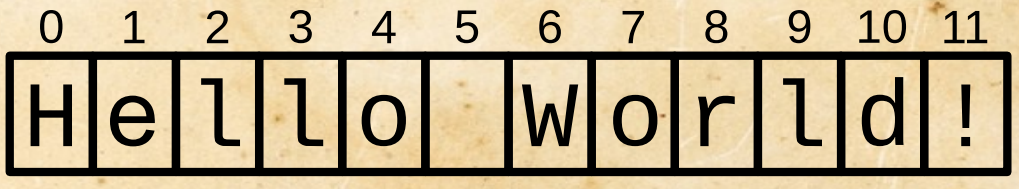


- Since we don't cover **`char`** in the AP Subset…
- The substring method returns a String within the range given by a start and end index:

```
public static void playWithSubstring() {
   String s = "Hello World!";
   String s1 = s.substring(6,7);
   System.out.println(s1);
}
```

```
W
```

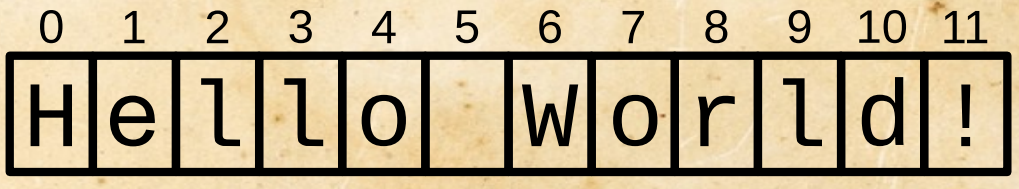
`String` Class – `substring()`



- What is the output of this code?

```
public static void playWithSubstring() {
   String s = "Hello World!";
   String s1 = s.substring(6);
   System.out.println(s1);
   String s2 = s.substring(3,8);
   System.out.println(s2);
}
```

String Class – substring()



- What is the output of this code?

```
public static void playWithSubstring() {
   String s = "Hello World!";
   String s1 = s.substring(6);
   System.out.println(s1);
   String s2 = s.substring(3,8);
   System.out.println(s2);
}
```

```
World!
Lo Wo
```

Using Classes

String Class