Objectives

- More Java fundamentals
  - java.lang classes: Math and String class
  - Control Structures
  - Arrays

Review: Assign 0

- How did it go?
  - How long did it take?

- Linux:
  - What is the syntax of the `cp` command?
  - How do you copy an entire directory?
  - How do you make a directory?
  - How do you view the contents of a directory?

- My conventions

Review

- How do we display output in Java?
- What are some of the primitive data types of Java?
- What is the syntax for declaring a variable in Java?

Review: Python Transition **Warning**

You cannot **redeclare** a variable name in the same scope

- **OK:**
  ```java
  int x = 3;
  x = -3;
  ```

- **Not OK:**
  ```java
  int x = 3;
  int x = -3;
  boolean x = true;
  ```

What is the output?

```java
System.out.println("The answer is " + 100L + 3.3f);
```

a) The answer is 1003.3  
b) The answer is 103.3  
c) The answer is 100  
d) Error (no output).
Java Libraries

• Organized into a hierarchy of packages
  java
    lang
      Object
      String
      net
      util
    Arrays
    java
    org

Java API Documentation

• API: Application Programming Interface

  • What the class can do for YOU!
  • Complete documentation of every class included with the JDK
  • Every method and variable contained in class
  • http://docs.oracle.com/javase/8/docs/api
  • Bookmark it!

  • Too many classes, methods to remember them all
  • Refer to it often

java.math.Math class

• Similar to Python’s math module
• Included by default in every Java program
• Contains useful mathematical functions (methods) and constants (fields):

• Look at java.math.Math API online

  ➢ http://docs.oracle.com/javase/8/docs/api/
  ➢ Note how API is specified

java.lang.String class

• Similar functionality to Python but different ways to use
• Strings are represented by double quotes: ""
  ➢ Single quotes represent chars only
• Examples:

  String emptyString = "";
  String niceGreeting = "Hello there.";
  String badGreeting = "What do you want??";

Strings

• A char at each position of String
  stringvar = "The Beatles"

<table>
<thead>
<tr>
<th>'T'</th>
<th>'h'</th>
<th>'e'</th>
<th>' '</th>
<th>'B'</th>
<th>'e'</th>
<th>'a'</th>
<th>'t'</th>
<th>'l'</th>
<th>'e'</th>
<th>'s'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

  ➢ End at stringvar.length()-1
  ➢ Start at 0
  ➢ Length of the string: 11

  ➢ Use charAt method to access chars

Sept 16, 2015  Sprenkle - CSCI209
String methods: substring

• Like slicing in Python
• String substring(int beginIndex)
  ➢ Returns a new String that is a substring of this string, from beginIndex to end of this string
• String substring(int beginIndex, int endIndex)
  ➢ Returns a new String that is a substring of this string, from beginIndex to endIndex-1

```java
String language = "Java!";
String subStr = language.substring(1);
String subStr2 = language.substring(2, 4);
```

String Concatenation

• Use + operator to concatenate Strings

```java
String niceGreeting = "Hello";
String firstName = "Clark";
String lastName = "Kent";
String blankSpace = " ";

String greeting = niceGreeting + "," + blankSpace + firstName + blankSpace + lastName;
System.out.println(greeting);
```

Prints "Hello, Clark Kent"

String Concatenation

• If a String is concatenated with something that is not a String, the other variable is converted to a String automatically.

```java
int totalPoints = 110;
int earnedPoints = 87;
float testScore = (float) earnedPoints/totalPoints;
System.out.println("Your score is "+ testScore);
```

StringBuilders vs Strings

• Strings are “read-only” or immutable
  ➢ Same as Python
• Use StringBuilder to manipulate a String
  ➢ Instead of creating a new String using
  ➢ String str = prevStr + " more!"
  ➢ Use
  ➢ new keyword: allocate memory to a new object

```java
StringBuilder str = new StringBuilder(prevStr);
str.append(" more!");
```

• Many StringBuilder methods
  ➢ toString() to get the resultant string back

Effective Java: Code Inefficiency

• Avoid creating unnecessary objects:

```java
String s = new String("text"); // DON'T DO THIS
```

• Do this instead:

```java
String s = "text";
```

Why?

String Comparison: equals

• boolean equals(Object anObject)
  ➢ Compares this string to the specified object

```java
String string1 = "Hello";
String string2 = "hello";

boolean test;

test = string1.equals(string2);
```

• test is false because the Strings contain different values
**Python Gotcha: String Comparisons**

- `string1 == string4` will **not** yield the same result as `string1.equals(string4)`
  - `==` tests if the objects are the same
  - `!=` if the contents of the objects are the same
  - Similar to `is` operator in Python

- `string1 != string4` but `string1.equals(string4)`

---

**String methods: and many more!**

- `boolean` `endsWith(String suffix)`
- `boolean` `startsWith(String prefix)`
- `int` `length()`
- `String` `toLowerCase()`
- `String` `trim()`: remove trailing and leading white space
  - `...`
  - See `java.lang.String API` for all

---

**Review**

- What is the syntax of a *conditional statement* in Python?

---

**Control Flow: Conditional Statements**

- `if` statement
  - **Condition** must be surrounded by `()`
  - Condition must evaluate to a `boolean`
  - Body is enclosed by `{}` if multiple statements

```
if (purchaseAmount < availCredit) {
    System.out.println("Approved");
    availableCredit -= purchaseAmount;
} else
    System.out.println("Denied");
```

- Everything between `{ }` is a block of code
  - `Block of code`
  - **Has an associated scope**
Apple’s goto fail in SSL

```java
hashOut.data = hashes[SSL_MD5_DIGEST_LEN];
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0) goto fail;
if ((err = ReadyHash(SSLHashSHA1, &hashCtx)) != 0) goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0) goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0) goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0) goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0) goto fail;
```
Control Flow: \texttt{switch} statement
\begin{itemize}
\item Like a big \texttt{if/else if} statement
\item Works with variables with datatypes \texttt{byte}, \texttt{short}, \texttt{char}, and \texttt{int} (And \texttt{String} in Java 7)
\end{itemize}
\begin{verbatim}
int x = 3;
switch(x) {
    case 1:
        System.out.println("It's a 1.");
        break;
    case 2:
        System.out.println("It's a 2.");
        break;
    default:
        System.out.println("Not a 1 or 2.");
}
\end{verbatim}

Control Flow: \texttt{while} Loops
\begin{itemize}
\item \texttt{while} loop
\begin{itemize}
\item Condition must be enclosed in parentheses
\item Body of loop must be enclosed in \{\} if multiple statements
\end{itemize}
\begin{verbatim}
int counter = 0;
while (counter < 5) {
    System.out.println(counter);
    counter++;
} System.out.println("Done: "+ counter);
\end{verbatim}
\end{itemize}

Review
\begin{itemize}
\item How do you write a \texttt{for} loop in Python for counting?
\end{itemize}

Control Flow: \texttt{for} Loop
\begin{itemize}
\item Very different syntax from Python
\item Syntax:
\begin{verbatim}
for (<init>; <condition>; <execution_expr>)
\end{verbatim}
\end{itemize}

Changing control flow: \texttt{break}
\begin{itemize}
\item Exits the current loop
\begin{verbatim}
while (<reading data>) {
    if (<something bad>) {
        // shouldn't happen
        break;
    }
}\end{verbatim}
\end{itemize}
Control Flow: `for` Loop Example

```java
System.out.println("Counting down...");
for (int count=5; count >= 1; count--) {
    System.out.println(count);
} // shortcut
System.out.println("Blastoff!");
```

- What is the counter variable?
- What is the condition?
- What is the output?
- How written in Python?

### Arrays

Python Lists → Java Arrays

- A Java array is like a fixed-length list
- To declare an array of integers:
  ```java
  int[] array0fInts;
  ```
  - Declaration only makes a variable named `array0fInts`
  - Does not initialize array or allocate memory for the elements
- To declare and initialize an array of integers:
  ```java
  int[] array0fInts = new int[100];
  ```
  - new keyword: allocate memory to a new object

Array Initialization

- Initialize an array at its declaration:
  ```java
  int[] fibNums = {1, 1, 2, 3, 5, 8, 13};
  ```
  - Note that we do not use the `new` keyword when allocating and initializing an array in this manner
- `fibNums` has length 7

Array Access

- Access a value in an array as in Python:
  ```
  fibNums[0]
  fibNums[x] = fibNums[x-1] + fibNums[x-2]
  ```
- Unlike in Python, cannot use negative numbers to index items

Array Length

- All array variables have a field called `length`
  ```java
  int[] array = new int[10];
  for (int i = 0; i < array.length; i++) {
      array[i] = i*2;
  }
  for (int i = array.length-1; i >= 0; i--) {
      System.out.println(array[i]);
  }
  ```
  - Note: no parentheses because not a method
Overstepping Array Length
- Java safeguards against overstepping length of array
  - Runtime Exception: "Array index out of bounds"
  - More on exceptions later...

Example
```java
int[] array = new int[100];
```
- Attempts to access or write to index < 0 or index >= array.length (100) will generate exception

Command-Line Arguments
- Similar to Python’s `sys` module
- Make sure there are sufficient arguments.
- If len(sys.argv) < 2:
  - print "Error: invalid number of command-line arguments"
  - print "Usage: python", sys.argv[0], "<filename>"
  - sys.exit(1)

```
public static void main(String[] args) {

  if (args.length < 1) {
    System.out.println("Error: invalid number of arguments");
    System.exit(1);
  } else {
    System.out.println("Usage: java MyProgram <filename>");
    System.exit(1);
  }

  Example Use:
  java MyProgram filename
```

Arrays
- Assigning one array variable to another ➔ both variables refer to the same array
  - Similar to Python
- Draw picture of below code:
  ```java
  int[] fibNums = {1, 1, 2, 3, 5, 8, 13};
  int[] otherFibNums;
  otherFibNums = fibNums;
  fibNums[2] and otherFibNums[2] are both equal to 99
  System.out.println(otherFibNums[2]);
  System.out.println(fibNums[2]);
  ```

Control Flow: `foreach` Loop
- Introduced in Java 5
  - Sun calls "enhanced for" loop
  - Iterate over all elements in an array (or Collection)
  - Similar to Python’s for loop
  ```java
  int[] a;
  int result = 0;
  for (int i : a) {
    result += i;
  }
  ```

Array Copying
- Copy an array (element-by-element) using the `arraycopy` method in the `System` class
  ```java
  System.arraycopy(fromIndex, to, count);
  ```
  For example:
  ```java
  int[] fibNums = {1, 1, 2, 3, 5, 8, 13};
  int[] otherNums = new int[fibNums.length];
  System.arraycopy(fibNums, 8, otherNums, 0, fibNums.length);
  otherNums[2] = 99;
  System.out.println(otherNums[2]);
  System.out.println(fibNums[2]);
  ```
**java.util.Arrays**

- Arrays is a class in `java.util`
- Methods for sorting, searching, `deepEquals`, fill arrays
- To use class, need `import` statement
  - Goes at top of program, before class definition

```java
import java.util.Arrays;
```

---

**To Do**

- Assign 1
  - Part 1: Fixing compiler and logic errors from program
  - Part 2: Writing a program to compute a gymnastics score at the Olympics
  - Part 3: Reverse a string
    - EC opportunity
  - Due Friday before class

---

**ArraysExample.java**