Intro to Javadoc

Documentation with Javadoc

Documenting your code is crucial to help others understand it, and even to remind yourself how your own older programs work. Unfortunately, it is easy for most external documentation to become out of date as a program changes. For this reason, it is useful to write documentation as comments *in the code itself*, where it can be easily updated with other changes. **Javadoc** is a documentation tool which defines a standard format for such comments, and which can generate HTML files to view the documentation from a web browser. As an example, see Oracle's Javadoc documentation for the Java 8 libraries at https://docs.oracle.com/javase/8/docs/api/. Click through and explore for five minutes to get familiar with the formatting and you'll see below how to generate documentation like this for your own code!

Javadoc in a way “compiles” your documentation from special comments you put above package, class, method and variable declarations, then generates HTML that you and others can use to browse the classes you've created. You can easily run Javadoc on your project from within BlueJ, by selecting Tools → Project Documentation (if it asks you if you want to regenerate documentation, go ahead and select that, as it'll be as up to date as possible!). Note that if you look inside your project directory, you'll see that it created/updated a “doc” directory that contains all of the HTML files for the classes in your project. Recall that the completed classes we've given you as part of the project have javadoc that you can find in the KivaWorld Workdocs folder.

Writing Javadoc Comments

In general, Javadoc comments are any multi-line comments that begin with `/**` instead of the usual `/*` ("`/** ... */`") that are placed before class, field, or method declarations. They *must* begin with a slash and two stars, and they include special tags (starting with `@`) to describe characteristics like method parameters or return values. The HTML files generated by Javadoc describe each field and method of a class, using the Javadoc comments in the source code itself.

Simple Comments. Normal Javadoc comments can be placed before any class, field, or method declaration to describe its intent or characteristics. For example, the Javadoc in the following simple `Student` class documents the class, the `name` field, and the `Student` constructor.

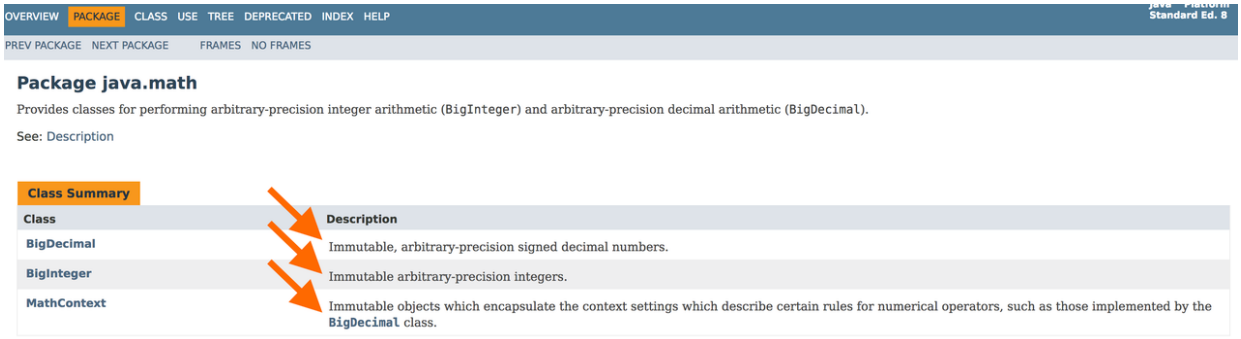
```
/**
 * Represents a student enrolled in the school.
 * A student can be enrolled in many courses.
 */
public class Student {

  /**
   * The first and last name of this student.
   */
  private String name;

  /**
   * Creates a new Student with the given name.
   * The name should include both first and
   * last name.
   */
  public Student(String name) {
    this.name = name;
  }
```

```
}
```

All documentation must start off with a single summarizing sentence that ends with a period. These are displayed in the javadoc HTML where a compact listing of packages, methods or fields are given–e.g. for developers to determine whether to click to view more. For example, here is the listing of classes in the package, java.math, showing one-line descriptions of the classes in a table:



Javadoc Tags: @param, @return. Tags can be used at the end of each Javadoc comment to provide more structured information about the code being described. For example, most Javadoc comments for methods include `@param` and `@return` tags when applicable, to describe the method's parameters and return value. The `@param` tag should be followed by the parameter's name, and then a description of that parameter. The `@return` tag is followed simply by a description of the return value. Examples of these tags are given below.

```
/**
 * Gets the first and last name of this Student.
 * @return this Student's name.
 */
public String getName() {
  return name;
}

/**
 * Changes the name of this Student.
 * This may involve a lengthy legal process.
 * @param newName This Student's new name.
 *                Should include both first
 *                and last name.
 */
public void setName(String newName) {
  name = newName;
}
```

The indentation convention is to indent subsequent lines of the same comment to line up with the left side of the beginning of the comment (note that the name of the parameter is not considered part of the “comment” so all the lines align with the left side of `This Student's...` rather than with `newName`

Other common tags: @see. You can also include `@see #setName` (to provide a link to a field or method named `setName` within the same class) or `@see DataInput`, to refer the reader to a related class or Interface (in this case, the `DataInput` interface).

Further reference (Optional). A good table showing the breadth of tags can be found at https://www.tutorialspoint.com/java/java_documentation.htm .