TL;DR Java Annotations

Objectives

- Define annotations
- Write annotation syntax
- Understand where annotations go
- Use built in Java annotations

What Are They

- Are a form of metadata for Java code
- Dont modify the execution of code
- 2 main groups: Compile-time & Run-time (we are learning compile-time)

Why do we use Java Annotations

- Information for the compiler
  - To detect errors
  - Suppress Warnings

- Compile-time and deployment-time processing
  - Can provide information to generate code, XML files, etc

Annotation Syntax

```
//An example:

@Entity

// @ = annotation
// Entity = The name of the annotation


//Another Example:

@Entity(table = "users")
```

When to use them

They are usually used for before:

- Classes
- Interfaces
- Fields
- Local Variables

Built in Java Annotations

These are compile-time annotations which are used to give the compiler instructions.

- `@Deprecated` - When something should no longer be used. Should include Javadoc comment.

- `@Override` - Indicates that a method overrides a method in a superclass. Not necessary to override, but considered best practice.

- `@SuppressWarnings` - Used to suppress warnings from the compiler. Use intentionally and document the reason. Most common values are `all`, `unchecked`, and `deprecation`

What you should know at this point

- What an is an annotation
- What are the most common types of annotations
- The parts and placement of annotation
- How to use `@Deprecated`, `@Override`, and `@SuppressWarnings`

Additional Resources

- Oracle Annotation Tutorial
- List of SuppressWarnings
- Java Revisted Blog - SuppressWarnings
- Blog BangBits - SuppressWarnings