CSCI 136
Data Structures & Advanced Programming

Spring 2021

Instructors

Sam McCauley & Bill Lenhart

Java III : The String & Scanner Classes

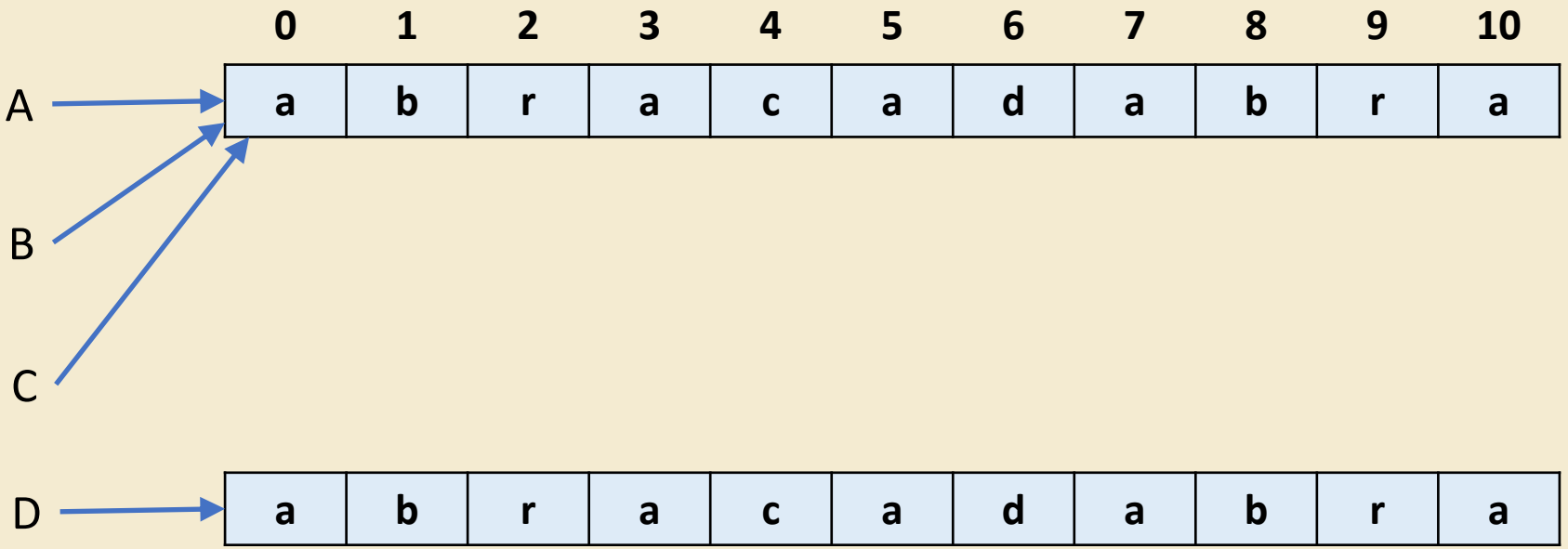
(but mostly Strings)

The String Class

- String is not a primitive type in Java, it is a *class type*
- However, Java provides language level support for Strings
  - String literals: `"Bob was here!", "-11.3", "A", ""`
- A single character can be accessed using `charAt()`
  - As with arrays, indexing starts at position 0
  - `String s = "computer";`
  - `char c = s.charAt(5);  // c gets value 't'`
  - `c = "oops".charAt(4);  // run-time error!`
- String provides a length method
  - `int len = s.length();  // len gets value 8`
  - `len = "".length();  // len gets value 0`
- Uninitialized `String` variables have the special value `null`

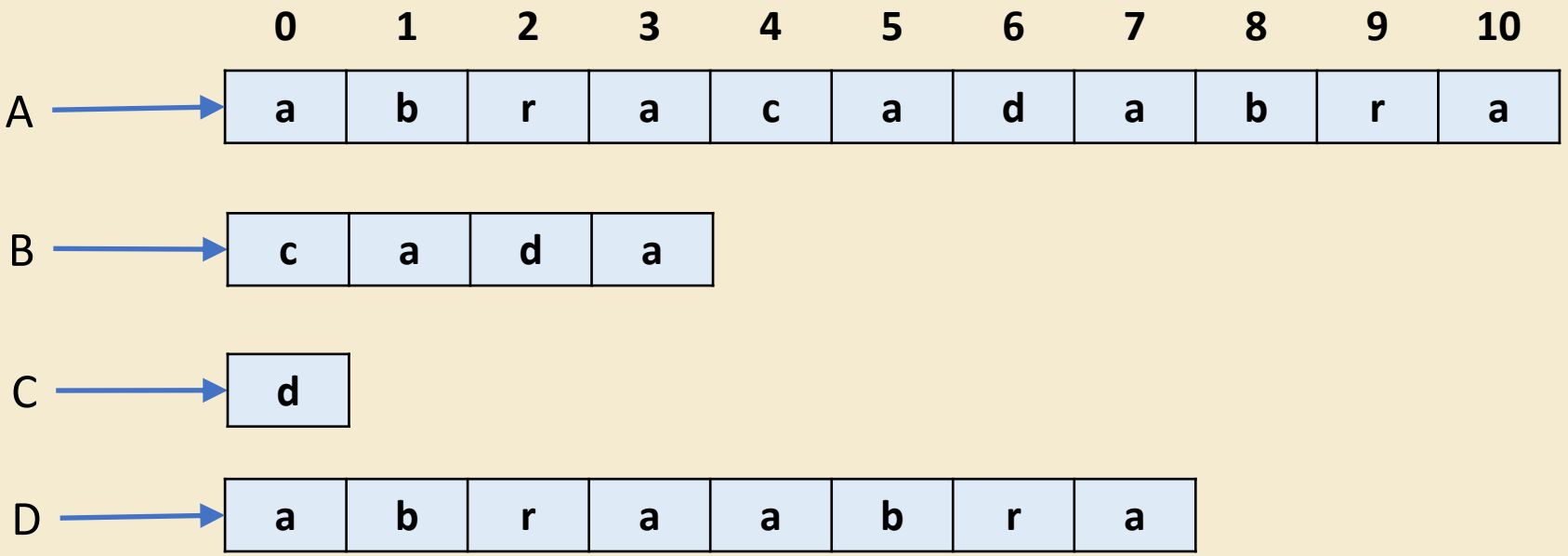
String Subtilties

```
String A = "abracadabra";
String B = A;
String C = "abracadabra";
String D = new String("abracadabra");
```



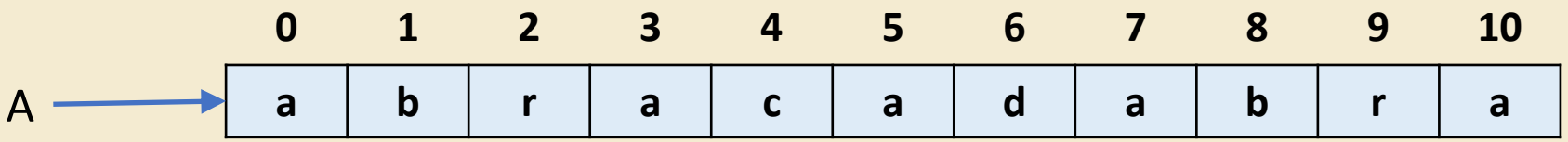
Substring Method

```
String A = "abracadabra";
String B = A.substring(4,8);
String C = A.substring(6,7);
String D = A.substring(0,4) + A.substring(7);
```



IndexOf Method

```
String A = "abracadabra";
int loc = A.indexOf("ra");
// loc = 2
loc = A.indexOf("ra",5);
// loc = 9
loc = A.indexOf("ra", A.indexOf("ra")+1);
// loc = 9
```



String methods in Java

- Useful methods (also check String javadoc page)
  - `indexOf(string) : int`
  - `indexOf(string, startIndex) : int`
  - `substring(fromPos, toPos) : String`
  - `substring(fromPos) : String`
  - `charAt(int index) : char`
  - `equals(other) : bool` ← *Always use this!*
  - `toLowerCase() : String`
  - `toUpperCase() : String`
  - `compareTo(string) : int`
  - `length() : int`
  - `startsWith(string) : bool`
- Understand special cases!

Example: Delete substring

Strings are immutable

- No portion of a String can be altered
- To modify a String, copy portions of it

```
// code to remove first occurrence of String sub from String s

  String result = s;          // if s doesn't contain sub, result is s
  int upTo = s.indexOf(sub);     // End of left part of s
  if( upTo > -1) {            // s contains sub
    int thenFrom = upTo + sub.length();     // Start of right part
    result = s.substring(0,upTo) + s.substring(thenFrom);
  }
```

Using Strings

- Application: Parsing an XML file of a CD collection
  - XML = eXtensible Markup Language
  - XML is used for many things
  - Music track info:

```
<TRACK>
   <NAME>Big Willie style</NAME>
   <ARTIST>Will Smith</ARTIST>
   <ALBUM>Big Willie style</ALBUM>
   <GENRE>Pop Rap</GENRE>
   <YEAR>1997</YEAR>
</TRACK>
```

- How can we find and print just the track names?
  - See TrackTitles.java
  - java TrackTitles < trackList.xml

Java's Scanner Class

- Use of `Scanner` class for input from Terminal
  - `System` class provides an object called `in` that allows low-level input
    - in is of type `InputStream`
  - `Scanner` class provides higher-level input reading from an `InputStream`
    - such as a Terminal window or a file
    - Intuition: `Scanner` provides methods to "consume" the data in an `InputStream`
  - `Scanner` method include
    - `hasNext()` → `boolean` : Is there more input remaining?
    - `nextLine()` → `String` : Consumes and returns the unread contents of current line
    - `next()` → `String` : Consumes and returns next "token" (String surrounded by white space)
    - `nextInt()` → `int` : Consumes and returns (as an int) next token, if token represents an int value
      - Throws an exception otherwise
      - Also `nextDouble()`, `nextFloat()`, `nextChar()`, …
  - The `Scanner` class must be imported (few classes, like `System`, don't need to be)
    - `import java.util.Scanner`