2.5 `Strings`

- The `String` Type:
  - Type Variable   Literal
  - `String name = “Harry”`
- Once you have a `String` variable, you can use methods such as:

```
  int n = name.length();  // n will be
   assigned 5
```

- A `String`’s length is the number of characters inside:
  - An empty `String` (length 0) is shown as “”
  - The maximum length is quite large (an `int`)

`String` Concatenation (+)

- You can ‘add’ one `String` onto the end of another

```
  String fName = “Harry”
  String lName = “Morgan“
  String name = fname + lname;  // HarryMorgan
```

- You wanted a space in between?

```
  String name = fname + “ “ + lname;  // Harry Morgan
```

- To concatenate a numeric variable to a `String`:

```
 String a = “Agent“;
 int n = 7;
 String bond = a + n;    // Agent7
```

- Concatenate `String`s and numerics inside `println`:

```
 System.out.println("The total is " + total);
```

`String` Input

- You can read a `String` from the console with:

```
System.out.print("Please enter your name: ");
String name = in.next();
```

  - The `next` method reads one word at a time
  - It looks for ‘white space’ delimiters

- You can read an entire line from the console with:

```
System.out.print("Please enter your address: ");
String address = in.nextLine();
```

  - The `nextLine` method reads until the user hits ‘Enter’

- Converting a `String` variable to a number

```
System.out.print("Please enter your age: ");
String input = in.nextLine();
int age = Integer.parseInt(input);  // only digits!
```

`String` Escape Sequences

- How would you print a double quote?
  - Preface the `"` with a `\` inside the double quoted `String`

```
System.out.print("He said \"Hello\"");
```

- OK, then how do you print a backslash?
  - Preface the `\` with another `\`!

```
System.out.print("“C:\\Temp\\Secret.txt“);
```

- Special characters inside `String`s
  - Output a newline with a ‘`\n`’

```
System.out.print("*\n**\n***\n");
```

```
*
**
***
```

`String`s and Characters

- `String`s are sequences of characters
  - Unicode characters to be exact
  - Characters have their own type: `char`
  - Characters have numeric values
    - See the ASCII code chart in Appendix B
    - For example, the letter ‘H’ has a value of 72 if it were a number

- Use single quotes around a `char`

```
char initial = ‘B’;
```

- Use double quotes around a `String`

```
String initials = “BRL”;
```

Copying a `char` from a `String`

- Each `char` inside a `String` has an index number:

0	1	2	3	4	5	6	7	8	9
c	h	a	r	s		h	e	r	e

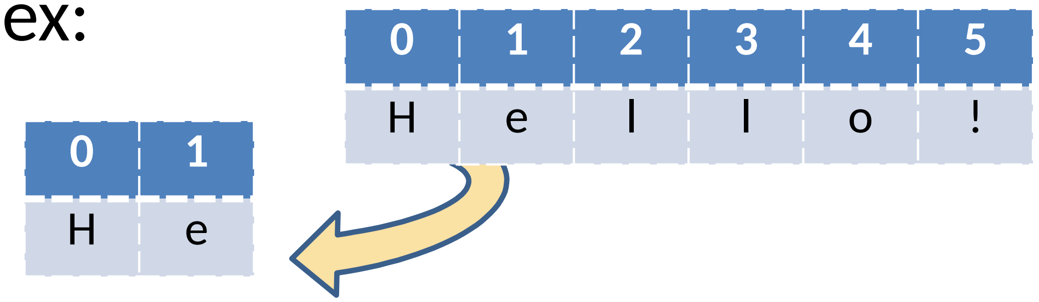
- The first `char` is index zero `(0)`
- The `charAt` method returns a `char` at a given index inside a `String`:

0	1	2	3	4
H	a	r	r	y

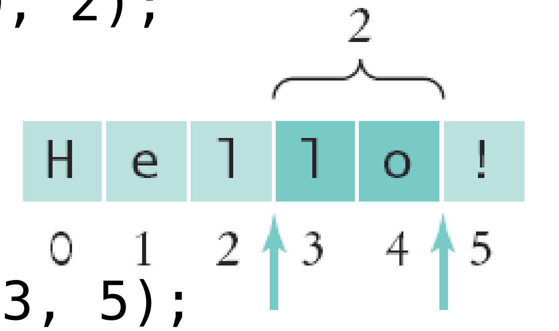
```
String greeting = "Harry";
char start = greeting.charAt(0);
char last = greeting.charAt(4);
```

Copying portion of a `String`

- A substring is a portion of a `String`
- The `substring` method returns a portion of a `String` at a given index for a number of `char`s, starting at an index:



```
String greeting = "Hello!";
String sub = greeting.substring(0, 2);
```



```
String sub2 = greeting.substring(3, 5);
```

Table 9: String Operations (1)

Table 9 String Operations

Statement	Result	Comment

`string str = "Ja";`
`str = str + "va";`
	`str` is set to `"Java"`	When applied to strings,
		`+` denotes concatenation.

`System.out.println("Please"`
`   + " enter your name: ");`
	Prints
	`Please enter your name:`
		Use concatenation to break up strings that don't fit into one line.

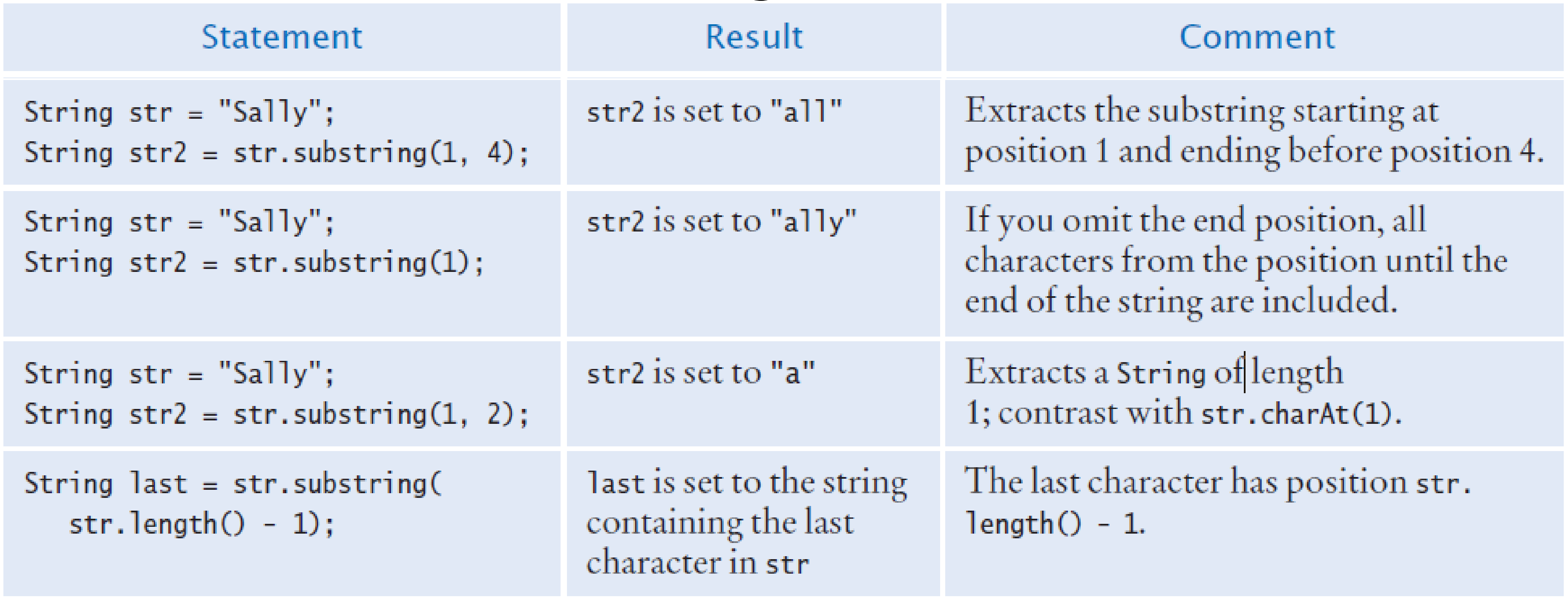
`team = 49 + "ers"`	`team` is set to `"49ers"`	Because `"ers"` is a string, 49 is converted to a string.

`String first = in.next();`
`String last = in.next();`
`(User input: Harry Morgan)`
	`first` contains `"Harry"`
	`last` contains `"Morgan"`
		The `next` method places the next word into the string variable.

`String greeting = "H & S";`
`int n = greeting.length();`
	`n` is set to 5	Each space counts as one character.

`String str = "Sally";`
`char ch = str.charAt(1);`
	`ch` is set to `'a'`	This is a `char` value, not a `String`. Note that the initial position is `0`.

Table 9: String Operations (2)



Statement	Result	Comment

`String str = "Sally";`
`String str2 = str.substring(1, 4);`
	`str2` is set to `"all"`	Extracts the substring starting at position 1 and ending before position 4.

`String str = "Sally";`
`String str2 = str.substring(1);`
	`str2` is set to `"ally"`	If you omit the end position, all characters from the position until the end of the string are included.

`String str = "Sally";`
`String str2 = str.substring(1, 2);`
	`str2` is set to `"a"`	Extracts a `String` of length 1; contrast with `str.charAt(1)`.

`String last = str.substring(`
`   str.length() - 1);`
	`last` is set to the string containing the last character in `str`	The last character has position `str.length() - 1`.