

## Some more about Classes and Objects

- String class
- File input/output processing
- 2D Arrays
- More on recursion



CSC 176 String, File I/O, 2D arrays, recursion

Slide 1

## String class

(Section 4.4 in textbook)

- Part of Java API
  - Each String value is an object
  - String objects are immutable
  - Conceptually similar to 1D array of characters
- Comparing String objects
  - `int compareTo(String otherStr)`
  - `int compareToIgnoreCase(String otherStr)`
  - `boolean equals(Object anObject)`
  - `boolean equalsIgnoreCase(String otherStr)`
  - Important
    - The relational operators (e.g., `<` `>` `==` `!=` ) do not work on String objects!



CSC 176 String, File I/O, 2D arrays, recursion

Slide 2

# String class

(Section 4.4 in textbook, cont'd)

- **Some commonly used String methods**

- `char charAt(int index)`
- `String concat(String str)`
  - Can also use the `+` symbol for String concatenation
- `boolean isEmpty()`
- `int length()`
- `String toLowerCase()`
- `String trim()`
- `String toUpperCase()`
- `static String valueOf(boolean b)`
- `static String valueOf(char d)`
- `static String valueOf(char[] data)`
- `static String valueOf(double d)`
- `static String valueOf(int i)`
- `static String valueOf(Object obj)`

CSC 176 String, File I/O, 2D arrays, recursion

Slide 3



# String class

(Section 4.4 in textbook, cont'd)

- **Looking for something in a String object**

- `boolean endsWith(String suffix)`
- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `int indexOf(String str)`
- `int indexOf(String str, int fromIndex)`
- `int lastIndexOf(int ch)`
- `int lastIndexOf(int ch, int fromIndex)`
- `int lastIndexOf(String str)`
- `int lastIndexOf(String str, int fromIndex)`
- `boolean startsWith(String prefix)`

- **Extracting String from a String object**

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

CSC 176 String, File I/O, 2D arrays, recursion

Slide 4



## String class

(Section 4.4 in textbook, cont'd)

- Some more String methods
  - `boolean matches(String regex)`
  - `String replace(char oldChar, char newChar)`
  - `String replaceAll(String regex, String replacement)`
  - `String replaceFirst(String regex, String replacement)`
  - `String[] split(String regex)`

CSC 176 String, File I/O, 2D arrays, recursion

Slide 5

## File Input/Output Processing

(Sections 12.10 through 12.13 in textbook)

- What's a text file?
  - A file that contains letters, digits, and special characters
    - Based on character set used by operating system
  - These files are readable by people
    - Simply open file in a text editor (e.g., Notepad)
  - Each Java source code file is a text file
  - Each web page (i.e., HTML file) is a text file

CSC 176 String, File I/O, 2D arrays, recursion

Slide 6

# File Input/Output Processing

(Sections 12.10 through 12.13 in textbook, cont'd)

- Text File Input Processing

- Scanner

- A Java API class
    - Construct a Scanner object
 

```
Scanner inFile;
inFile = new Scanner(new File("aFileName.txt"));
```
    - Read data using Scanner object
      - Use one or more of the following methods:
        - » One or more of the `next` methods
        - » May also use one or more of the `hasNext` methods
    - When done reading from file
 

```
inFile.close();
```

      - To close the file and release system resources

CSC 176 String, File I/O, 2D arrays, recursion

Slide 7

# File Input/Output Processing

(Sections 12.10 through 12.13 in textbook, cont'd)

- Text File Input Processing (cont'd)

- Sample Scanner methods to read data

- `String next()` `String token = inFile.next();`
    - `BigDecimal nextBigDecimal()`
    - `BigInteger nextBigInteger()`
    - `boolean nextBoolean()`
    - `byte nextByte()`
    - `double nextDouble()`
    - `int nextInt()`
    - `String nextLine()`

- Each of these has a corresponding "has" method

- Discuss assignment 10

CSC 176 String, File I/O, 2D arrays, recursion

Slide 8

# File Input/Output Processing

(Sections 12.10 through 12.13 in textbook, cont'd)

- Text File Output Processing

- PrintWriter

- A Java API class
    - Construct a PrintWriter object
 

```
PrintWriter outFile;
outFile = new PrintWriter("aFileName.txt");
```
    - Write data using PrintWriter object
      - Use one or more of the following methods:
        - » append, format, print, println, write
    - When done writing to file
 

```
outFile.close();
```

      - To close the file and release system resources

CSC 176 String, File I/O, 2D arrays, recursion

Slide 9

# File Input/Output Processing

(Sections 12.10 through 12.13 in textbook, cont'd)

- Text File Output Processing (cont'd)

- Sample PrintWriter methods to write data

- PrintWriter append(char c)
 

```
outFile.append('A');
```
    - void print(char c)
 

```
outFile.print('a');
```
    - void print(double d)
 

```
outFile.print(3.12);
```
    - void print(int i)
 

```
outFile.print(-23);
```
    - void print(Object obj)
    - void print(String s)
 

```
outFile.print("Hello");
```
    - void println()
 

```
outFile.println();
```

      - println methods similar to the print methods
    - void write(char[] buf)
    - void write(String s)

- Discuss assignment 11

CSC 176 String, File I/O, 2D arrays, recursion

Slide 10

# File Input/Output Processing

- Binary File I/O
  - What's a binary file?
    - A file that uses bits and bytes to represent data values
    - These files are NOT readable by people
      - Need specially written software to read/write to these types of files
    - Each Java compiled file (.class) is a binary file
    - Each executable image (.exe) is a binary file
  - Java API classes
    - FileInputStream
    - FileOutputStream

CSC 176 String, File I/O, 2D arrays, recursion

Slide 11

# 2D Arrays

(Chapter 8)

- Syntax
  - Define variable
 

```
dataType[][] arrayVarName;    //preferred
dataType arrayVarName[][];
```
  - Allocate memory
 

```
arrayVarName = new dataType[nbrRows][nbrCols];
```
- Semantics
  - Define a variable that will refer to a memory location where the 2D array data begins
  - First index is row number
    - Ranges from 0 to nbrRows-1
  - Second index is column number
    - Ranges from 0 to nbrCols-1

CSC 176 String, File I/O, 2D arrays, recursion

Slide 12

## 2D Arrays

(Chapter 8, cont'd)



- Example

- Define and initialize an array with 3 rows and 8 columns

```
int[][] numbers = { {1,3,5,7,9,11,13,15},
                    {2,4,6,8,10,12,14,16},
                    {2,3,5,7,11,13,17,19} };
```

- Display contents of this array

```
for (int row=0; row < numbers.length; row++)
{
    for (int col=0; col < numbers[row].length; col++)
        System.out.print(" " + numbers[row][col]);
    System.out.println();
}
```

CSC 176 String, File I/O, 2D arrays, recursion

Slide 13

## 2D Arrays

(Chapter 8, cont'd)



- Another example

- Define and initialize a ragged array

- Each row can have a different number of columns

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

- Display contents of this ragged array

```
for (int row=0; row < numbers.length; row++)
{
    for (int col=0; col < numbers[row].length; col++)
        System.out.print(" " + numbers[row][col]);
    System.out.println();
}
```

CSC 176 String, File I/O, 2D arrays, recursion

Slide 14

## 2D Arrays

(Chapter 8, cont'd)

- Yet another example

- Define an array that can contain any type of object

```
final int MAX = 4;  
Object[][] objs;
```

- Allocate memory

```
objs = new Objects[MAX][MAX];
```

- Store objects in rows 0 and 1 of this array

```
int row = 0;  
for (int col=0; col < MAX; col++)  
    objs[row][col] = new Integer(col * 10);  
row++;  
for (int col=0; col < MAX; col++)  
    objs[row][col] = new Double(col * 0.5);  
//etc.
```

CSC 176 String, File I/O, 2D arrays, recursion

Slide 15

## More on Recursion

- Review

- What is it?
- What should we think about when using recursion?
- What is a helper function?

CSC 176 String, File I/O, 2D arrays, recursion

Slide 16



## More on Recursion

(cont'd)

- **More advanced recursive approaches**
  - Will not implement any of these approaches in this course
  - **Mutual recursion**
    - Method A calls method BAND THEN
    - Method B calls method A
    - This recursion continues until a base case has been reached
  - A recursive algorithm that also uses iteration

