

# CS 302: Introduction to Programming in Java



## Lectures 17&18

It's insanely hot. People desperately  
need some snowflakes

# Static variables Again (class variables)

- Variables unique to the class (all objects of this type will have access to these variables and there is only 1 of each variable)
- Different from regular instance variables
  - All objects of this type get their own copies of the regular instance variables
- Often used for identification of objects

# ArrayLists and Objects

- ArrayLists can be a list of objects

```
ArrayList<BankAccount> accounts = new  
    ArrayList<BankAccount>();
```

- Now can call any of the ArrayList methods (.add(), .get(), etc.) on accounts
- Each index of accounts will be a BankAccount object

```
accounts.get(0).deposit(100);
```

# Chapter 8

- Grab-bag chapter
- 3 Foci:
  - Reading/Writing Files
    - Parsing text files
    - Navigating directories
  - Command Line arguments
  - Exception Handling
    - Throwing exceptions
    - Catching exceptions
- Don't mistake this chapter for being unimportant (material will show up on the final)



# Reading / Writing Text Files

- Use Scanner
  - Scanner takes input from whatever you pass it in its constructor
  - `Scanner stdin = new Scanner(System.in);`
    - `System.in` is default input (i.e. keyboard) from the operating system
  - `File inputFile = new File("input.txt");`
  - `Scanner fileReader = new Scanner(inputFile)`
    - Can now use familiar scanner methods to parse `inputFile` (`next`, `nextLine`, `nextInt`, `nextDouble`)

# The File Object

- Represents a path (not necessarily a file)
- To instantiate:
  - File whatever = new File("path");
  - "Path" must be absolute path ("C:\\folder1\\folder2...\\file.extension");
    - Exception: the file you want to open is in your Java project
    - In this case just give the file name
- Several useful methods (creating files, deleting files, etc.)
  - take a look at the Java API
- Can represent a path where no file is yet
  - File fileToCreate = new File("newFile.txt");
  - fileToCreate.createNewFile();
- Can represent a directory (folder)

# Using Files and Scanner - Input

- What does this loop do?

```
File file = new File("input.txt");
```

```
Scanner fileReader = new Scanner(file);
```

```
while (in.hasNextLine())
```

```
{
```

```
    String line = in.nextLine();
```

```
    //Do something with line
```

```
}
```

# File Output: PrintWriter

- To output data to a file use PrintWriter object
- Can construct in 2 ways:
  - `PrintWriter out = new PrintWriter("filepath");`
  - `PrintWriter out = new PrintWriter(File file);`
- Caution: if file already exists, PrintWriter will clear it completely before printing anything to it
  - If file didn't exist PrintWriter will create it



# PrintWriter Example

```
PrintWriter out = new PrintWriter("out.txt");  
out.println("Hello fileoutput world!");  
out.print("I can use any of the familiar System.out"  
+  
    " calls and conventions");  
out.println("\nIncluding the escape sequence");  
out.close();
```

# Notes on Text input/output

- ALWAYS close your Scanner and PrintWriter objects before exiting your program
  - `scannerName.close(); printWriterName.close();`
  - If you don't, you might lose some data
- FileNotFoundException
  - Can be thrown for a variety of reasons such as:
    - the file you used to construct your Scanner didn't exist
    - the file you gave your PrintWriter had an illegal name
    - you didn't have write permissions to create file
  - To fix: add "throws FileNotFoundException" to the method header in which the file input/output is done

# Processing Text Input

- Read a file word by word:

```
while(in.hasNext())
```

```
{
```

```
    String word = in.next();
```

```
}
```

- Read a file line by line:

```
while(in.hasNext())
```

```
{
```

```
    String line = in.nextLine();
```

```
}
```



# Example

Input file:

Mary had a little lamb.

China 1440044605

India 1147995898

United States 31382464

How can parsing be done using word by word or  
line by line?

# Reading numbers

- Use `in.hasNextInt()`, `in.nextInt()`, `in.hasNextDouble()`, `in.nextDouble()` methods:

```
if (in.hasNextDouble())
```

```
{
```

```
    double val = in.nextDouble();
```

```
}
```

- NOTE: these methods do NOT consume anything that follows a number (whitespace or newline)

# Using Numbers and Words

Code:

```
String country =  
    in.nextLine();
```

```
int population =  
    in.nextInt();
```

```
String nextCountry =  
    in.nextLine();
```

What is the value of  
nextCountry?

Input file:

China

1330044605

India

1147995898

United States

303824646

# Reading Characters

- Can read a single character at a time:

```
Scanner in = new Scanner("whatever.txt");
```

```
in.useDelimiter("");
```

```
while(in.hasNext())
```

```
{
```

```
    char ch = in.next().charAt(0);
```

```
    //do something with ch
```

```
}
```



# Practice 1

- ❖ You have a short poem you made long time ago about Lake Mendota. One day on a whim, you wanna add line numbers for each line in the poem.

- ❖ Input file:

dusk embraces Lake Mendota

Author: Yinggang Huang

When the golden sun was reflected in the water, the lake was ignited with sparks which is soo eye enchanting.

Hordes of ducks were sporting on the lake.

A breeze swept across and then this scene got more dynamic.

Waves and bubbles caused by sailboats danced together forming a visual rhythm.

This state of harmony was just irresistible; it's like all these come back to me again soo naturally, it's really yesterday once more.

- ❖ Write a file named "The Lake of Dreams.txt" with line numbers added into the same folder as the original file.



# Review

- What 2 objects are used for reading from text files
- What object is used to write to text files
- When giving absolute paths, what do you have to be careful of?
- If you don't give an absolute path, where does Java search for the file?
- What must you ALWAYS do when you are done reading / writing to/from files?
- What exception can be thrown when dealing with File IO

# Command Line Arguments

- `public static void main(String[] args)`
- Before IDE's developers used simple text editors to write code (think notepad)
- They then used a shell (think cmd from Windows or Terminal from Mac OS) to compile and run their code (2 steps)
- You can still do this if you like:
- Compile: `javac whatever.java`
  - Will produce a compiled file called "whatever.class"
- Run: `java whatever`
  - Only works if there is already a "whatever.class"

# Command Line Arguments

- Can pass arguments into the main method:
  - `java whatever -v input.txt`
  - Here 2 arguments were passed: "-v" and "input.txt"
  - Arguments go to the `String[]` `args` in the main method
    - `String args` now contains: `["-v", "input.txt"]`
  - Can do anything with the `args` array that you can do with normal arrays
    - Example: check if any arguments were passed in:  
`if (args.length != 0)`

# Command Line Arguments - Eclipse

- Can pass in Command Line Arguments even if you aren't running from a shell
- In Eclipse:
  - Run -> Run...
  - Select "Arguments" tab
  - Enter arguments seperated by spaces

# Intro to Exceptions

- Exceptions = special but common error conditions that arise during runtime
- Two aspects: Detecting and Handling
- So far, our handling has simply been to quit the program, however we can do much more
  - Ex. if we get a `FileNotFoundException`, why not prompt the user to enter a different file
- Detecting and Handling exceptions **MUST** be done separately
  - Ex. Scanner objects can detect `FileNotFoundException`s but cannot deal with them, they simply report it up the hierarchy

# Throwing Exceptions

- If you only want to detect exceptions, your job is easy – just "throw" a new exception object
- The only tricky part is figuring out which type of exception to throw
- Can create your own exceptions, but Java has many built in
- See textbook for common exceptions to throw
- If an exception is thrown, it immediately exits the method (like a return statement)

# Throwing Exceptions Example

- What if we tried to delete a contact from our phonebook that wasn't in the phonebook?
    - A "NoSuchElementException" – lets use it
- ```
if (!contactsList.contains(contactToDelete))  
{  
    throw new NoSuchElementException("Contact  
    " + contactToDelete.getName() + " wasn't in  
    the     phonebook");  
}
```

# Catching Exceptions

- All exceptions should be handled somewhere
  - If an exception is not caught, your program will exit and you will get an error message
- To handle exceptions: use try/catch
- Try surrounds code that might throw an exception
- Catch deals with any exceptions should they arise





# Try/Catch Example

```
Contact aContact = new Contact("Billy", "Bob", 1234567,  
    "nowhere");
```

```
try {  
    phoneBook.remove(aContact);
```

```
}  
catch(NoSuchElementException e)
```

```
{  
    System.out.println("Billy Bob wasn't in the phonebook");
```

```
}
```