Documentation

Technical documentation in Markdown & Mermaid.

Writing code isn’t enough

- We want to build maintainable software, that can be modified, update and remain useful over a long period of time.
    - This is the motivation for modular, flexible software design.
- However, writing well-structured code isn’t enough!
    - The person writing the code may not be the person maintaining it over time.
    - Even if you are maintaining code that you wrote, you will not remember your reasons for the design decisions you made 6+ months afterwards.
    - Code isn’t accessible to everyone in your organization! What about sales? Marketing?

Generating and maintaining documentation is essential to long-term success.
Effective and complete documentation is critical for the *communication of complex ideas*.

What is documentation?

There are many forms of documentation that we care about:

Project documentation

- Tracking project details to help us remember our project constraints. Useful for planning later phases. e.g., Issues lists; Milestones; Project plans; Gantt charts.

Design documentation

- Why we made specific design decisions; materials to help new developers understand rationale. e.g., UML diagrams; design documents.

Code documentation

- Inline documentation (code comments) to explain peculiarities of an implementation.

User documentation

- Help users understand how something works! e.g., how to install; what features exist; what has changed in a new release.

Docs as Code

Documentation as Code (*Docs as Code*) is the philosophy that you should be writing documentation with the same tools you use to author and maintain code.

- Issue tracking – use this to track doc changes.
- Version control (Git) – version your docs with your code.
- Prefer plain text documents – you can diff them, use Git.
- Code reviews – docs should be included in feature reviews.
- Automated tests – unit test your docs!

“This [also] means following the same workflows as development teams and being integrated in the product team. It enables a culture where writers and developers both feel [collective] ownership of documentation and work together to make it as good as possible.”

Docs as code does NOT mean that “your code is the documentation”.

WRITE THE DOCS

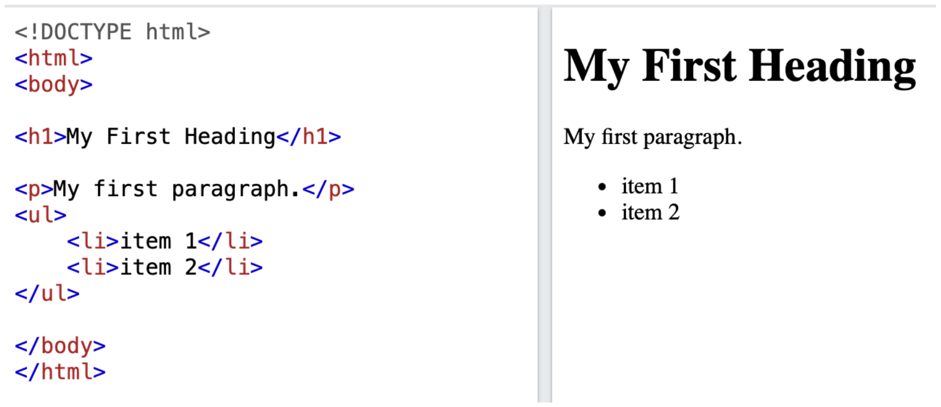
www.writethedocs.org

Markup Languages (Markdown)

Authoring documents using plain text.

What is a markup language?

A markup language is a system of annotating a document to describe its structure and presentation. It uses tags or codes to define elements such as headings, paragraphs, lists, images, links, and more. Examples include HTML, AsciiDoc, reStructuredText and Markdown.



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>
<ul>
    <li>item 1</li>
    <li>item 2</li>
</ul>

</body>
</html>
```

My First Heading

My first paragraph.

- item 1
- item 2

www.w3schools.com

What is Markdown?

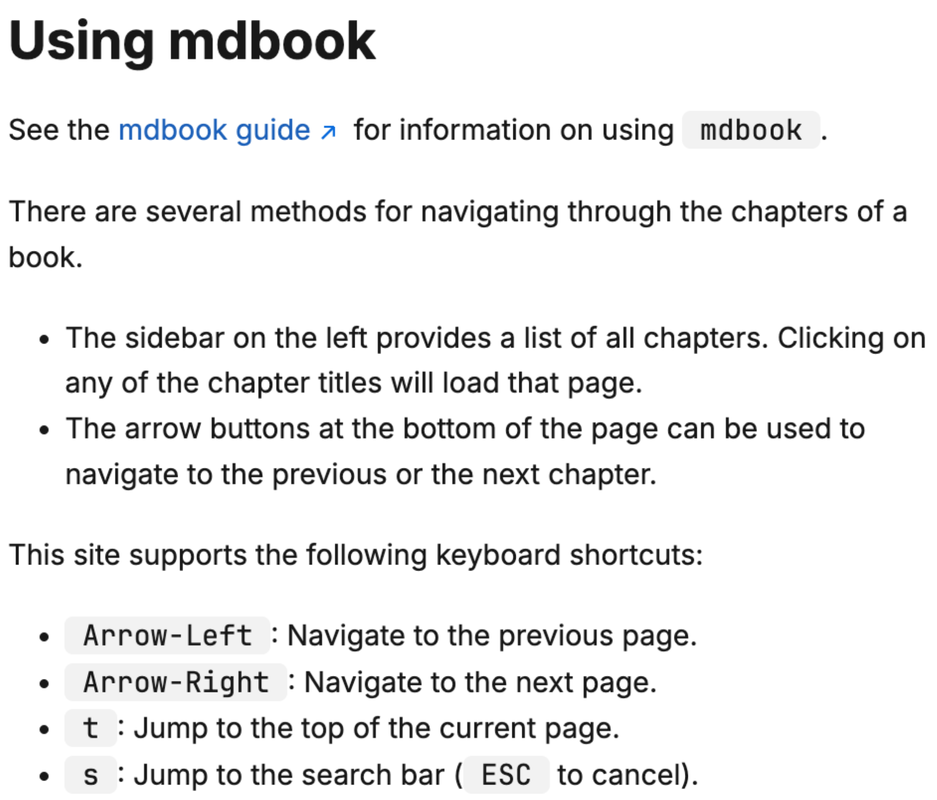
Markdown is a simple markup language that allows you to add formatting elements to a text file. Markdown was designed with a focus on generating HTML (see this blog post from 2004).

In its original form, Markdown is both:

- A formatting specification, and
- A tool for converting markdown files to HTML for publication.

In recent years, Markdown has become the *defacto* standard for technical documentation. It is less complete than other markup languages (e.g., AsciiDoc) but is simpler to use.

```
## Using mdbook
See the [mdbook guide](https://rust-lang.github.io/mdBook/for_developers/index.html) for information on using `mdbook`.

There are several methods for navigating through the chapters of a book.

* The sidebar on the left provides a list of all chapters. Clicking on any of the chapter titles will load that page.
* The arrow buttons at the bottom of the page can be used to navigate to the previous or the next chapter.

This site supports the following keyboard shortcuts:

* `Arrow-Left`: Navigate to the previous page.
* `Arrow-Right`: Navigate to the next page.
* `t`: Jump to the top of the current page.
* `s`: Jump to the search bar (`ESC` to cancel).
```



Using mdbook

See the mdbook guide ↗ for information on using `mdbook`.

There are several methods for navigating through the chapters of a book.

- The sidebar on the left provides a list of all chapters. Clicking on any of the chapter titles will load that page.
- The arrow buttons at the bottom of the page can be used to navigate to the previous or the next chapter.

This site supports the following keyboard shortcuts:

- `Arrow-Left` : Navigate to the previous page.
- `Arrow-Right` : Navigate to the next page.
- `t` : Jump to the top of the current page.
- `s` : Jump to the search bar (`ESC` to cancel).

The course website is generated from Markdown! It's also used for documentation on GitLab, GitHub etc.

Basic Syntax

Symbol	Meaning
#	Heading 1
##	Heading 2
###	Heading 3
text	*Emphasis*
text	*Emphasis alt.*
text	**Embolden**
* item	Bulleted list
1. item	Numbered list
(title)[URL]	Link to a URL
!(title)[URL]	Embed an image

Why would we use Markdown?

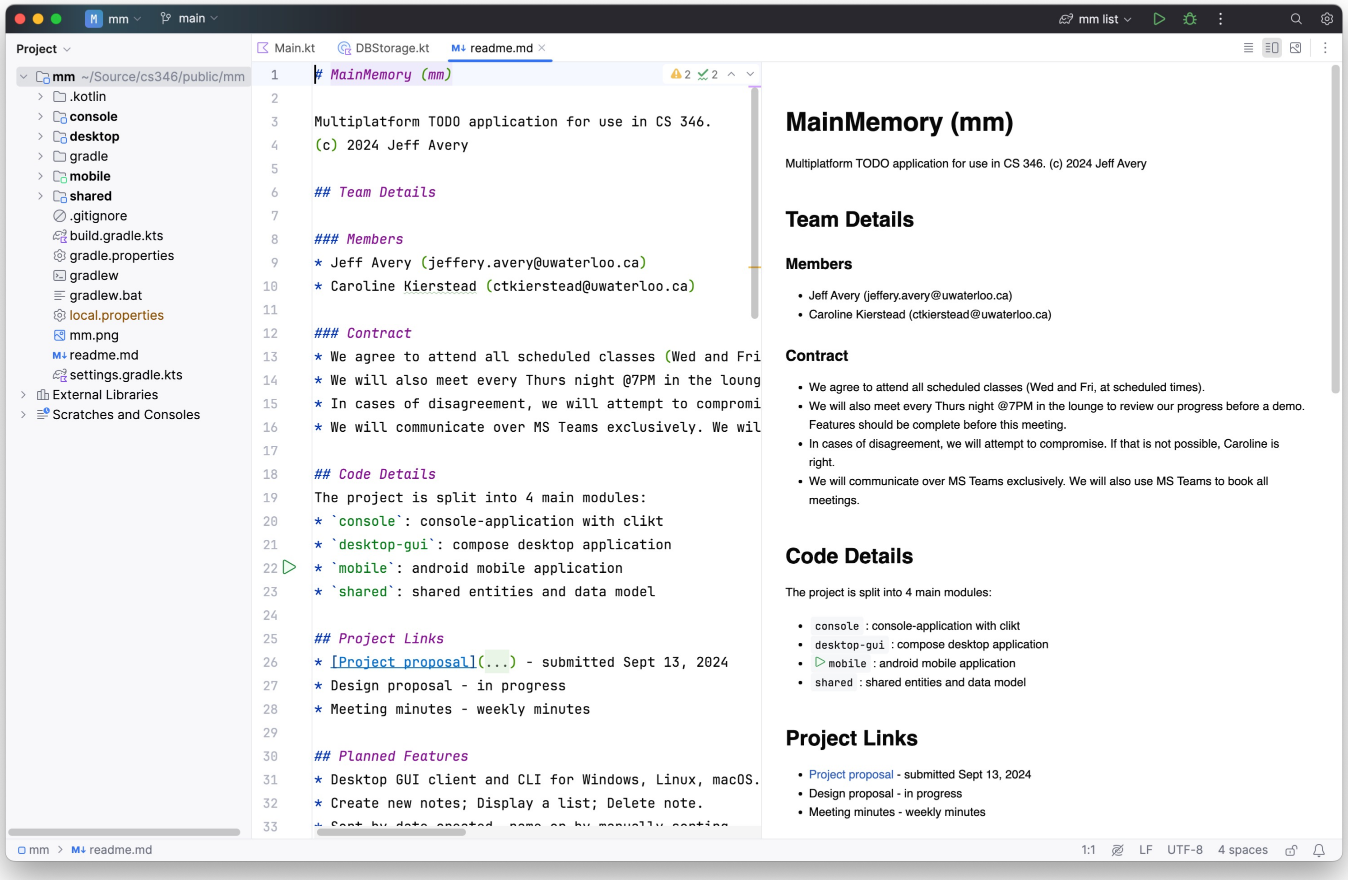
- You can write documentation in any text editor.
- Text, so you can version control it, diff it etc.
- VS Code, most IDEs, GitHub, GitLab support it.
- Defacto standard for software development.

Why not use Markdown?

- There is no standard specification (GitHub and a few organizations have produced extensions).
- Missing support for important features:
    - Footnotes
    - References
    - Floating images
    - Columns
- Works best at generating simple-HTML docs.

How do I use it?

- Editing Markdown
  - VS code, IntelliJ IDEA and most editors have support for Markdown.
- Integrating into your code/documents:
  - Online sites like GitLab, GitHub have built-in support i.e., you can enter text as markdown, and it will be shown “pretty-printed” when possible.
  - You can even embed diagrams into MD in your code projects!
- Generating HTML?
  - Tools like `pandoc` and `Marked` can convert markdown to HTML.
  - Static site generators: Jekyll, Hugo, Retype all generate websites from markdown.



```
# MainMemory (mm)

Multiplatform TODO application for use in CS 346.
(c) 2024 Jeff Avery

## Team Details

### Members
* Jeff Avery (jeffery.avery@uwaterloo.ca)
* Caroline Kierstead (ctkierstead@uwaterloo.ca)

### Contract
* We agree to attend all scheduled classes (Wed and Fri
* We will also meet every Thurs night @7PM in the loung
* In cases of disagreement, we will attempt to compromi
* We will communicate over MS Teams exclusively. We wil

## Code Details
The project is split into 4 main modules:
* `console`: console-application with clikt
* `desktop-gui`: compose desktop application
* `mobile`: android mobile application
* `shared`: shared entities and data model

## Project Links
* [Project proposal](...) - submitted Sept 13, 2024
* Design proposal - in progress
* Meeting minutes - weekly minutes

## Planned Features
* Desktop GUI client and CLI for Windows, Linux, macOS.
* Create new notes; Display a list; Delete note.
```

MainMemory (mm)

Multiplatform TODO application for use in CS 346. (c) 2024 Jeff Avery

Team Details

Members

- Jeff Avery (jeffery.avery@uwaterloo.ca)
- Caroline Kierstead (ctkierstead@uwaterloo.ca)

Contract

- We agree to attend all scheduled classes (Wed and Fri, at scheduled times).
- We will also meet every Thurs night @7PM in the lounge to review our progress before a demo. Features should be complete before this meeting.
- In cases of disagreement, we will attempt to compromise. If that is not possible, Caroline is right.
- We will communicate over MS Teams exclusively. We will also use MS Teams to book all meetings.

Code Details

The project is split into 4 main modules:

- `console` : console-application with clikt
- `desktop-gui` : compose desktop application
- `mobile` : android mobile application
- `shared` : shared entities and data model

Project Links

- Project proposal - submitted Sept 13, 2024
- Design proposal - in progress
- Meeting minutes - weekly minutes

Most development tools will work with Markdown. IntelliJ IDEA for example has support for Markdown syntax and will even pretty-print the output.

Tools > Mermaid.js

How do we generate diagrams-as-code?

Diagramming

- Documentation requires diagrams.
- We can imagine adding many different types of diagrams and charts to our documentation, including:
    - Gantt charts to project management documents.
    - Timeline charts to show milestones and your delivery schedule.
    - UML diagrams for design, and to document implementation details.
        - Component diagrams, class diagrams, sequence diagrams, state diagrams…
    - Flowcharts, and requirements diagrams to explain features to customers.
    - Pie charts to show results.

Diagramming Tools

There are many types of diagramming tools:

1. Pixel manipulation tools

- Produce image formats e.g., PNG. Poor for diagrams; large files, don't scale well.

2. Vector drawing tools

- Produce SVG files or a similar format, which you can embed as images.
- Very precise; complete control over the results!
- e.g., Affinity Designer, Adobe Illustrator

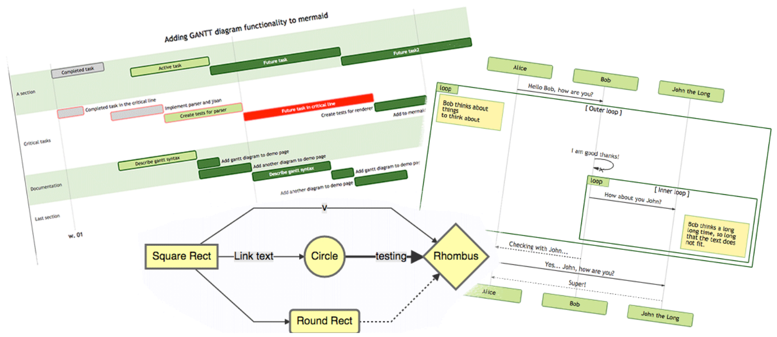
3. **Markup-based drawing tools**

- You use a markup language to describe your diagram.
- A diagram "engine" decides on format, layout etc., so it's less precise.
- e.g., PlantUML, Mermaid.js

Mermaid.js

Mermaid is a JavaScript based diagramming and charting tool that renders Markdown-inspired text definitions to create and modify diagrams dynamically.

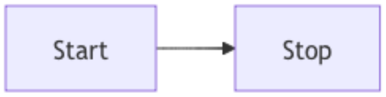
-- [Mermaid.js.org](https://mermaid.js.org)



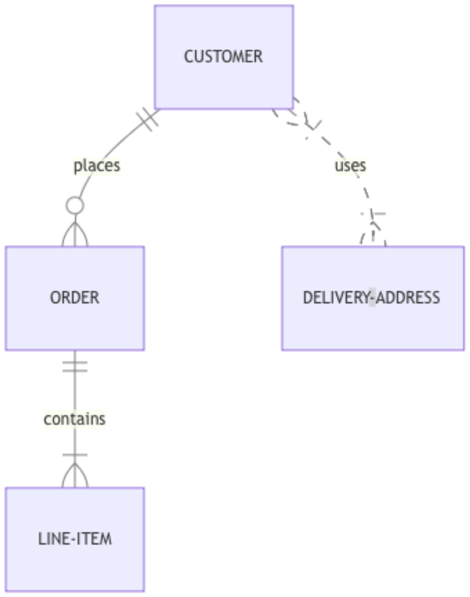
Mermaid supports a HUGE range of diagrams, including all UML diagrams, project charts, etc.

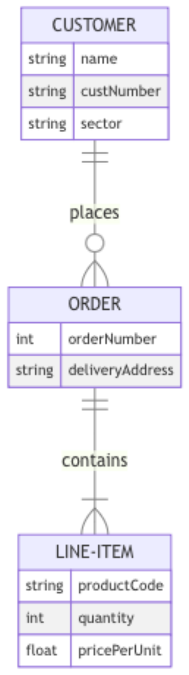
Mermaid.js Diagram Syntax

~~~
```mermaid
flowchart
    LR Start --> Stop
```
~~~

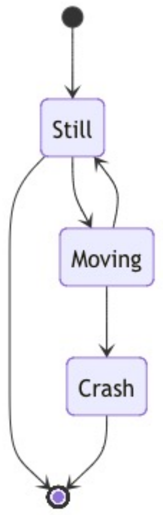


~~~
``` mermaid
erDiagram
    CUSTOMER ||--o{ ORDER : places
    ORDER ||--|{ LINE-ITEM : contains
    CUSTOMER }|..|{ DELIVERY-ADDRESS : uses
```
~~~

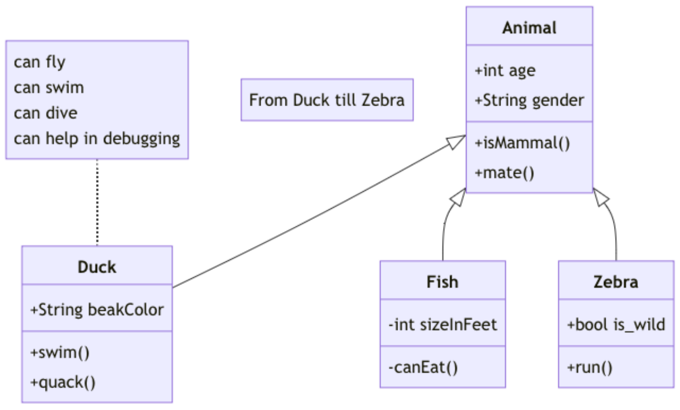




er diagram

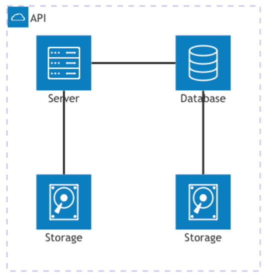


state diagram

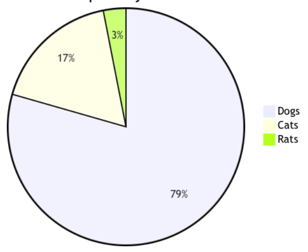


class diagram

See [Mermaid.js documentation](https://mermaid.js.org) for examples

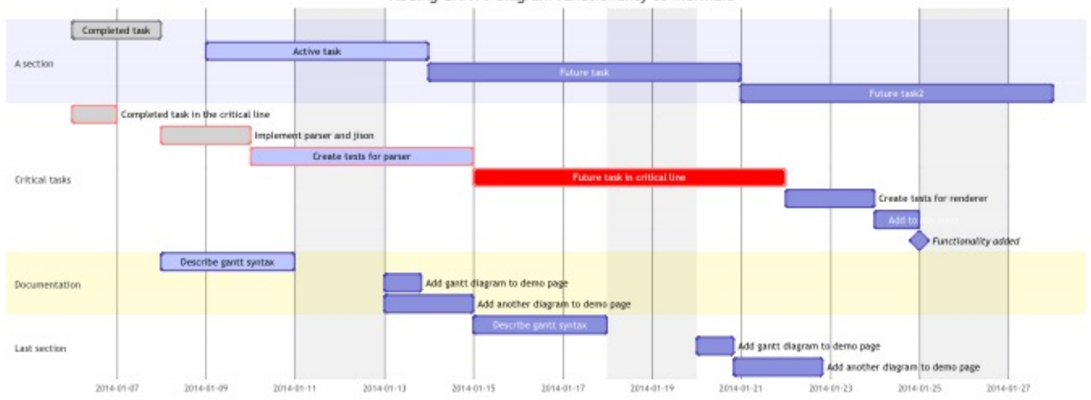


Pets adopted by volunteers



pie chart

Adding GANTT diagram functionality to mermaid

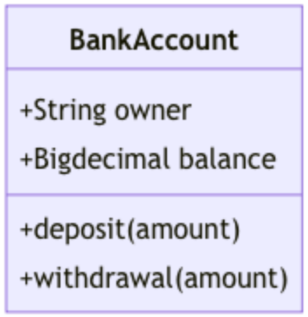


Gantt chart

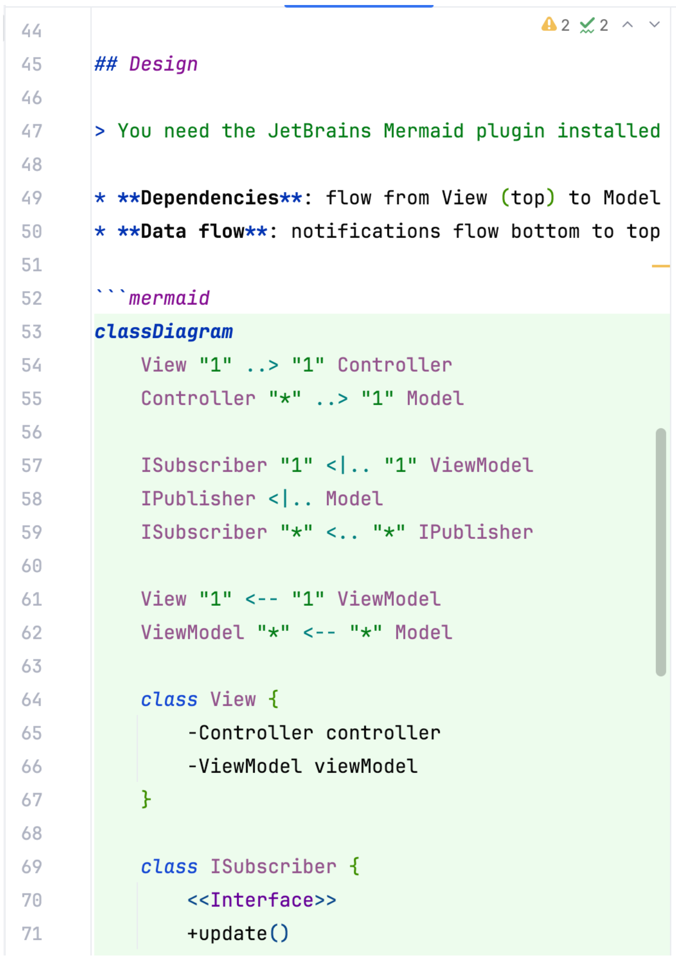
Mermaid.js + Markdown

- Most environments that support Markdown also support Mermaid.
- This includes GitLab, GitHub, VS Code, IntelliJ IDEA, pandoc, …

~~~
``` mermaid
classDiagram
        class BankAccount
        BankAccount : +String owner
        BankAccount : +Bigdecimal balance
        BankAccount : +deposit(amount)
        BankAccount : +withdrawal(amount)
```
~~~



You can wrap Mermaid expressions in code blocks in your Markdown documents and they will be rendered inline.



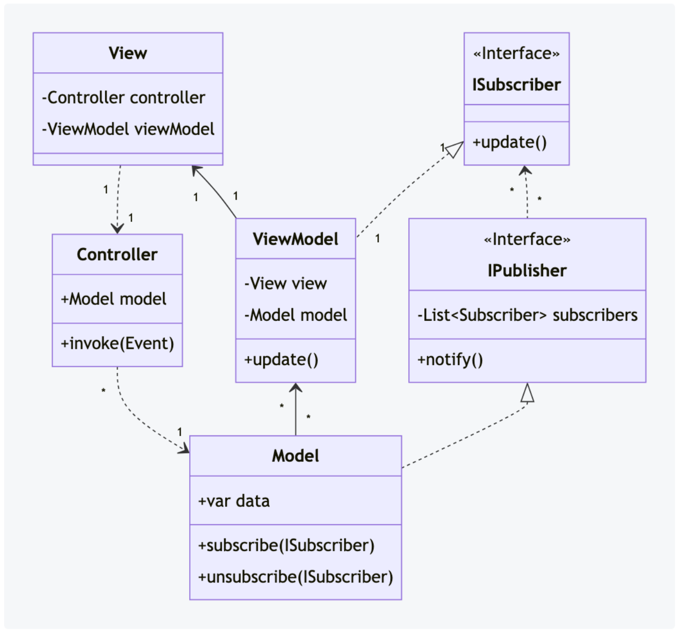
~~~

Design

> You need the JetBrains Mermaid plugin installed

* **Dependencies**: flow from View (top) to Model
* **Data flow**: notifications flow bottom to top

```mermaid
classDiagram
    View "1" ..> "1" Controller
    Controller "*" ..> "1" Model

    ISubscriber "1" <|.. "1" ViewModel
    IPublisher <|.. Model
    ISubscriber "*" <.. "*" IPublisher

    View "1" <-- "1" ViewModel
    ViewModel "*" <-- "*" Model

    class View {
        -Controller controller
        -ViewModel viewModel
    }

    class ISubscriber {
        <<Interface>>
        +update()
~~~

- 1.2. Added Clikt support. Added version catalog.
- 1.3. Desktop GUI support. Split project into modules.
- 1.4. Android support. Added mobile project.

## Design

> You need the JetBrains Mermaid plugin installed to show this diagram.

- **Dependencies**: flow from View (top) to Model (bottom).
- **Data flow**: notifications flow bottom to top via interfaces.



Most tools support Mermaid diagrams in Markdown documents. IntelliJ IDE above shows this diagram inline with Markdown documentation.