Announcements

• Final exam is Saturday June 7, 3-6pm in this room. We have not heard that anyone has conflicts so we will assume you will be there at that time.

• Survey for BONUS HW points! Due 8am on Monday, June 9 (same time as your CAPEs)
  https://www.surveymonkey.com/s/CSE12Sp14_Alvarado
private Map<String, Integer> units;  // The units with exponents
// ... a little later in the code
this.units = new HashMap<String, Integer>();

Hash Tables (HashMaps)
Implementing the Map interface with Hash Tables
Imagine you want to look up your neighbors’ names, based on their house number

- House numbers: 2555 through 10567 (roughly 4000 houses)
- Names: one last name per house
Array vs Tree

- You could store them in a balanced TreeMap of some kind
- Log(n) to do get, put, delete

- Or you could store them in an array
- Array is really fast lookup! O(1)
- Just look in myarray[housenumber] to get the name
Hash Table is just a modified, more flexible array

- Keys don’t have to be integers 0-(size-1)
- (Ideally) avoids big gaps like our gap from 0 to 2555 in the house numbers

- Hash function is what makes this all work:
Hash key collisions

• Hash function takes key and maps it to an integer
• Sometimes will map two DIFFERENT keys to the same integer
  • “Collision”
• We can NOT overwrite the value the way we would if it really were the same key
• Need a way of storing multiple values in a given “place” in the hash table
Closed Hashing (Open addressing) with Linear Probing

- Where does “Annie” go if hashkey(“Annie”) = 3?
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. Other

<table>
<thead>
<tr>
<th>Array index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&quot;Annie&quot;</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Closed Hashing (Open addressing) with Linear Probing

Where does “Juan” go if hashkey(“Juan”) = 4?

A. 1  B. 2  C. 3  D. 4  E. Other

<table>
<thead>
<tr>
<th>Array index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Annie</td>
</tr>
<tr>
<td>4</td>
<td>“Juan”</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Where does “Julian” go if hashkey(“Julian”) = 3?

A. 1  
B. 2  
C. 3  
D. 4  
E. Other
Where does “Solange” go if hashkey(“Solange”) = 5?

A. 3
B. 4
C. 5
D. 6
E. Other

<table>
<thead>
<tr>
<th>Array index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Annie</td>
</tr>
<tr>
<td>4</td>
<td>Juan</td>
</tr>
<tr>
<td>5</td>
<td>Julian</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
C for Java Programmers

Slides based on those by Philip Papadopoulos
A short history of C

• Invented in the early 1970s by Brian Kernighan and Dennis Ritchie (AT&T)
  • Affectionately known as “K&R” C (1977)
• The core of the Unix operating systems and was co-developed as Unix was being developed.
• In the mid/late 80s it became an ANSI standard language 1988
  • Book is 274 pages.
• C as a language is stable, and has highly-optimized compilers.
• It is a small
Why C, Why Learn C?

• C is sometimes thought of as a “high-level” assembly language
  • It’s structure mirrors the fundamental architecture of CPUs
  • It’s portable across different CPUs
    • for the most part, C programs are unchanged when compiled for
      • x86 (PC’s), MC68000 (Older Macs), PPC (Some IBM mainframes), ARM (what’s in your Android phone), Sparc (Sun/Oracle), PDP-11, RS600, and other CPUs

• Why learn it?
  • C is a “low level” language. It’s very flexible
  • Its close relationship to the processor (especially in memory management) enables one to understand and debug higher-level languages like Java/Python
  • If you want to drive fast, you need to understand how the engine is built — C is the engine that underpins many modern languages.
Even Simple things can be measurably faster when coded in C

- HW1 (Reverse and lines of a file using an Array and List implementation)
- Input: 200K lines random-strings.txt from your homework

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>Java</th>
<th>Java vs. C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>0.065s</td>
<td>2.018s</td>
<td>31x Slower</td>
</tr>
<tr>
<td>List</td>
<td>0.079s</td>
<td>1.037s</td>
<td>13X slower</td>
</tr>
</tbody>
</table>

Goal of these lectures is to learn enough C to be able to read and understand the C implementation.
C and Java are related, but they are very different languages:

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object-Oriented</td>
<td>Procedural</td>
</tr>
<tr>
<td>Interpreted</td>
<td>Compiled</td>
</tr>
<tr>
<td>Memory Management</td>
<td>No Memory Management</td>
</tr>
<tr>
<td></td>
<td>(User is responsible)</td>
</tr>
<tr>
<td>References</td>
<td>Pointers</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Error Codes</td>
</tr>
<tr>
<td>--</td>
<td>Pre-processor</td>
</tr>
</tbody>
</table>
Object Oriented vs. Procedural

• A Java Program is a collection of objects
  • Objects contain data and methods
  • Methods operate on their data

• A C program is a collection of procedures
  • Data used by the procedures can be internal to or external to the procedures
  • A notion of objects does not exist in C

• Both Java and C use the special method/function called `main()` to start the program
Interpreted vs. Compiled

- Java is interpreted
  - The java compiler converts source to bytecode
  - The java virtual machine (JVM) interprets bytecode
  - Once a java class is compiled, it (theoretically) does not need recompiling to run on different physical hardware (“compile once, run everywhere”)

- C is compiled
  - The C compiler converts source code directly to machine code
  - The machine code is executed directly by the processor
  - C programs are portable, but must be re-compiled to run on different operating systems/architecture
Memory management vs. No Memory Management

• Java performs most memory management automatically for the user
  • When more memory is required, a user creates a new object instance with the `new` keyword
  • When objects are no longer being used, the garbage collector frees memory for recycling

• C provides no memory management
  • Users create new memory with `malloc`
  • Program logic must determine when it is safe to recycle memory and then call `free`
  • When you don’t properly free no-longer needed memory, you create a `memory leak`
References vs. Pointers

• Java
  • A reference “points to” an Object.
  • Only Objects can be referenced (primitive variables cannot be referenced)

• C
  • A pointer “points to” memory
  • what’s at the end of the pointer depends upon what is stored at that location in memory
  • One can point to primitive variables (e.g. ints), more complex data types (C structures), functions, and pointers.
 Exceptions vs. Error Codes

• Java
  • When an error occurs, an exception is thrown
  • Exceptions can be caught

• C
  • When an error occurs, a function may return an error code (integer)
  • When an error condition goes unnoticed, the program behaves oddly or crashes later
The C preprocessor

• The C preprocessor is a text-to-text conversion process that is integral to C compilation.
• A C source file is preprocessed, then compiled
• A variety of capabilities
  • Define untyped constants (#define) or undefined (#undef)
  • Conditionally compile sections of code
    • #if ... #else ... #endif
    • #ifdef
    • #ifndef
  • Include source code from other files (#include)
  • parameterized macros (#define)
### The C Keyword Universe

<table>
<thead>
<tr>
<th>auto</th>
<th>break</th>
<th>case</th>
<th>char</th>
<th>const</th>
<th>continue</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>do</td>
<td>double</td>
<td>else</td>
<td>enum</td>
<td>extern</td>
</tr>
<tr>
<td>float</td>
<td>for</td>
<td>goto</td>
<td>if</td>
<td>int</td>
<td>long</td>
</tr>
<tr>
<td>register</td>
<td>return</td>
<td>short</td>
<td>signed</td>
<td>sizeof</td>
<td><strong>static</strong></td>
</tr>
<tr>
<td>struct</td>
<td>switch</td>
<td>typedef</td>
<td>union</td>
<td>unsigned</td>
<td><strong>void</strong></td>
</tr>
<tr>
<td>volatile</td>
<td>while</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bold keywords in both C and Java.** **static** has very different meanings
Many Java keywords that are NOT defined in C

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>assert</td>
<td>boolean</td>
<td>byte</td>
</tr>
<tr>
<td>catch</td>
<td>class</td>
<td>extends</td>
<td>finally</td>
</tr>
<tr>
<td>implements</td>
<td>imports</td>
<td>instanceof</td>
<td>interface</td>
</tr>
<tr>
<td>package</td>
<td>private</td>
<td>protected</td>
<td>public</td>
</tr>
<tr>
<td>super</td>
<td>synchronized</td>
<td>this</td>
<td>throw</td>
</tr>
<tr>
<td>throws</td>
<td>transient</td>
<td>try</td>
<td></td>
</tr>
</tbody>
</table>

These concepts are not part of the C language.
A first C program

```
$ cat first.c
#include <stdio.h>
int main(int argc, char *argv[])
{
    float x = 34.5;
    int i = 4;
    printf("x = %f, i = %d, i * x = %f\n", x, i, i * x);
}
$ cc first.c
$ ./a.out
x = 34.500000, i = 4, i * x = 138.000000
```
A first C program

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    float x = 34.5;
    int i = 4;
    printf("x = %f, i = %d, i * x = %f\n", x, i, i * x);
}
```

$ cc first.c

$ ./a.out

x = 34.500000, i = 4, i * x = 138.000000

What is a.out?
A. The compiled first.c program
B. A file that stores the output of running the first.c program
C. A file that stores a class named a
D. I have no idea!
#include <stdio.h>

• This is a preprocessor directive

• include the contents of the file stdio.h
  • often referred to as an “include file”
  • stdio.h, stdlib.h, string.h are common ones

• `<stdio.h>` -- indicates to the compiler, look in system areas for the file

• man stdio.h to see details.