DESIGN PATTERNS

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." [1]

The idea was adopted by software engineering
Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (GoF) - Design Patterns: Elements of Reusable Object-Oriented Software (1994)

A pattern has four essential elements:

1. The **pattern name** is a handle we can use to describe a design problem, its solutions, and consequences in a word or two. It makes it easier to think about designs and to communicate them and their trade-offs to others.

2. The **problem** describes when to apply the pattern. It explains the problem and its context.

3. The **solution** describes the elements that make up the design, their relationships, responsibilities, and collaborations. **The solution doesn't describe a particular concrete design or implementation**, because a pattern is like a template that can be applied in many different situations. Instead, the pattern provides an abstract description of a design problem and how a general arrangement of elements (classes and objects in our case) solves it.

4. The **consequences** are the results and trade-offs of applying the pattern.

Categories of patterns

- Architectural patterns
- Design patterns
- Language level patterns. This is the lowest level of the pattern-categories, also known as idioms

---

Idioms - STRING COPY

Naive implementation of string copy

```
void myStrcpy1(char *dest, char *source){
  while ((*source)!='\0'){
    *dest=*source;
    source++;
    dest++;
  }
  *dest='\0';
}
```

Idioms - STRING COPY

The implementation you can find in standard libraries or in Kernighan & Ritchie

```
void myStrcpy2(char *dest, char *source){
   while(*dest++ = *source++);
}
```

Idioms - SIZEOF TO VARIABLES

```
HelloTelegram *p_telegram = malloc( sizeof( HelloTelegram ));


HelloTelegram *p_telegram = malloc(sizeof( *p_telegram ));
```

1. Christopher Alexander, "A Pattern Language", 1977. https://en.wikipedia.org/wiki/Christopher_Alexander ↩