To Dos for Next Code Review

- For next code review: GoFish++
  - Build on your existing GoFish code
- Internationalize (as discussed on Tuesday)
- Implement (at least) 1 more smarter strategy
  - Strategy used by each player set when program run
    - E.g., command line: GoFish stupid stupid smart smart
      - Four player game w/players 0 & 1 = stupid, 2 & 3 = smart
  - Smarter strategy uses other player’s previous actions
    - If A asked B for 7s, and I have a 7, I should ask A for 7s.
    - You can decide exact behavior, but it should significantly outperform ‘stupid’
Game
run game C

Target
all other players

Smart Player C
Player 1

Stupid Player

Broadcast & locally model
Log

player

game

append Entry
get length
get (i)
Android Pre-requisites
XML (eXtensible Markup Language)

- For “marking up” data so it can be processed by computers
  - Much like JSON in purpose

```xml
<?xml version="1.0"?>
<weatherReport>
  <date>7/14/97</date>
  <city>North Place</city>, <state>NX</state>
  <country>USA</country>
  High Temp: <high scale="F">103</high>
  Low Temp: <low scale="F">70</low>
  Morning: <morning>Partly cloudy, Hazy</morning>
  Afternoon: <afternoon>Sunny & hot</afternoon>
  Evening: <evening>Clear and Cooler</evening>
</weatherReport>
```
XML vs. HTML

- HTML and XML look similar, because they are both SGML languages (SGML = Standard Generalized Markup Language)
  - Both HTML and XML use elements enclosed in tags (e.g. `<body>This is an element</body>`)  
  - Both use tag attributes (e.g., `<font face="Verdana" size="+1" color="red">`)  
  - Both use entities (\&lt;, \&gt;, \&amp;, \&quot;, \&apos;)

- More precisely:
  - HTML is defined in SGML; XML is a (small) subset of SGML

- Differences:
  - XML describes content; HTML describes structure & presentation  
  - HTML has fixed set of tags; XML you define your own tags
XML Structure

- An XML document may start with one or more processing instructions (PIs) or directives:
  ```xml
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/css" href="ss.css"?>
  ```

- Following the directives, there must be exactly one root element containing all the rest of the XML:
  ```xml
  <weatherReport>
    ...
  </weatherReport>
  ```
XML building blocks

- Aside from the directives, an XML document is built from:
  - tags, in pairs: `<high scale="F">103</high>`
  - elements: `high in <high scale="F">103</high>`
  - attributes: `<high scale="F">103</high>`
  - entities: `<afternoon>Sunny & hot</afternoon>`
  - character data, which may be:
    - parsed (processed as XML) -- this is the default
    - unparsed (all characters stand for themselves)
  - comments: `<!-- anything & here is comment -->`
Elements and attributes

- Attributes and elements are somewhat interchangeable
- Example using just elements:
  ```xml
  <name>
    <first>David</first>
    <last>Matuszek</last>
  </name>
  ```
- Example using attributes:
  ```xml
  <name first="David" last="Matuszek"/>
  ```
- You will find that elements are easier to use in your programs -- this is a good reason to prefer them
- Attributes often contain metadata, such as unique IDs
- Generally speaking, browsers display only elements (values enclosed by tags), not tags and attributes
Well-formed XML

- Every element must have both a start tag and an end tag, e.g. `<name> ... </name>`
  - But empty elements can be abbreviated: `<break />`.
  - XML tags are case sensitive
  - XML tags may not begin with the letters `xml`, in any combination of cases
- Elements must be properly nested, e.g. not `<b><i>bold and italic</b></i>`
- Every XML document must have one and only one root element
- The values of attributes must be enclosed in single or double quotes, e.g. `<time unit="days">`
- Character data cannot contain `<` or `&`
Entities

- Five special characters must be written as entities:
  - `&amp;` for `&` (almost always necessary)
  - `&lt;` for `<` (almost always necessary)
  - `&gt;` for `>` (not usually necessary)
  - `&quot;` for `"` (necessary inside double quotes)
  - `&apos;` for `'` (necessary inside single quotes)
- These entities can be used even in places where they are not absolutely required
- These are the only predefined entities in XML
XML declaration

The XML declaration looks like this:
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
- The XML declaration is not required by browsers, but is required by most XML processors (so include it!)
- If present, the XML declaration must be first---not even whitespace should precede it
- Note that the brackets are `<?` and `?>`
- `version="1.0"` is required (this is the only version so far)
- `encoding` can be "UTF-8" (ASCII) or "UTF-16" (Unicode), or something else, or it can be omitted
- `standalone` tells whether there is a separate DTD
Names in XML

- Names (as used for tags and attributes) must begin with a letter or underscore, and can consist of:
  - Letters, both Roman (English) and foreign
  - Digits, both Roman and foreign
    - (dot)
    - (hyphen)
    - (underscore)
  - (colon) should be used only for namespaces
  - Combining characters and extenders (not used in English)
Comments

- <!-- This is a comment in both HTML and XML -->
- Comments can be put anywhere in an XML document
- Comments are useful for:
  - Explaining the structure of an XML document
  - Commenting out parts of the XML during development and testing
- Comments are not elements and do not have an end tag
- The blanks after <!-- and before --> are optional
- The character sequence -- cannot occur in the comment
- The closing bracket must be -->
- Comments are not displayed by browsers, but can be seen by anyone who looks at the source code
You can make up your own XML tags and attributes, but...
  ...any program that uses the XML must know what to expect!

A DTD (Document Type Definition) defines what tags are legal and where they can occur in the XML

An XML document does not require a DTD

XML is well-structured if it follows the rules given earlier

In addition, XML is valid if it declares a DTD and conforms to that DTD

A DTD can be included in the XML, but is typically a separate document

Errors in XML documents will stop XML programs

Some alternatives to DTDs are XML Schemas and RELAX NG
Review of XML rules

- Start with `<?xml version="1"?>`
- XML is case sensitive
- You must have exactly one root element that encloses all the rest of the XML
- Every element must have a closing tag
- Elements must be properly nested
- Attribute values must be enclosed in double or single quotation marks
- There are only five predeclared entities
GUI terminology

- **window**: A first-class citizen of the GUI.
  - Also called a *top-level container*.

- **component**: A GUI widget that resides in a window.
  - Called a View in Android
  - examples: Button, CheckBox, TextView

- **container**: A logical grouping for storing components.
  - examples: LinearLayout, ListView,
GUI interface example
android.view.View

Breakdown of a Layout

Lifestyle
Etsy, Pinterest, LifeHacker, Fast Co.Design, Maxim, Elle

Linear layout
icon
icon

Linear layout

Linear layout

icon

Linear layout

text view 1

Linear layout

text view 2
Design Pattern: Composite

- Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
Design Pattern: Composite (cont.)

- Client doesn’t need to know whether an object is a leaf or a composite