Lecture Set #13: Args Array and Commenting Review

1. Arguments to Main Method
2. Commenting Review

Arguments to main

- Recall prototype of main method
  public static void main (String[] args);
  - args is array of Strings
  - args come from operating system
    - When user runs executable …
    - … s/he can provide arguments
- Demonstrations of Using the Args array
Types of Documentation

- Two kinds of code commentary in Java
  - Implementation comments
  - Interface comments
- JavaDoc
  - from inside Eclipse
  - from outside of Eclipse
- Common Errors in Commenting
  
  Too many comments: This can obscure the flow of your program.
  Too few comments: Your intent may not be understood (code is never “self-documenting”!)

Comments that repeat the code:
  ```
  int total = 0;  // initialize integer total to 0
  ```

Using comments to conceal unclear code:
  ```
  double a = h*w;  // set the area (a) to the height (h)
  ```

Uninformative comments: “What the heck does that mean?”
  ```
  double d = processValue();  // Change later (legacy)
  ```

Misleading / erroneous comment: These are dangerous
  ```
  for (int i = 0; i < a.length-1; i++)  // run through the whole array
  ```

Javadoc Documentation

- **Class comments**: Immediately prior to each public class, add a javadoc comment that explains what the class does. You can also add the following special “tags”, which javadoc recognizes and provides special formatting for:
  ```
  @author – the author of the class
  @version – the current software version number
  @see – refer the reader to related classes
  ```

- **Example**: In Rational.java
  ```
  /**
   * This class implements a rational number object,
   * and provides methods for performing arithmetic
   * on rational numbers.
   * @see java.lang.Math
   * @author Schultzie von Wienerschnitzel III
   * @version 3.14159
   */
  public class Rational { … }
  ```
Javadoc Documentation

- **Method comments**: Immediately prior to each public method, add a javadoc comment explaining what the method does, the meanings of the parameters, the return value, and any errors. The following tags are recognized:
  - `@param` – give the name and description of each parameter. There should be one for each parameter.
  - `@return` – describe the return value (unless it is void)
  - `@throws` – (exceptions that could propagate out from this method)
  - `@deprecated` – *(Usually for system use: indicates that a method should be avoided, since better alternatives exist)*

- **Example**: 

```java
/**
 * Multiplies two rational numbers and returns their the product.
 * @param q The first operand.
 * @param r The second operand.
 * @return A reference to a newly created Rational with the sum.
 */

public static Rational multiply(Rational q, Rational r) { … }
```