IMPORTANT DATES

• Midterm this Thursday (11/2) 7-9PM !!!!

• Assignment 4 Hangman → Monday *November 6, 1:30PM*
Assignment 4: Hangman

• Due Monday *November 6, 1:30PM*

• Good practice with multiple classes and strings

• Do it in parts!
Quick Review....
Classes? Objects? Instances?
Classes and Instances

• A **class** is like a dictionary entry for something – it defines what something is supposed to do

• An **instance** is an *actual copy* of what that entry describes
“OBJECT”

= 

INSTANCE OF A CLASS

car
/kær/ (n)
noun
a road vehicle, typically with four wheels, powered by an internal combustion engine and able to carry a small number of people.
"we're going by car"
synonyms: automobile, motor vehicle, vehicle; More
• a railroad vehicle for passengers or freight.
  "the first-class cars"
synonym: carriage, coach
  "the dining car"
• the passenger compartment of an elevator, cableway, airship, or balloon.
Hangman Blueprint

Hangman

HangmanCanvas

HangmanLexicon
import acm.util.*;

public class HangmanLexicon {

    /** Returns the number of words in the lexicon. */
    public int getWordCount() {
        return 10;
    }

    /** Returns the word at the specified index. */
    public String getWord(int index) {
        switch (index) {
            case 0: return "BUOY";
            case 1: return "COMPUTER";
            case 2: return "CONNOISSEUR";
            case 3: return "DEHYDRATE";
            case 4: return "FUZZY";
            case 5: return "HUBBUB";
            case 6: return "KEYHOLE";
            case 7: return "QUAGMIRE";
            case 8: return "SLITHER";
            case 9: return "ZIRCON";
            default: throw new ErrorException("getWord: Illegal index");
        }
    }
}
Creating An Instance

HangmanLexicon lexicon =
    new HangmanLexicon();

int wordCount =
    lexicon.getWordCount();  // 10

String word =
    lexicon.getWord(0);  // BUOY
Part I: Console Game

• Choose a random word
• Keep track of partially-guessed word
• Game structure – guess, guesses remaining, messages, game end, etc.
chars and Strings
Characters

c char ch = ‘a’;
ch = Character.toUpperCase(ch);
String str = “” + ch; // char -> string
println(str);

Can’t just write (for line 2):
Character.toUpperCase(ch);
## Useful Methods in the `Character` Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static boolean isDigit(char ch)</code></td>
<td>Determines if the specified character is a digit.</td>
</tr>
<tr>
<td><code>static boolean isLetter(char ch)</code></td>
<td>Determines if the specified character is a letter.</td>
</tr>
<tr>
<td><code>static boolean isLetterOrDigit(char ch)</code></td>
<td>Determines if the specified character is a letter or a digit.</td>
</tr>
<tr>
<td><code>static boolean isLowerCase(char ch)</code></td>
<td>Determines if the specified character is a lowercase letter.</td>
</tr>
<tr>
<td><code>static boolean isUpperCase(char ch)</code></td>
<td>Determines if the specified character is an uppercase letter.</td>
</tr>
<tr>
<td><code>static boolean isWhitespace(char ch)</code></td>
<td>Determines if the specified character is whitespace (spaces and tabs).</td>
</tr>
<tr>
<td><code>static char toLowerCase(char ch)</code></td>
<td>Converts <code>ch</code> to its lowercase equivalent, if any. If not, <code>ch</code> is returned unchanged.</td>
</tr>
<tr>
<td><code>static char toUpperCase(char ch)</code></td>
<td>Converts <code>ch</code> to its uppercase equivalent, if any. If not, <code>ch</code> is returned unchanged.</td>
</tr>
</tbody>
</table>
Comparing chars

Let's write a program that:
- prompts the user for 2 words
- print out “they match” if the first letters of the two words are the same
Solution

String first = readLine("Enter a word: ");
String second = readLine("Enter another: ");

if(Character.toLowerCase(first.charAt(0)) ==
   Character.toLowerCase(second.charAt(0))) {
   println("The first letters match!");
} else {
   println("The first letters differ.");
}

Still 1 edge case to cover here! – EMPTY STRING
Strings

String s = "Hello!";
s = s.toUpperCase();
println(s); // prints HELLO!

Can’t just write (for line 2):
s.toUpperCase();
# Useful Methods in the String Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int length()</td>
<td>Returns the length of the string</td>
</tr>
<tr>
<td>char charAt(int index)</td>
<td>Returns the character at the specified index. Note: Strings indexed starting at 0.</td>
</tr>
<tr>
<td>String substring(int p1, int p2)</td>
<td>Returns the substring beginning at p1 and extending up to but not including p2</td>
</tr>
<tr>
<td>String substring(int p1)</td>
<td>Returns substring beginning at p1 and extending through end of string.</td>
</tr>
<tr>
<td>boolean equals(String s2)</td>
<td>Returns true if string s2 is equal to the receiver string. This is case sensitive.</td>
</tr>
<tr>
<td>int compareTo(String s2)</td>
<td>Returns integer whose sign indicates how strings compare in lexicographic order</td>
</tr>
<tr>
<td>int indexOf(char ch) or int indexOf(String s)</td>
<td>Returns index of first occurrence of the character or the string, or -1 if not found</td>
</tr>
<tr>
<td>String toLowerCase() or String toUpperCase()</td>
<td>Returns a lowercase or uppercase version of the receiver string</td>
</tr>
</tbody>
</table>

*Using portions of slides by Eric Roberts*
Take 2

String first = readLine(“Enter a word: ”).toLowerCase();
String second = readLine(“Enter another: ”).toLowerCase();

If(first.charAt(0) == second.charAt(0)) {
    println(“The first letters match!”);
} else {
    println(“The first letters differ.”);
}

Still 1 edge case to cover here! – EMPTY STRING
Comparing Strings

String s1 = “racecar”;
String s2 = reverseString(s1);
// How do we check equality?
if(s1.equals(s2)) {
    ....
}

--------OR--------

if(s2.equals(s1)) {
    ....
}
Don’t do this!

```java
String s1 = "racecar";
String s2 = reverseString(s1);
// How do we check equality?

if(s1 == s2) {
    ...
}
```
Searching Strings

• Search using the `indexOf` method: `string.indexOf(pattern)`
• `indexOf` returns the start index of the first occurrence of pattern, if one exists.
• Otherwise, it returns -1.

```java
int index = "hello".indexOf("el"); // 1
```
Building Strings

• 1) Use substrings – smaller pieces of strings

OR

• 2) Make new string, build up over time
1) Obtaining Substrings

- To get all of the characters in the range [start, stop), use
  
  ```
  string.substring(start, stop)
  ```

- To get all of the characters from some specified point forward, use
  
  ```
  string.substring(start)
  ```

```

```

str.substring(0, 2);  str.substring(6);
```
2) Building a New String

• Start with nothing and build up a new string
• Iterate through the old string
• Use Character methods at each position to decide what to concatenate to the new string
• See this week’s section handout for examples
## Game Flow

<table>
<thead>
<tr>
<th>String secretWord</th>
<th>S E C R E T</th>
</tr>
</thead>
<tbody>
<tr>
<td>String wordState</td>
<td>_ _ _ _ _ _ _ _ _</td>
</tr>
<tr>
<td>char guess</td>
<td>e</td>
</tr>
<tr>
<td>String newWordState</td>
<td>- E - - E -</td>
</tr>
</tbody>
</table>

(most important slide!)
Guess 2

String secretWord  S E C R E T

String wordState   - E - - E -

char guess  \text{note: guesses are case-insensitive} R

String newWordState - E - R E -

\text{?? int guessesLeft ??}
User Guesses

• Case-insensitive
• Only 1 letter guesses allowed
• Re-guess correct guess – do nothing
• Re-guess incorrect guess – another wrong guess!
Part II: Graphics
public void reset() {
...
}

public void displayWord(String word) {
...
}

public void noteIncorrectGuess(char guess) {
...
}
HangmanCanvas Usage

In hangman.java:

```java
private HangmanCanvas canvas;
...

public void init() {
    canvas = new HangmanCanvas();
    add(canvas);
}
```
Part III: Files

- BufferedReader – open, read, close
- try/catch
- Read in line by line and store all lines in an ArrayList
- “catch” an error if there is one
- close your BufferedReader!
  - .close()
try {
    BufferedReader rd = new BufferedReader(new FileReader("test.txt"));

    while(true) {
        String line = rd.readline();
        if(line == null) break;
        println(line); // do something with line
    }
    rd.close(); // close when you’re done!
} catch (IOException ex) {
    // do something in response to exception
    throw newErrorException(ex);
}
ARE YOU AN EXCEPTION?
BECAUSE I CAN'T WAIT TO CATCH YOU.
Constructors

```java
public class HangmanLexicon {

    // This is the HangmanLexicon constructor
    public HangmanLexicon() {
        // your initialization code goes here
    }

    // rest of HangmanLexicon class...
}
```

HangmanLexicon lexicon =
    new HangmanLexicon(); // triggers HangmanLexicon constructor above
Constructors

- A **constructor** is a special method defined in a class that is responsible for setting up class's instance variables to appropriate values.
- Syntax:

  ```java
  public NameOfClass(parameters) {
    /* ... body of constructor ... */
  }
  ```
- Inside a constructor:
  - Give initial values to instance variables.
  - Set up instance variables based on values specified in the parameters.
- Constructor called when instance created with `new`. 
Testing/Coding Tips

• Manually set the word to guess so you know what it is each game
• Watch the cases of your strings/chars!
• Add extra printlns along the way if you want to know what your string or char variables are
Final Tips

• Follow the specifications carefully
• Extensions! Graphics, etc.
• Comment!
• Go to the LaIR if you get stuck
• Incorporate IG feedback!

• Have fun!