Announcements

- E2 has been graded
- A3 due Thursday (early deadline tonight)
- A4 draft written
  - working on autograder next
  - a fair bit lighter than last time, I think
  - recommendation: use the extra time to test and refactor!
E2 solution
review
What is coupling?

Learning objective: *define coupling*.

- **coupling (n)**: a British TV show from the early 2000s.

Source: IMDB
What is coupling?

Learning objective: *define coupling*.

- **coupling (n)**: a device that serves to connect the ends of adjacent parts or objects.

Sources:

What is coupling?

Learning objective: *define coupling in a software context.*

- **coupling (n):** when two units of code are linked together.
Coupling example 1

Learning objective: *given example code, identify a point of coupling*

```clojure
(defn print-table-row [csv-line]
  (print "<tr>")
  (doseq [field (str/split csv-line ",")]
    (print "<td>")
    (print field)
    (print "</td>"))
  (print "</tr>"))
```

This couples:
- the formatting of a table row
- the effect of output
Coupling example 2

Learning objective: *given example code, identify a point of coupling*

```
(defn write-blob [file-path]
  (let [data (slurp file-path)
         header+data (format "blob %d\000%s" (count string) string)
         addr (sha/sha1-sum header+blob)
         first2 (subs addr 0 2)
         last38 (subs addr 2)
         obj-path (format ".git/objects/%s/%s"
                      first2 last38)]
    (io/make-parents (io/file obj-path))
    (io/copy (zip/zip-str header+data)
             (io/file obj-path))
    (println addr)))
```
Coupling example 2

(defn write-blob [file-path]
  (let [data (slurp file-path)
         header+data (format "blob %c
         addr (sha/sha1-sum header+bl
         first2 (subs addr 0 2)
         last38 (subs addr 2)
         obj-path (format ".git/obj
         first2 last
         (io/make-parents (io/file obj-pa
         (io/copy (zip/zip-str header+da
         (io/file obj-path))
     (println addr)))))

   • the effect of slurping the file
   • the logic of where the object lives
   • the logic of computing the object header
   • the effect of writing the object
Indirection

Learning objective: *define indirection*.

- **indirection (n)**: the ability to reference something using a name, reference, or container instead of the value itself (source: [Wikipedia](https://en.wikipedia.org/wiki/Indirection))

**Discussion**

- Coupling involves *direct access*
- Introducing a mediator between the two pieces makes the access *indirect*
- So indirection is the solution to coupling (and vice-versa)
Adding indirection

Learning objective: *given example code, add indirection to remove a point of coupling*

```clojure
(defn add-5-to-each [xs]
  (map #(+ 5 %) xs))
```

- In this example, 5 is hardcoded; that is, the function is coupled to the value 5.
- Let's add indirection to make this flexible.

```clojure
(defn add-to-each [n xs]
  (map #(+ n %) xs))
```
Adding indirection - discussion

- We added a parameter to give flexibility
- This is one way to add indirection
- Another is to extract a function
- Think of indirection as using an elbow instead of a coupling: it provides a dimension of flexibility
Tradeoffs of coupling and indirection

Learning objective: *discuss the tradeoffs of coupling vs. indirection*

- The benefit of coupling is that, when flexibility is not needed, it keeps things simple
  - Adding flexibility where it's not needed is just adding complexity
  - Remember YAGNI: you ain't gonna need it
- The benefit of indirection is that, when flexibility is needed, it is a simple way to provide it

**Recommendation:** Don't add indirection until you have a clear reason that it's needed.