Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
The First AI?

- The Turing test
- MIT's ELIZA (1960s)
  - The first computerized psychotherapist

```
EEEEEEE L I I I I I L Z Z Z Z Z A A A
E E E E E L I I I I I I Z A A A
E E E E E L I I I I I I Z A A A
E E E E E L I I I I I I Z A A A
E L L L L L L L L L L L L L L L L L L L
EEEEEEE L I I I I I L Z Z Z Z Z A A A
```

ELIZA > (PLEASE TYPE IN ALL CAPS) WHAT'S YOUR NAME DEAR?
AMIT
ELIZA > HI, I'M ELIZA. WHAT DO YOU WANT TO TALK ABOUT?
AMIT > CAN I TALK ABOUT MY PROBLEM?
ELIZA > SURE...
AMIT > DUE TO ON GOING EXAMS IN COLLEGE, I HAVE A LOT OF STRESS.
ELIZA > PLEASE RELAX A LITTLE. SLEEP WELL.
AMIT > THANKS FOR YOUR ADVICE.
ELIZA > NO MENTION.
AMIT > BYE
ELIZA > BYE AND KEEP IN TOUCH...
```
David's First Job: Tech Support

- David's first job (c. 1995-2001)
  - Technical Support Staff
    Gateway to Internet Services (GWIS)

- Usually I just tried to frustrate customers so they'd hang up and leave me alone.
  - So easy!

- Too easy?
  - Today, A.I. (so called chat bots) are replacing live tech support.

One example of robots taking over jobs:

- I'm having trouble with my internet connection
  - Sure, I can help you with that.
  - Have you tried power-cycling your modem?
    - Yes, I tried that  No, I didn't try that
    - What's that?
  - Power-cycling your modem/router
    - View instructions
      - It worked!  That didn't work
      - How long has it been?
      - It worked!
Let's write (a naive) chat bot

- When a `start()` method is called:
  - Print a welcome message
  - Continuously prompt user for input from terminal, until user enters "bye" *(How?)*
    - Print a random snarky response
    - From a set of David's patented "canned" responses:
      - "No one has ever complained about this before."
      - "I just Googled it. It doesn't know either."
      - "Calm down, I don't want to argue."
      - "You're not making any sense. Could you ask in a different way?"
      - "Say, do you like cats?"
Sample ChatBot Interaction

- What user types is shown in blue.

Tech support! How can I help?
Type 'bye' to end this session.
Hi! I'm so glad you're a real person! Can I get help with my wifi?
I just Googled it. It doesn't know either.
Does anyone there know what the problem might be?
I just Googled it. It doesn't know either.
You said that, but maybe someone in your office might know?
No one has ever complained about this before.
So was it something I did?
Say, do you like cats?
Yes, but what does that have to do with anything?
Say, do you like cats?
bye
Glad I was able to help. Bye!

Can only enter input while your programming is running
Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
Scaner Object to Get User Input

- Need to import the Scanner

```java
import java.util.Scanner;
```

- We can use a Scanner object to get keyboard input!
  - Here's are the relevant methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public Scanner(File source)</code></td>
<td>Constructor. Creates a scanner on the given file.</td>
</tr>
<tr>
<td><code>public Scanner(InputStream source)</code></td>
<td>Constructor. Creates a scanner on the given stream.</td>
</tr>
<tr>
<td><code>public String nextLine()</code></td>
<td>Returns an entire line of input as a String.</td>
</tr>
<tr>
<td><code>public void close()</code></td>
<td>Closes the scanner.</td>
</tr>
</tbody>
</table>
General Scanner Usage Pattern

How Scanners are used:

1. Connect a Scanner object to the keyboard.
2. Print a prompt for the input.
3. Get a line of user-input from the Scanner.
4. Process the input.
5. [Optional] Repeat (2-4) until user enters a keyword to stop.
6. Close the scanner.
Using Scanner

- **Usage example:**

```java
Scanner keyboard = new Scanner(System.in); // Connect a Scanner to the keyboard

String input; // This will hold a line of keyboard input
System.out.print("Enter a phrase: "); // Prompt user
input = keyboard.nextLine(); // Grab a line from the keyboard
System.out.println("You entered: "+input.length()+" characters");

keyboard.close(); // Close the keyboard connection
```

- **On the terminal:**

```
Enter a phrase: I like cats
You entered: 11 characters
```
Repeating Inputs

- You can use a sentinel loop to continuously retrieve input

```java
Scanner keyboard = new Scanner(System.in);

String input;
do {
    System.out.print("Enter a phrase: "); // prompt for input
    input = keyboard.nextLine(); // grab a line from the keyboard
    System.out.println("You entered: "+ input.length()+" characters");
} while (!input.equalsIgnoreCase("quit"));

keyboard.close();
```

Enter a phrase: I like cats
You entered: 11 characters

Enter a phrase: I like dogs too
You entered: 15 characters

Enter a phrase: quit
Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
Smarter ChatBot

Chatbot makes for a good laugh, but isn't very helpful.

- Basically ignores what the user's problems are

Smarter ChatBot:

```
Tech support! How can I help?
Type 'bye' to end this session.
> I ran a program I wrote, but nothing's happening and my computer's really hot
One of your programs is stuck in an infinite loop. Terminate the offending program.
> Thanks that TOTALLY worked!
You're welcome.
> Well I think there's still another bug in my code though...
It's not a bug, it's a feature!
> You're right, I'll tell Professor Chiu that. I'm done with my HW for the week! Lol
You're welcome.
> bye
Glad I was able to help!
```
Smarter ChatBot:

- First: Associate keywords with more detailed answers:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Our Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;wifi&quot;, &quot;network&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;slow&quot;, &quot;sluggish&quot;</td>
<td>&quot;I recommend installing more memory!&quot;</td>
</tr>
<tr>
<td>&quot;heat&quot;, &quot;hot&quot;,  &quot;scalding&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop. Terminate the offending program.&quot;</td>
</tr>
<tr>
<td>&quot;bug&quot;</td>
<td>&quot;It's not a bug; it's a feature!&quot;</td>
</tr>
<tr>
<td>&quot;thanks&quot;</td>
<td>&quot;You're welcome. Type 'bye' if you're satisfied, or ask another question.&quot;</td>
</tr>
</tbody>
</table>

- Second: Look for these keywords in the user's question!
  - Hmm, need to split their question into words first

- Third: Choose proper response based on keyword
Smarter ChatBot

First: Associate keywords with more detailed answers:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Our Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;wifi&quot;, &quot;network&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;slow&quot;, &quot;sluggish&quot;</td>
<td>&quot;I recommend installing more memory!&quot;</td>
</tr>
<tr>
<td>&quot;heat&quot;, &quot;hot&quot;,</td>
<td>&quot;One of your programs is probably stuck in an infinite loop. Terminate the</td>
</tr>
<tr>
<td>&quot;scalding&quot;</td>
<td>offending program.&quot;</td>
</tr>
<tr>
<td>&quot;bug&quot;</td>
<td>&quot;It's not a bug; it's a feature!&quot;</td>
</tr>
<tr>
<td>&quot;thanks&quot;</td>
<td>&quot;You're welcome. Type 'bye' if you're satisfied, or ask another question.&quot;</td>
</tr>
</tbody>
</table>

Second: Look for these keywords in the user's question!
- Hmm, need to split their question into words first

Third: Choose proper response based on keyword
Problem: What Data Structure Makes Sense?

- We know *Arrays* and *ArrayLists*
  - Main similarities between them?
    - Both are lists
    - Elements are *indexed* by an integer
  - But do integers always make sense for the index?

- Consider coding up a contact directory...
What We Really Need: Make Our Own Index

- We want to index by name (which are Strings), not by integer
  - Still want to store emails (Strings)

- Can we make our own index? These "collections" are called Maps

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;brad&quot;</td>
<td>&quot;<a href="mailto:brad@pugetsound.edu">brad@pugetsound.edu</a>&quot;</td>
</tr>
<tr>
<td>&quot;adam&quot;</td>
<td>&quot;<a href="mailto:adam@pugetsound.edu">adam@pugetsound.edu</a>&quot;</td>
</tr>
<tr>
<td>&quot;tony&quot;</td>
<td>&quot;<a href="mailto:tony@pugetsound.edu">tony@pugetsound.edu</a>&quot;</td>
</tr>
<tr>
<td>&quot;david&quot;</td>
<td>&quot;<a href="mailto:david@pugetsound.edu">david@pugetsound.edu</a>&quot;</td>
</tr>
</tbody>
</table>
```
Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
The Map Collection

- Java's Map collections associate a key (index) to a value
  - Important: You declare the data types of the key and the value

- There are several Map classes in Java, we'll study the HashMap
  - To use it, you must first import the HashMap class:

```java
import java.util.HashMap;
```
Using HashMaps

- HashMap constructor also uses the diamond notation:

\[
\text{HashMap<K,V> myMap = new HashMap<>();}
\]

- \(K\) is the key's data type (what's being used to index)
- \(V\) is the value's data type (what's being stored)

- Example: to create a HashMap that indexes on names (Strings) and stores emails (Strings also):

\[
\text{HashMap<String,String> myContacts = new HashMap<>();}
\]
### Useful HashMap Methods

Some key methods from the **HashMap API**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public HashMap&lt;&gt;()</code></td>
<td>Constructs a new HashMap that maps indices of type K to values of type V.</td>
</tr>
<tr>
<td><code>public V put(K key, V value)</code></td>
<td>Associates given value with given key in the map</td>
</tr>
<tr>
<td><code>public void clear()</code></td>
<td>Removes all elements from list</td>
</tr>
<tr>
<td><code>public boolean containsKey(K key)</code></td>
<td>Searches for the given key in the map</td>
</tr>
<tr>
<td><code>public boolean containsValue(V value)</code></td>
<td>Searches for the given value in the map</td>
</tr>
<tr>
<td><code>public V get(K key)</code></td>
<td>Returns the element at given key, or null if no such key exists</td>
</tr>
<tr>
<td><code>public V remove(K key)</code></td>
<td>Removes the element at given key. Returns the deleted element</td>
</tr>
<tr>
<td><code>public int size()</code></td>
<td>Returns the number elements in the current map</td>
</tr>
</tbody>
</table>

For the full HashMap API: [https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html](https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html)
Example Usage (Code Pad)

```java
HashMap<String, String> myContacts;
myContacts = new HashMap<>();
```
Example Usage (Code Pad)

```java
HashMap<String,String> myContacts;
myContacts = new HashMap<>();

myContacts.put("brad", "brad@pugetsound.edu");
myContacts.put("adam", "adam@pugetsound.edu");
myContacts.put("tony", "tony@pugetsound.edu");
myContacts.put("david", "dadaavid@pugetsound.edu");
myContacts.put("david", "david@pugetsound.edu"); // Maps don't allow duplicate keys
System.out.println(myContacts.size()); // Duplicate values allowed
> 4

myContacts.put("adam (work)", "adam@pugetsound.edu"); // as long as key is unique
```
Example Usage (Code Pad)

```java
HashMap<String, String> myContacts;
myContacts = new HashMap<>();

myContacts.put("brad", "brad@pugetsound.edu");
myContacts.put("adam", "adam@pugetsound.edu");
myContacts.put("tony", "tony@pugetsound.edu");
myContacts.put("david", "daaaavid@pugetsound.edu");
myContacts.put("david", "david@pugetsound.edu"); // Maps don't allow duplicate keys
System.out.println(myContacts.size());
> 4

myContacts.put("adam (work)", "adam@pugetsound.edu"); // Duplicate values allowed
    // as long as key is unique
String email = myContacts.get("david");
System.out.println(email);
> david@pugetsound.edu
```
Example Usage (Code Pad)

```java
HashMap<String, String> myContacts;
myContacts = new HashMap<>();

myContacts.put("brad", "brad@pugetsound.edu");
myContacts.put("adam", "adam@pugetsound.edu");
myContacts.put("tony", "tony@pugetsound.edu");
myContacts.put("david", "daaaavid@pugetsound.edu");
myContacts.put("david", "david@pugetsound.edu"); // Maps don't allow duplicate keys
System.out.println(myContacts.size());
> 4

myContacts.put("adam (work)", "adam@pugetsound.edu"); // Duplicate values allowed
    // as long as key is unique

String email = myContacts.get("david");
System.out.println(email);
> david@pugetsound.edu

email = myContacts.get("David"); // Find someone who doesn't exist!
System.out.println(email); // Keys are case sensitive!
> null
```
Smarter ChatBot:

- **First: Associate keywords with more detailed answers:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Our Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;wifi&quot;, &quot;network&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;slow&quot;, &quot;sluggish&quot;</td>
<td>&quot;I recommend installing more memory!&quot;</td>
</tr>
<tr>
<td>&quot;heat&quot;, &quot;hot&quot;, &quot;scalding&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop. Terminate the offending program.&quot;</td>
</tr>
<tr>
<td>&quot;bug&quot;</td>
<td>&quot;It's not a bug; it's a feature!&quot;</td>
</tr>
<tr>
<td>&quot;thanks&quot;</td>
<td>&quot;You're welcome. Type 'bye' if you're satisfied, or ask another question.&quot;</td>
</tr>
</tbody>
</table>

- **Second: Look for these keywords in the user's question!**
  - Hmm, need to split their question into words first

- **Third: Choose proper response based on keyword**
Employ a HashMap!

- Map keywords to relevant responses!

**Update code to fill a HashMap!**

```java
responseMap

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;wifi&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;network&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;slow&quot;</td>
<td>&quot;I recommend installing more memory!&quot;</td>
</tr>
<tr>
<td>&quot;heat&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop...&quot;</td>
</tr>
<tr>
<td>&quot;hot&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop...&quot;</td>
</tr>
<tr>
<td>&quot;fan&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop...&quot;</td>
</tr>
<tr>
<td>&quot;bug&quot;</td>
<td>&quot;It's not a bug; it's a feature!&quot;</td>
</tr>
<tr>
<td>&quot;thanks&quot;</td>
<td>&quot;You're welcome. Type 'bye' if you're satisfied, or ask another...&quot;</td>
</tr>
</tbody>
</table>
```
Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
Smarter ChatBot:

- First: Associate keywords with more detailed answers:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Our Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;wifi&quot;, &quot;network&quot;</td>
<td>&quot;Have you tried resetting your router?&quot;</td>
</tr>
<tr>
<td>&quot;slow&quot;, &quot;sluggish&quot;</td>
<td>&quot;I recommend installing more memory!&quot;</td>
</tr>
<tr>
<td>&quot;heat&quot;, &quot;hot&quot;, &quot;scalding&quot;</td>
<td>&quot;One of your programs is probably stuck in an infinite loop. Terminate the offending program.&quot;</td>
</tr>
<tr>
<td>&quot;bug&quot;</td>
<td>&quot;It's not a bug; it's a feature!&quot;</td>
</tr>
<tr>
<td>&quot;thanks&quot;</td>
<td>&quot;You're welcome. Type 'bye' if you're satisfied, or ask another question.&quot;</td>
</tr>
</tbody>
</table>

- Second: Look for these keywords in the user's question!
  - Hmm, need to split their question into words first

- Third: Choose proper response based on keyword
Finding Keywords

- Users enter queries as Strings
  - **Want:** Add a method to SmartBot
    ```java
    public void printResponse(String query)
    ```
    - Loops through each word, look in the map for a response
    - If none of the words is in the map, print random snarky comment as before
  - **Problem:** Need to extract each word from a query.
    - Process is called string splitting or string tokenizing

- From the String API:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public String[] split(String delim)</code></td>
<td>Tokenizes the current String by the given delimiter. Return an array of tokens.</td>
</tr>
</tbody>
</table>
Using String's `split()` Method

Example

```java
String str = "Hello! My computer's acting slow";
String[] words = str.split(" "); // tokenizes by whitespace
```
Using String's `split()` Method

- **Example**

```java
String str = "Hello! My computer's acting slow";
String[] words = str.split(" ");  //tokenizes by whitespace

System.out.println(words.length);  //how many tokens?
> 5
```
Using String's `split()` Method

Example

```java
String str = "Hello! My computer's acting slow";
String[] words = str.split(" "); //tokenizes by whitespace

System.out.println(words.length); //how many tokens?
> 5

System.out.println(words[0]);
> "Hello!"

System.out.println(words[1]);
> "My"

System.out.println(words[2]);
> "computer's"

System.out.println(words[3]);
> "acting"

System.out.println(words[4]);
> "slow"
```
Users enter queries as Strings

- **Want**: Add a method to SmartBot
  ```java
  public void printResponse(String query)
  ```
  - Loops through each word, look in the map for a response
  - If none of the words is in the map, print random snarky comment as before

*Update SmartBot code to add this method!*
Outline

- David's First Job
- Scanner Class
- Smarter ChatBot
- HashMap Class
- Tokenizing Strings
- Conclusion
Conclusion

- Scanner is used to get user input straight from the keyboard.
  - And later, from files!
  - Usually a sentinel loop is used to prompt and retrieve input from terminal
    - And commonly a do-while.

- HashMap is a new collection that lets you create your own indices.

- Strings can be tokenized using their `split()` method
Administrivia 3/29

- Canvas still down
- Posted
  - Homework 5 solution
  - Homework 6: Tweet Processor
    - Doesn't need Scanner, HashMaps
    - Needs an ArrayList
- Lab 8 (Life) Post-mortem
  - Abstraction and modularization: Life should use Cell!
  - Even after instantiation, arrays are still empty!
- Today: User interaction with Scanner
Administrivia 4/3

- Reminder:
  - Review next Wednesday
  - Exam 2 next Thursday

- Hwk 6: TweetStats due tonight!
  - Questions about implementation?

- Lab tomorrow:
  - HashMaps!