Topic 4: Guidelines for Class Design

Part 3: Interface Quality (Ch. 3.5)

Interface Quality

Interface

Points of View

- Can view a class interface from 2 points of view:
  1. Class's User / Client
  2. Class's Designer / Programmer

Interface (2)

Points of View

- Challenge
  The easiest way to implement a feature may not be the easiest way to understand it (and vice versa)

- Illustration: Getting info from Person class:

```
/**
 * Pass the number:
 * 1 = name
 * 2 = gpa
 * 3 = birthday
 * ...
 */
String getPersonInfo(int id){
   ...
}
```

```
String getName(){...}
String getGPA(){...}
Date getBirthday (){...}
```

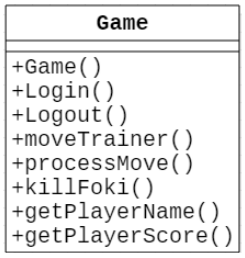
Interface Quality

- Analyze the interface checking for:

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

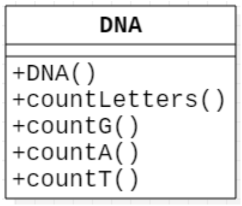
Cohesion

- **Cohesion**: Are all interface methods related to a single abstraction?
- Single Responsibility Principle: A class should have only one reason to change.
  - All of its code should deal with one responsibility.
  - Example:



Completeness & Convenience

- **Completeness** - Interface should have the features client code needs
- DNA Example: DNA made up of G, A, T, and C nucleotides.
  - It's missing `countC()` method – incomplete



```
int numC = myDna.countLetters()
                - myDna. countLettersG()
                - myDna.countLettersA()
                - myDna.countLettersT();
```

- Convenience – simple tasks should be simple
- Example: Reading input 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();
```

Clarity

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

- Example: Compare these Stack methods
  - `getTop(), setTop()`
  - `push(), pop()`
- Example: Consider these ListIterator methods
  - `next(), hasNext(), previous(), hasPrevious(), add(), remove()`
- (IteratorClarity.java)

Consistency

- **Consistency** - operations in a class should be consistent with each other with respect to names, parameters and return values, and behavior. Pertains to things like:
  - Indices
  - Naming conventions
  - Argument order
  - etc …

- (Consistency.java)

Interface Quality Checks

- Other ways to check Interface Quality
  - Constructor create fully formed objects
  - One name for each idea
  - Command-query
  - Not implementing Iterable when appropriate
  - Breaking encapsulation

Quick Review Exercise

```
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);
}
```