JavaDoc

Javadoc is the JDK tool for producing professional documentation. Javadoc can document the source code, and this documentation can be enhanced through the use of structured comments. Typical examples of Javadoc comments include

- Descriptions of classes
- Version numbering and author attribution
- Descriptions of method signatures and return types
- Details about instance variables, class variables and constants

Javadoc comments start with /** and end with */. The leading /** distinguishes a Javadoc comment from a regular multi-line (C-style) comment. Javadoc comments are placed immediately before the item they describe. Javadoc tags which start with an "@" identify elements that will be processed by the Javadoc doclet engine. A comprehensive list is shown at the end of this document (from https://www.tutorialspoint.com/java/java_documentation.htm).

Adding JavaDoc to a Class

Here is an example of a set of requirements for a class.

Write an immutable class Point that describes a point on a plane. In mathematics, a plane is a flat, two-dimensional surface that extends infinitely far. Provide a constructor to create a Point object at a specified point, also the no-argument constructor should create a Point object at the origin (0,0). Declare methods getX(), and getY() that return the x and y coordinates of the point. Declare methods translate() and scale(). The translate() method moves the point by a specified amount in the x- and y-directions. The scale() method scales the coordinates (x and y) by the same amount. Implement these methods so that they return new points with the results.

Example use

```
Point p = new Point(3, 4).translate(1, 3).scale(0.5);
```

The above will cause p to identify a Point object at (2, 3.5)

Description based on Core Java for the Impatient by Cay Horstmann

To Do

In this practical we will we implement and document this class.

Write the class with skeleton (empty) implementations only. Remember Javadoc processes the declarations only, so documentation can begin before implementation.

Think about what is included in the requirements and what is not. For example, a plane is infinite, is this a practical consideration for the implementation. The requirements describe that translate “moves the point by a specified amount”, however this is inconsistent with the Point being immutable.

@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a **Throws** subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the nearest inheritable class or implementable interface.	Inherits a comment from the immediate superclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced class.	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serialisable field.	@serial field-description | include | exclude

@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods.	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant.	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text