Laboration 1

Utveckling av ett tolkat diagram språk och användargränssnitt

Syfte och Bakgrund

- Bakgrund:
  - Visualisering av DSL
- Syfte:
  - Beskriva processen från text till diagram
- Programmeringsspråk:
  - Go
- Mål:
  - Användarvänligt

Översikt över komponenter

- DSL med egen grammatik och tokens
- Lexer & Parser
- Tolk
- Renderer
- Användargränssnitt

DSL - Grammatik och Syntax

- Kontextfri grammatik
- Struktur: `diagram <typ> (attribut) { statements }`
- Exempel: `node A "Start"` och `A -> B "Next"`

```
diagram flowchart {
  node A "Start"
  node B

  A -> B "Next"
}
```

Lexer Design

- Tokenisering med unicode-stöd
- Ignorerar blanksteg, fångar nyckelord och symboler
- Tokenlista:
  - Keywords
  - Identifiers
  - Symbols
  - Strings
  - Arrows
  - Braces
  - EOF

```
node A "Start"

TOKEN_KEYWORD: node

TOKEN_IDENTIFIER: A

TOKEN_STRING: Start
```

```
diagram flowchart (layout=vertical) {
       node A "Start" (color=lightgreen, shape=ellipse, text=black)
       node B "Process"
       node C "END"

       A -> B "Step 1" (color=red, width=3)
       B -> C "Step 2"
}
```

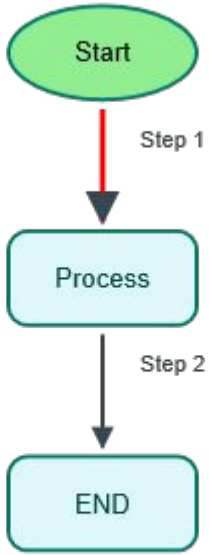
Parser Design

- Recursive descent parser
- Generar AST

Rendering

- Genererar SVG direkt
- Stöd för vertikal/horisontell layout
- SVG komponenter
  - `<rect>`
  - `<ellipse>`
  - `<text>`
  - `<line>`

```
<ellipse cx="100" cy="50" rx="40" ry="20" fill="lightgreen"
stroke="black" />
<text x="100" y="55" text-anchor="middle"
fill="black">Start</text>
```


Start
Step 1
Process
Step 2
END

Användargränsnitt

- CLI: Automatisk rendering från .diag-filer
- TUI: Interaktiv terminal via Bubble Tea och lipgloss

Sammanfattning

- Komplett system i Go: DSL -> SVG
- Flexibelt, Utbyggbart
- Demo