Internal and External Documentation

Internal source-code documentation explains a class, method, or a piece of code.

External documentation is at a higher level and explains e.g. problem definition, requirements, architecture, design, project plans, test plans.

Internal Documentation

Internal documentation should explain things that aren't represented in code (and ideally anything that can be represented in code should be) – you should comment the code intent, rather than implementation details.

Self-documenting code uses a good programming style to make the intent of the code clear (e.g. good names, simple logic, good formatting, consistency).

Questions About the Code

What is the code supposed to do?

- should be answered by comments and method signatures

How does it do it?

- answered by the implementation – comments and code

Annotations

These provide information about a program that's not part of the program itself.

They can be used to catch errors (e.g. `@Override`), suppress warnings, generate code, or augment program behaviour.

Annotations affect the next non-annotation element in the code.

`@Retention` can be used to specify how long an annotation should be retained (e.g. just until compile-time, or until run-time). There are three options:

- `RUNTIME`
- `CLASS`
- `SOURCE`

Create Your Own

Can create your own annotations:

```java
@interface Hints {
    Hint[] value();
}

@interface Hint {
    String value();
}
```

Java 8 also allows you to use multiple annotations of the same type, by declaring it with the "Repeatable" annotation.

Annotations can replace comments in code, and defining an annotation type allows you to create templates:

```java
@interface ClassPreamble {
    String author();
    String date();
    int currentRevision() default 1;
    String[] reviewers();
}

@ClassPreamble (
    author = "John Doe",
    […]
)
```

It's more structured and consistent than comments. We can also add them to documentation by putting the `@Documented` annotation.

In Java 8

Annotations can now be used anywhere you use a type. Type annotations improve analysis of programs and ensure stronger type checking.

Javadoc

This is a tool which parses the declarations and documentation comments in java source files.

Limitation

Javadoc can't do incremental builds, and produces one complete document each time it's run.

Workings

Javadoc uses the java compiler to compiler the declarations, but ignores the implementation.

Javadoc includes user documentation from special comments in the source code.

You can actually start documenting before implementing things – starting in the earliest stages of design.

Javadoc tool will also document implicit code, e.g. default constructors.