

sieh die
AUSARBEITUNG
und das CODE
hier an!

scanne mich!



Das Expression Problem: Lösung mit Generics in TypeScript

wird präsentiert von
Mohammadreza Javadi

Schnall dich an – Ich haben heute viel zu zeigen!

Gliederung

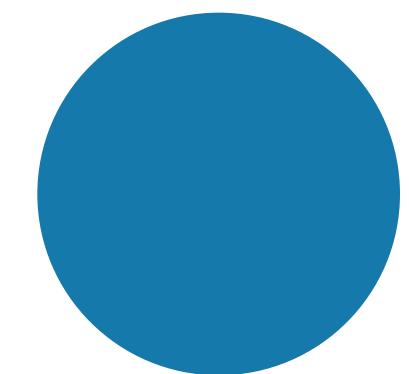
History

Was ist Genau EP?

Lösung mit Generics in TS

Fazit

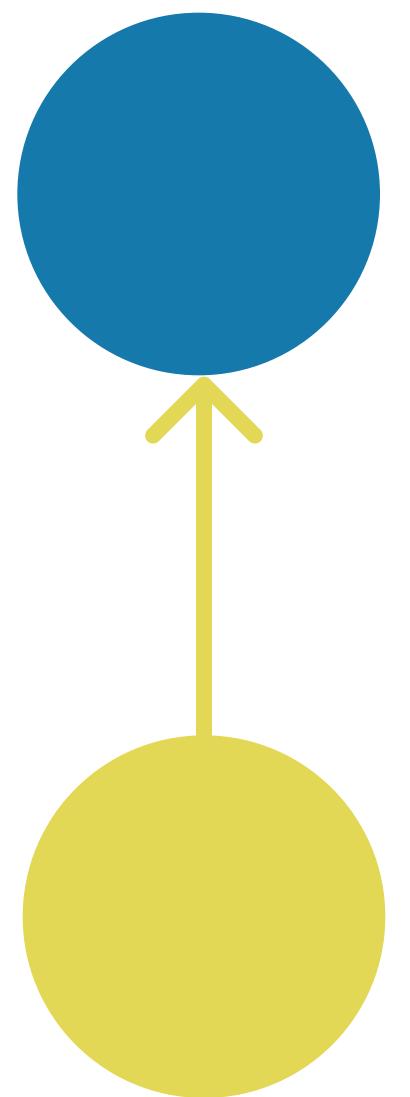
1975



John Reynolds

- User-defined Types
- Procedural Data Structures

1975

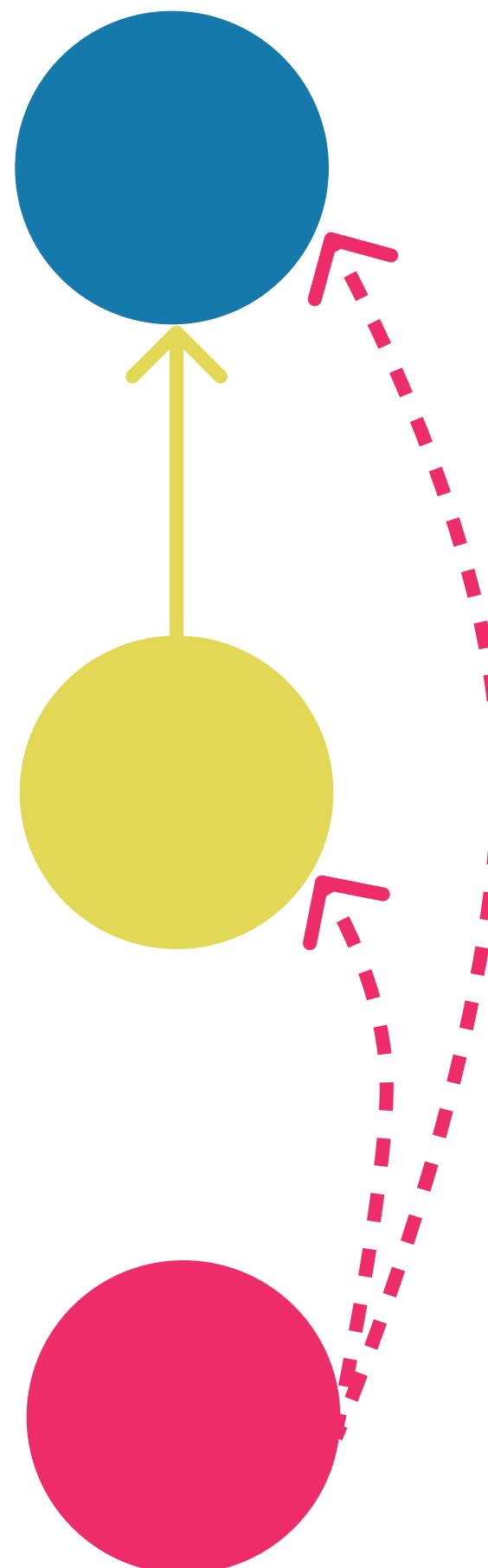


John Reynolds

William Cook

	Cal Area	Cal Perimeter	New Op
type			
Circle	✓	✓	✗
Rectangle	✓	✓	✗
New type	✓	✓	

1975



John Reynolds

1990

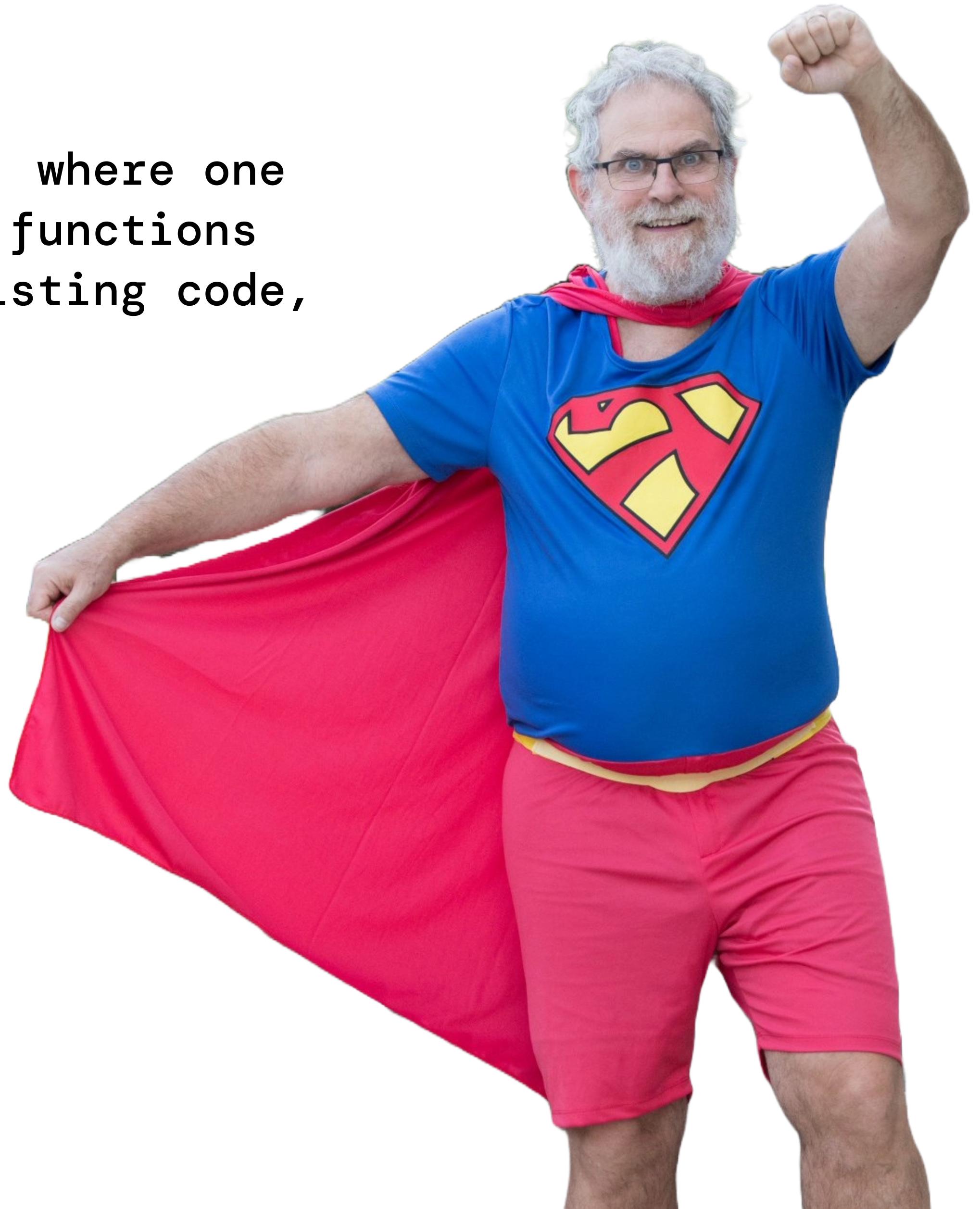
William Cook

1998

Philip Wadler

The goal is to define a datatype by cases, where one can add new cases to the datatype and new functions over the datatype, without recompiling existing code, and while retaining static type safety.

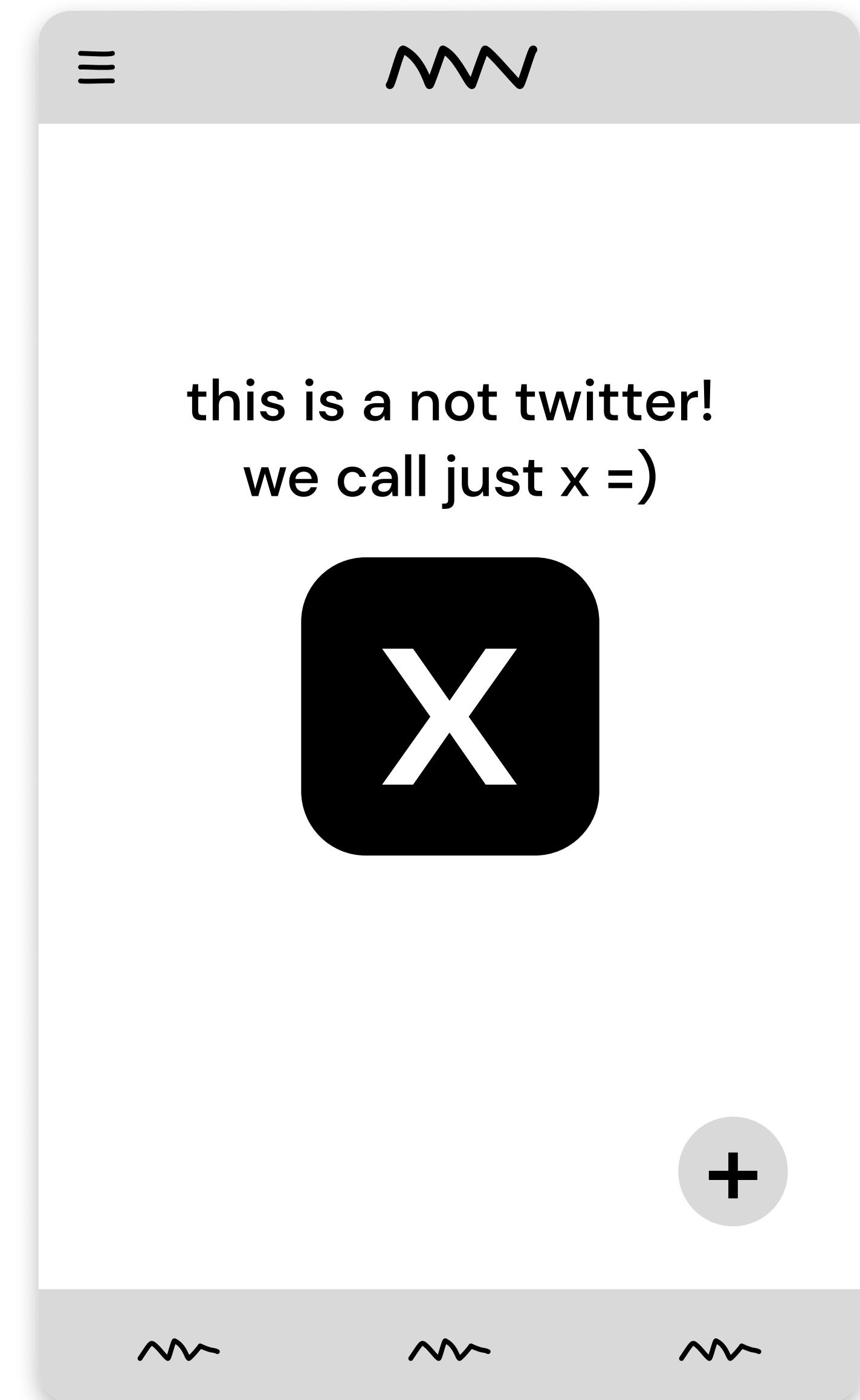
Philip Wadler, 12 November 1998



also....

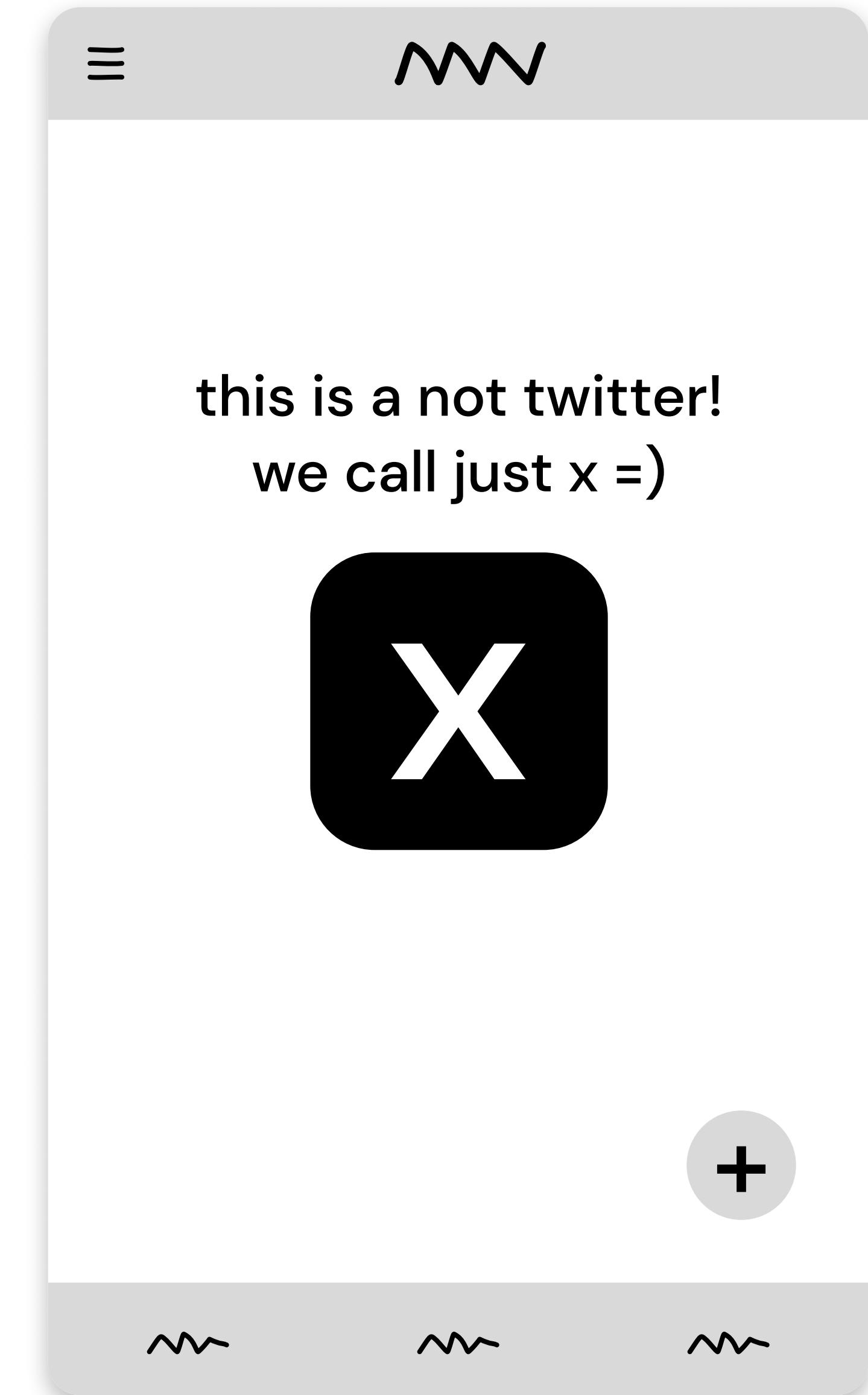
Was ist Genau
Expression Problem?

**Denk an eine
Social Media App
wie X (früher Twitter)**



Denk an eine Social Media App wie X (früher Twitter)

- aufgabe: entwickelt eine einfache app, die text als string anzeigt!



Wir installieren einen Datentyp für POST!

IPost.ts TypeScript

```
interface Post {  
    display(): void;  
}
```



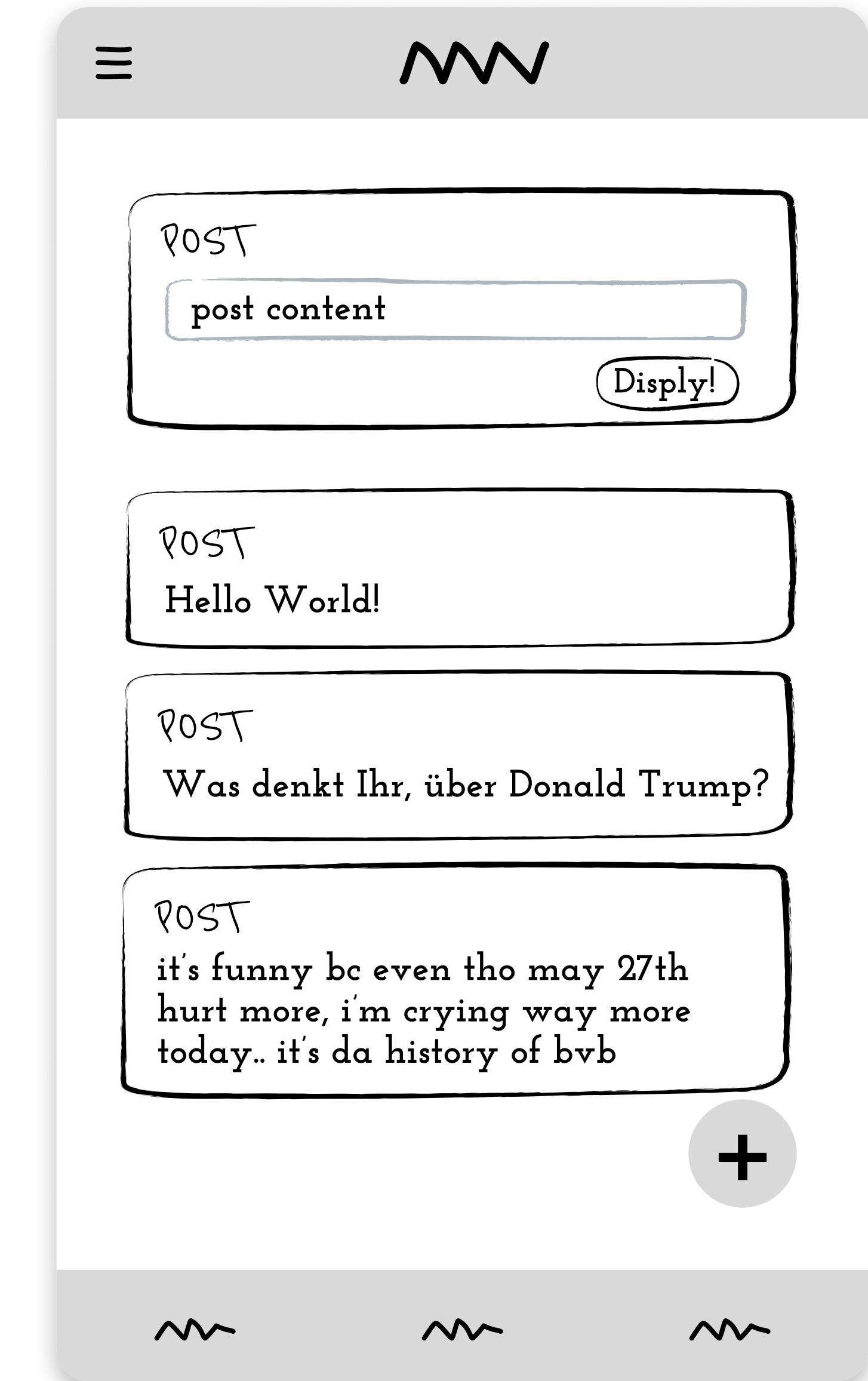
Dann erstellen wir unsere Klasse für TextPost!

IPost.ts TypeScript

```
class TextPost {
    content: string;

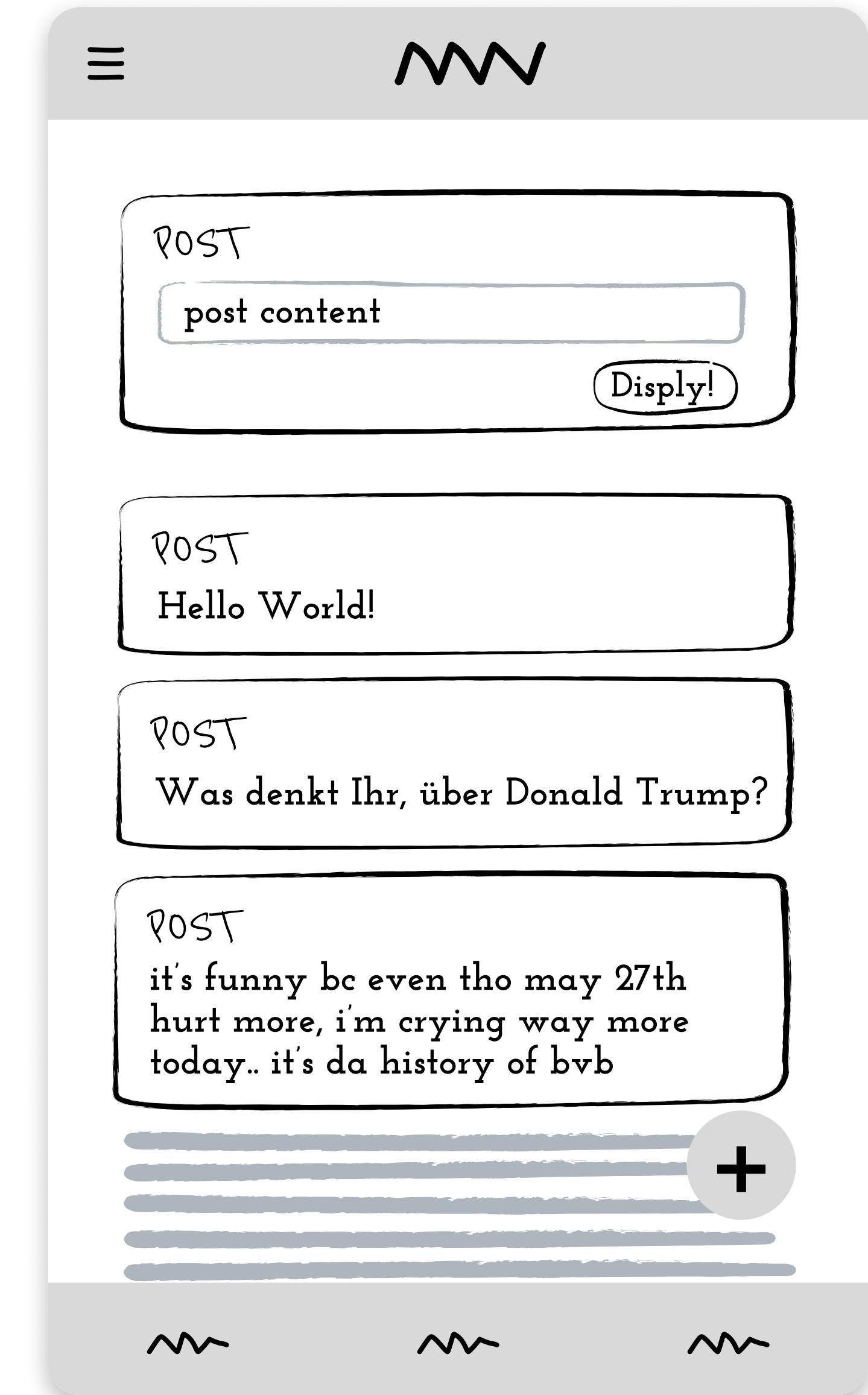
    constructor(content: string) {
        this.content = content;
    }

    display(): void {
        console.log(`POST: ${this.content}`);
    }
}
```





**Herzlichen
Glückwunsch!
Unsere App hat
1000 Posts
erreicht!**

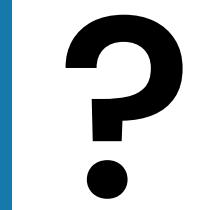


FRAGE:

Wie können wir unsere
App erweitern?

FRAGE:

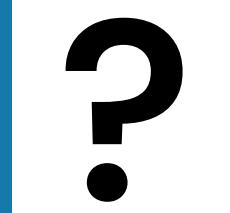
Wie können wir unsere
App erweitern?



Neuer Datentyp

FRAGE:

Wie können wir unsere
App erweitern?



Neuer Datentyp

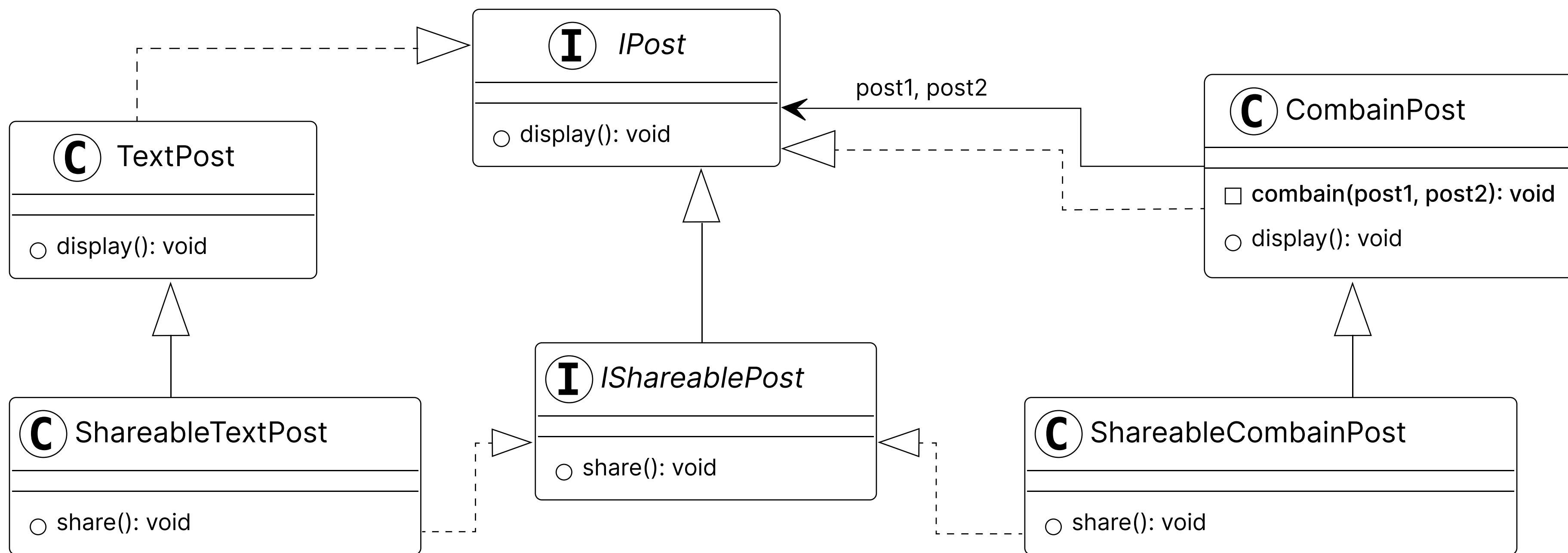


Neue Funktionalität

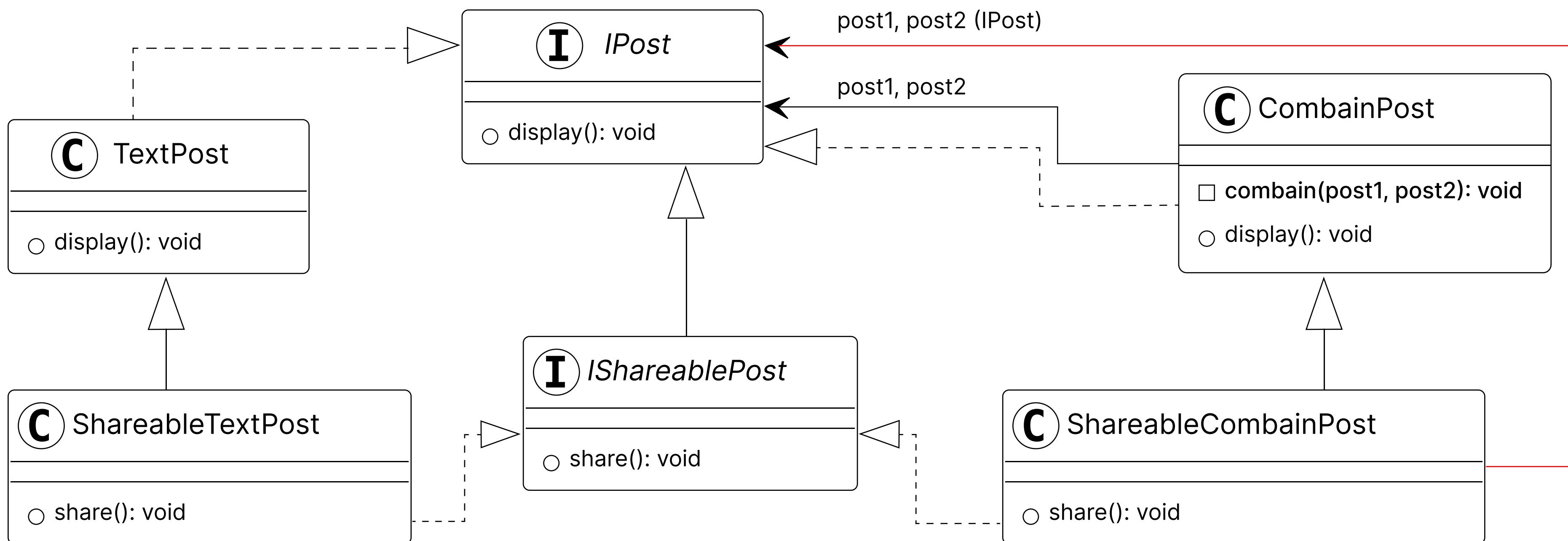
um darauf zu antworten, schauen wir uns an

Klassische Ansätze

klassen-basiert ansatz



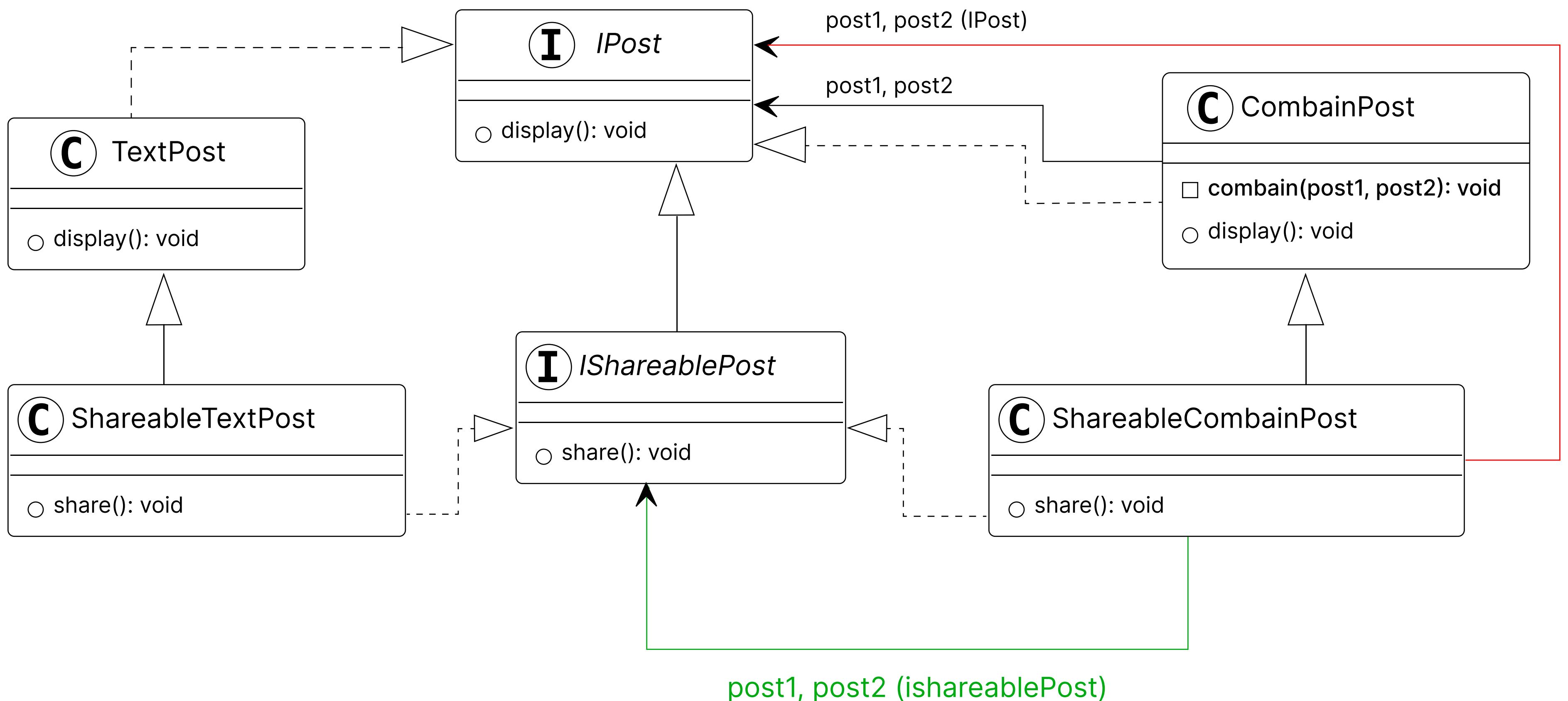
klassen-basiert ansatz



Das Expression Problem tritt auf.

Dabei sollen 'post1' und 'post2' das Interface **IShareablePost** implementieren.

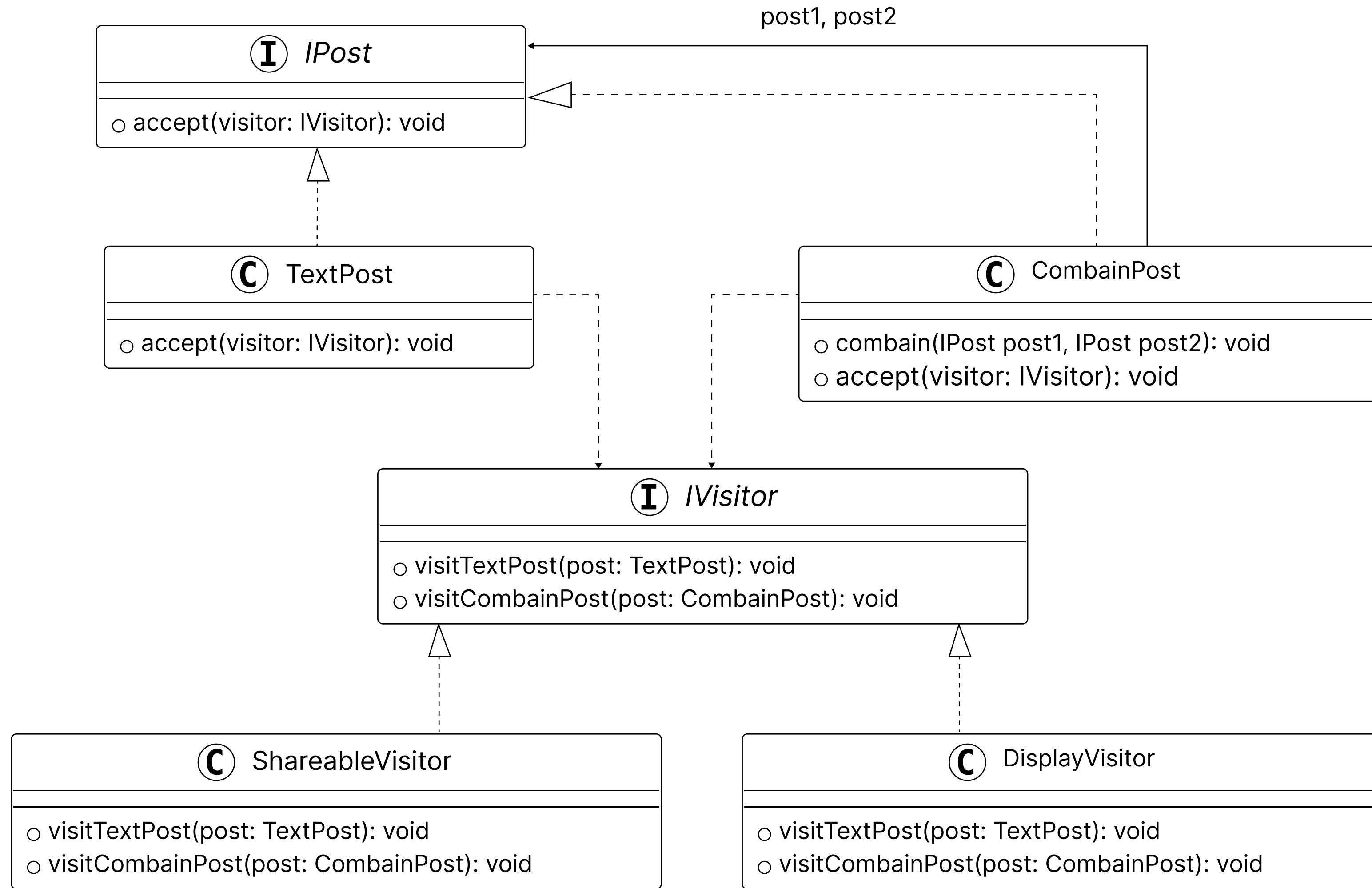
klassen-basiert ansatz



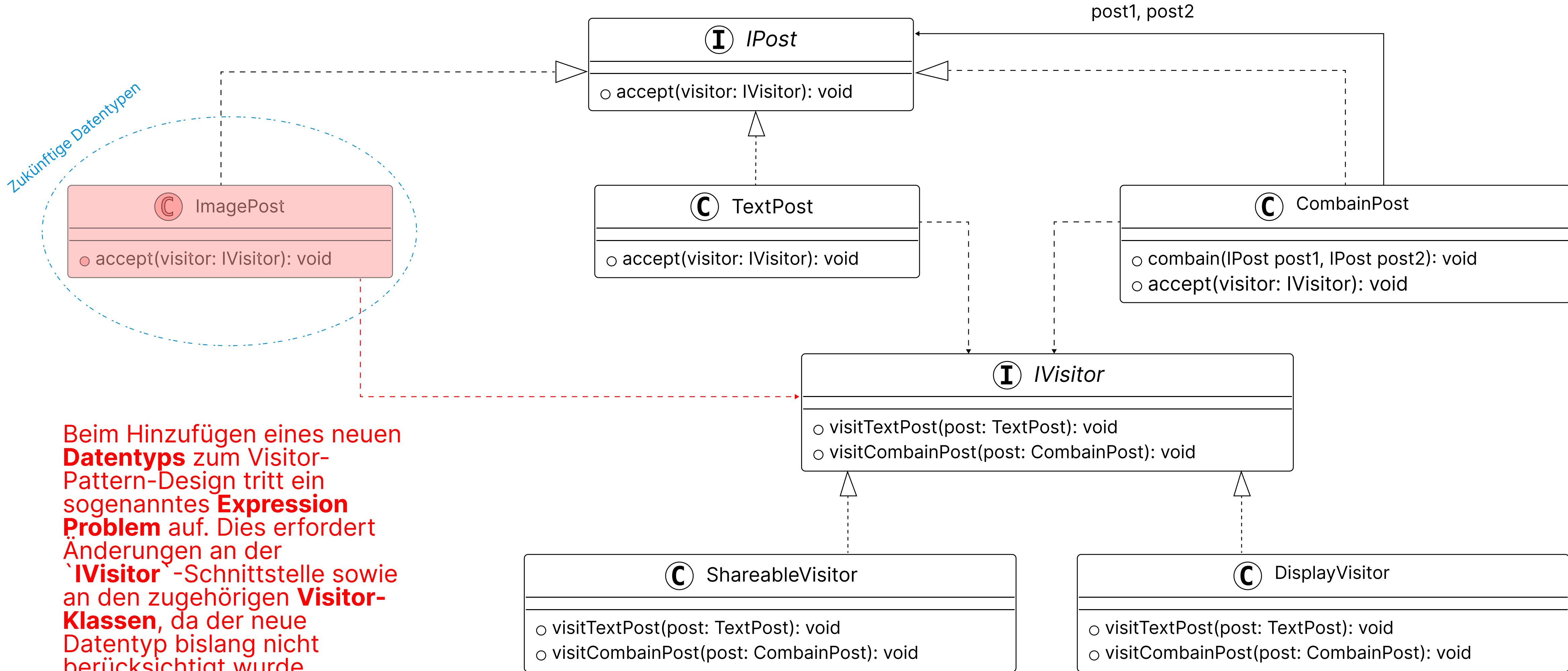
Das Expression Problem tritt auf.

Dabei sollen 'post1' und 'post2' das Interface *IShareablePost* implementieren.

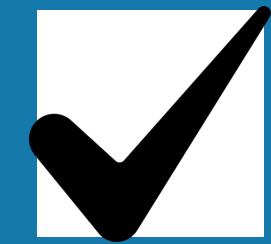
visitor-pattern ansatz



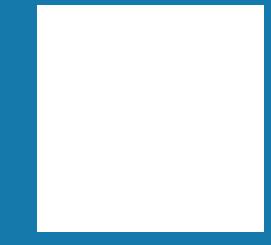
visitor-pattern ansatz



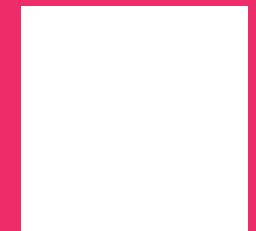
Klassen-basiert vs Visitor-Pattern



Neuer Datentyp



Neue Funktionalität



Neuer DatenType



Neue Funktionalität

Eine Lösung sind Generics!

Aber...

Was ist Generics?

```
function identity<T>(arg: T): T {  
    return arg;  
}
```

Implementierung

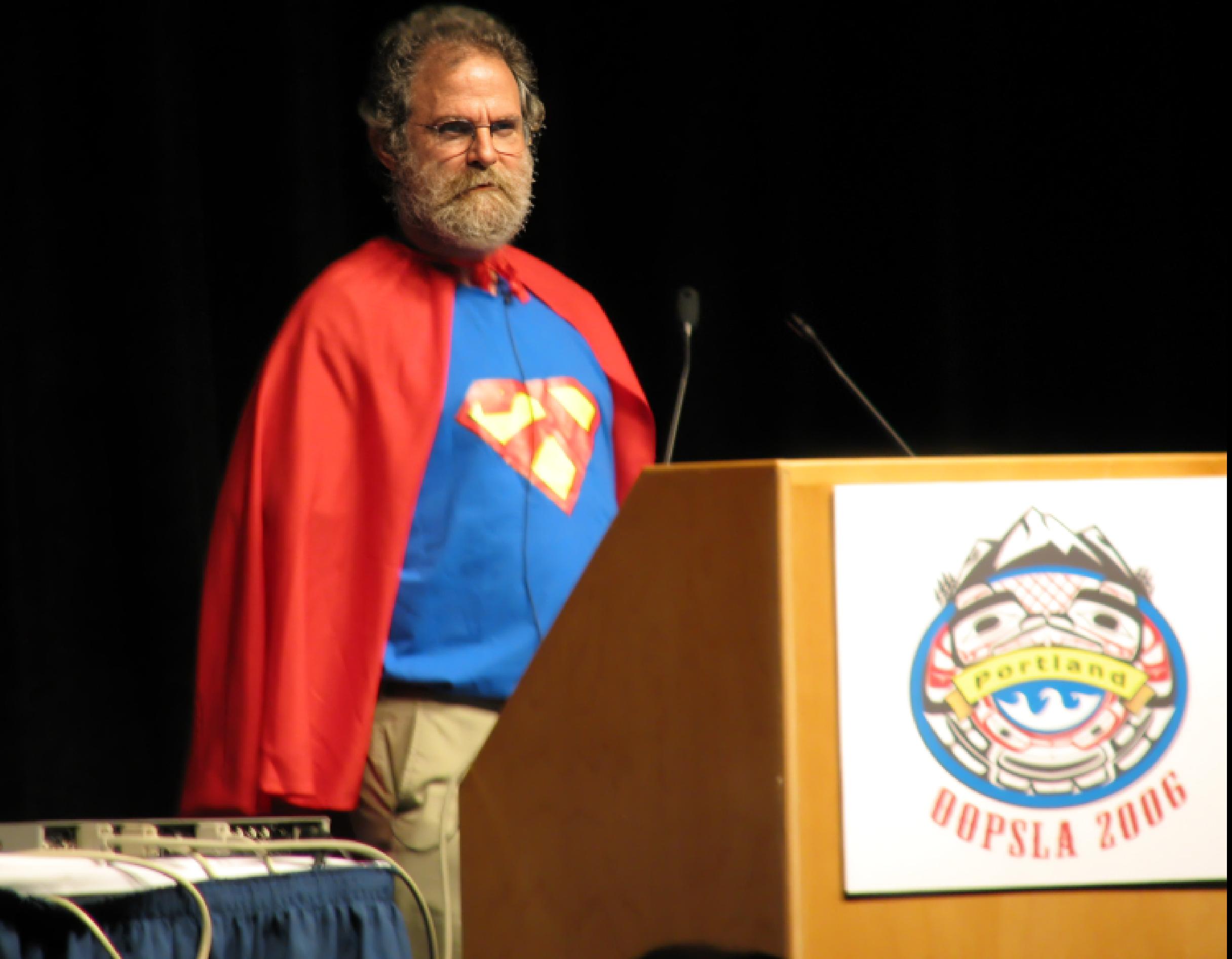
in

<TypeScript/>

FAZIT

- **effektive nutzung von generics:** typescript ermöglicht flexible, typsichere softwarelösungen.
- **modularität und wartbarkeit:** die anwendung von generics fördert die erweiterbarkeit und modularität des codes.
- **lösung des expression problems:** generics verbessern die erweiterbarkeit und wiederverwendbarkeit von software erheblich.
- **zukunftssichere softwareentwicklung:** die techniken bieten eine solide grundlage für die entwicklung zukünftiger projekte.
- **herausforderungen:** trotz vorteilen erhöht sich die komplexität, die durch sorgfältiges design und planung minimiert werden kann.

FRAGEN?



*The expression problem is so called because, once you start thinking about it, you keep thinking about it.
This is an expression of frustration!*

It's quoted from Al

Quellen

- [1] philip wadler. "the expression problem." online verfügbar unter: <https://homepages.inf.ed.ac.uk/wadler/papers/expression/expression.txt>, letzter zugriff: 18. juli 2024.
- [2] william cook. "object-oriented programming versus abstract data types." in j. w. de bakker, w. p. de roever, und g. rozenberg (hrsg.), foundations of object-oriented languages (fool), rex school/workshop, band 489 der lecture notes in computer science, seiten 151–178, noordwijkerhout, niederlande, 1990. springer berlin heidelberg.
- [3] john c. reynolds. "user-defined types and procedural data structures as complementary approaches to data abstraction," seiten 309–317. springer new york, new york, ny, 1978. ursprünglich präsentiert auf der ifip working group 2.1 konferenz zu new directions in algorithmic languages, münchen, 1975.
- [4] nathan rozentals. mastering typescript. packt publishing, 2019.