1) **Who** cares about the quality of an interface?

2) How can we **analyze the quality of a class's interface**?
2 Points Of View

- Can view a class interface from 2 points of view:
  1..
    - Goals:
      - Easy to understand, clear abstraction
      - Easy to use
  2..
    - Goals:
      - Easy to design
      - Easy to implement
Interface Design Challenge

- **Challenge**
  The easiest way to implement a feature may not be...

- **Example**
  - Getting MP3 song's info:

    Option 1:
    ```
    /**
     * Pass the ID number:
     * 1 = artist
     * 2 = song title
     * 3 = recording year
     * ...
     */
    String getSongInfo(int id);
    ```

    Option 2:
    ```java
    String getArtist();
    String getSongTitle();
    int getYearRecorded();
    ```
Interface Quality

• Analyze the interface checking for:

  1. Cohesion
  2. Completeness / Convenience
  3. Clarity
  4. Consistency
Cohesion

- **Cohesion**
  - Are all interface methods..

- **Single Responsibility Principle:**
  - A class should have..
  - i.e., all its code should deal with one responsibility.

- **Example:**
  - All relates to a "game"; cohesion?
  - each handling one responsibility
Completeness & Convenience

- **Completeness / Convenience**
  - Interface should have the...

- **Example: Reading a line from System.in**
  BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
  String line1 = reader.readLine();

  Scanner scanner = new Scanner(System.in);
  String line2 = scanner.nextLine();

- **DNA Example:**
  - DNA made up of G, A, T, and C nucleotides.
  - Missing..
    Client could write it, but class incomplete!

Clarity

- The interface should be clear to the programmer.
- Use well named classes, methods and variables to be..
- Use..

**Example:** Compare these Stack methods
- `getTop()`, `setTop()`
- `push()`, `pop()`

**Example:** Consider these ListIterator methods
- `next()`, `hasNext()`, `previous()`, `hasPrevious()`, `add()`, `remove()`
- Which element does..
• Consistency:
  
  ```java
public class GameBoard {
    // row: 0-indexed row.
    // col: 1-indexed column.
    Piece getPiece(int row, int col) { ... }

    void setPieceOnBoard(int col, int row, Piece element) { ... }

    boolean positionHasPiece(int x, int y) { ... }
}
```

• Consistency Problems:
  
  ```
  0 indexed for Java
  ```

  ```
  (row, col) vs (col, row)
  ```
Additional Class/Interface Quality Checks

• **4C’s**
  - Cohesion
  - Completeness
  - Clarity
  - Consistency

• **Some other ways to review quality**
  - Constructor create fully formed objects
  - One name for each idea
  - Command-query
  - Implementing Iterable/Comparable/... when appropriate
  - Breaking encapsulation
• Analyze the quality of the following interface:

```java
/**
 * Represent a point in 2D space.
 */
interface Point2D {
    void setLocation(int x, int y);
    void setHeight(int height);

    int getX();
    int getYValue();

    double getDistanceTo(int y, int x);

    void drawStarAtPoint();
    void drawCircleAtPoint(int radius);
    double computeTriangle(Point2D p1, Point2D p2);
}
```
Summary: “4C's” Analysis Process

1. Check..
   - Interface relate to a single abstraction?
   - If not, split into multiple classes.

2. Check..
   - All required methods provided?
   - Client code have functions which should be in the class?

3. Check..
   - All classes, methods, variables have the best names?
   - Is the abstraction clear?

4. Check..
   - All names, numbering, and ordering consistent?

- Goals often conflict; strike the best balance you can.