UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Modularidade

QXD0007 - Programação Orientada a Objetos

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Conteúdo

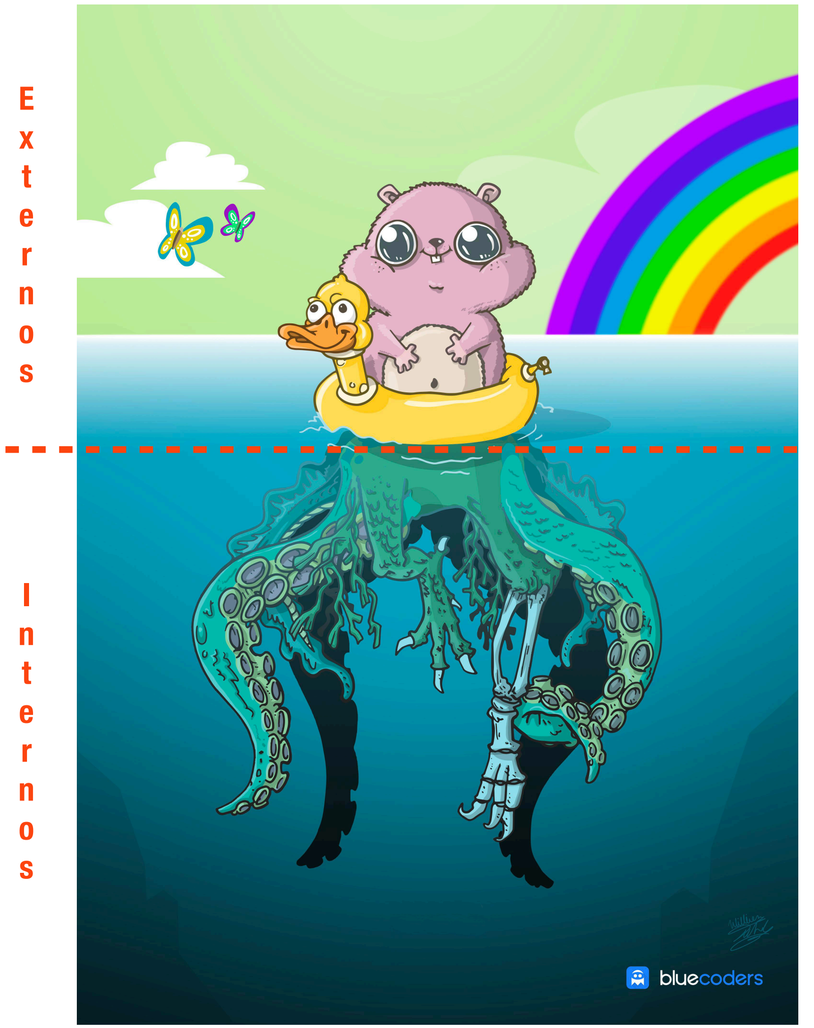
- Qualidade de Software
- Modularidade de Software
- Exemplo prático de modularização
- Orientação a Objetos e modularização

Qualidade de Software

Qualidade de Software

Visão geral

- Qualidade de software é uma combinação de vários fatores
  - Externos e Internos
    - No final, os **aspectos externos são os que importam**
    - A chave para alcançá-los são os fatores internos
- A engenharia de software tem como objetivo a construção de software de qualidade



Qualidade de Software

Principais aspectos

Corretude

- É a habilidade do software realizar suas tarefas de acordo com suas definições na especificação
- **Desafio:** Especificar os requisitos de um sistema de forma precisa

Qualidade de Software

Principais aspectos

Robustez

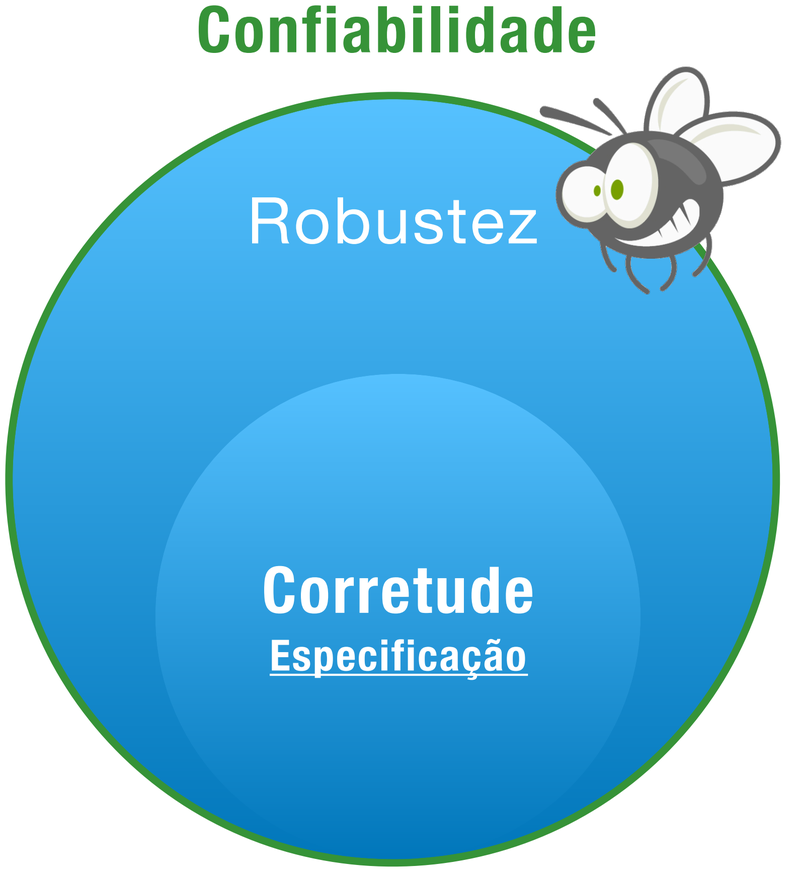
- É a habilidade do software de reagir apropriadamente a condições anormais

Qualidade de Software

Principais aspectos

Corretude x Robustez

- Corretude trata do comportamento previsto na especificação
- Robustez lida com o que acontece fora do especificado
- Corretude + Robustez = Confiabilidade

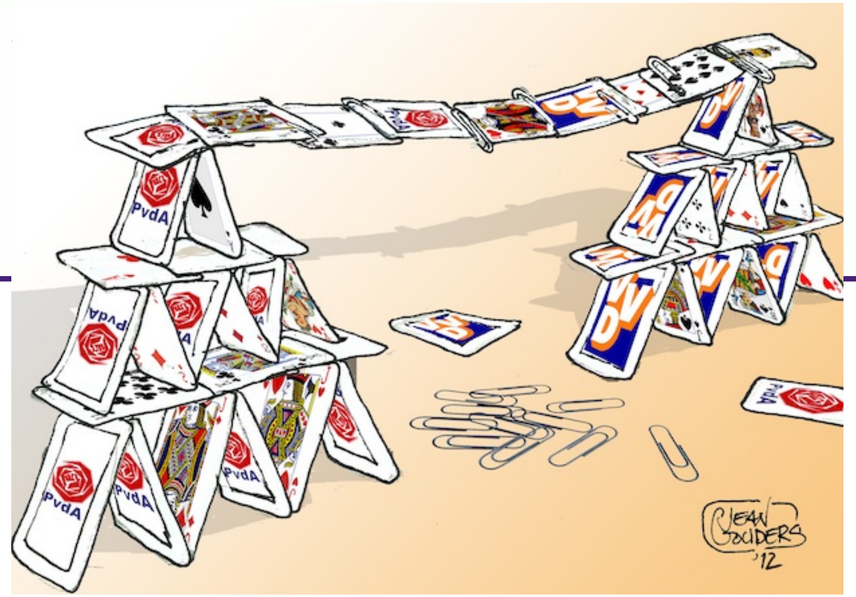


Qualidade de Software

Principais aspectos

Extensibilidade

- É a facilidade de adaptar o software à mudanças de especificação
- Fácil em pequenos programas
- A medida que o software cresce tende a ficar mais difícil



Qualidade de Software

Principais aspectos

Reusabilidade

- É a habilidade que um elemento do software tem de ser utilizado para construir diferente aplicações
- Software frequentemente seguem padrões similares
- Influência todos os outros aspectos da qualidade de software

Qualidade de Software

Principais aspectos

Extensibilidade x Reusabilidade

- Softwares devem ser fáceis de modificar
- Elementos de software que nós produzimos devem ser aplicáveis de forma geral
- Extensibilidae + Reusabilidade = Modularidade

A orientação a objetos pode melhorar significantemente esse quatro aspectos da qualidade e outros mais

Modularidade de Software

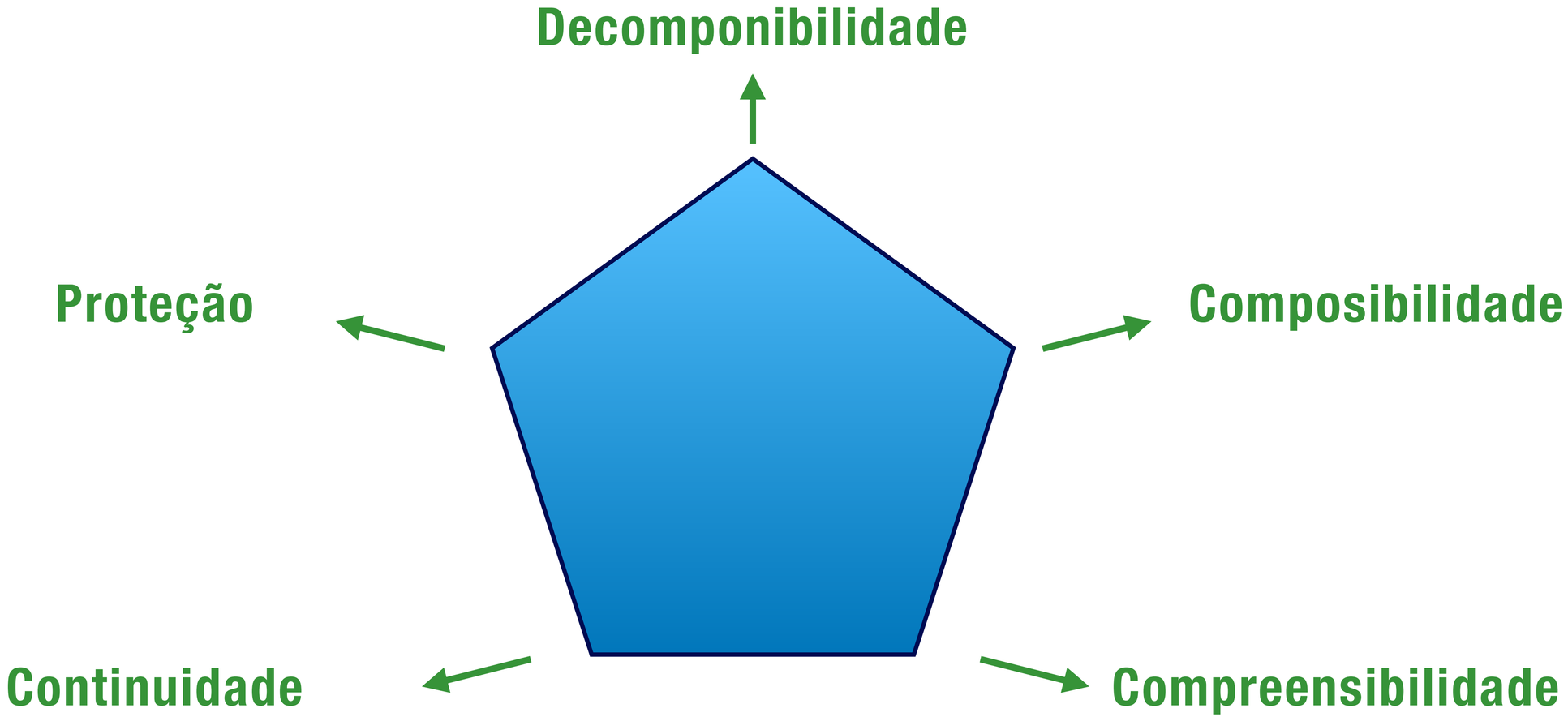
Modularidade de Software

Programação modular

- Construção de programas como um conjunto de peças menores (sub-rotinas)
- Os módulos sejam auto-contidos e organizados em uma arquitetura estável
- Software modular -> Software mais manutenível
  - Divisão e conquista

Modularidade de Software

5 requisitos fundamentais de modularidade

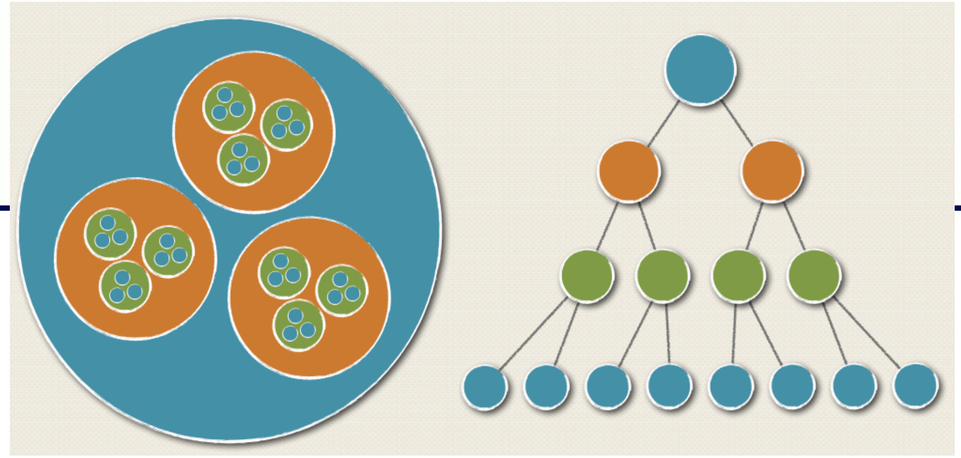


Modularidade de Software

5 requisitos fundamentais de modularidade

Decomponibilidade

- Favorece a tarefa de decompor um software em um conjunto pequeno de subproblemas menores e menos complexos

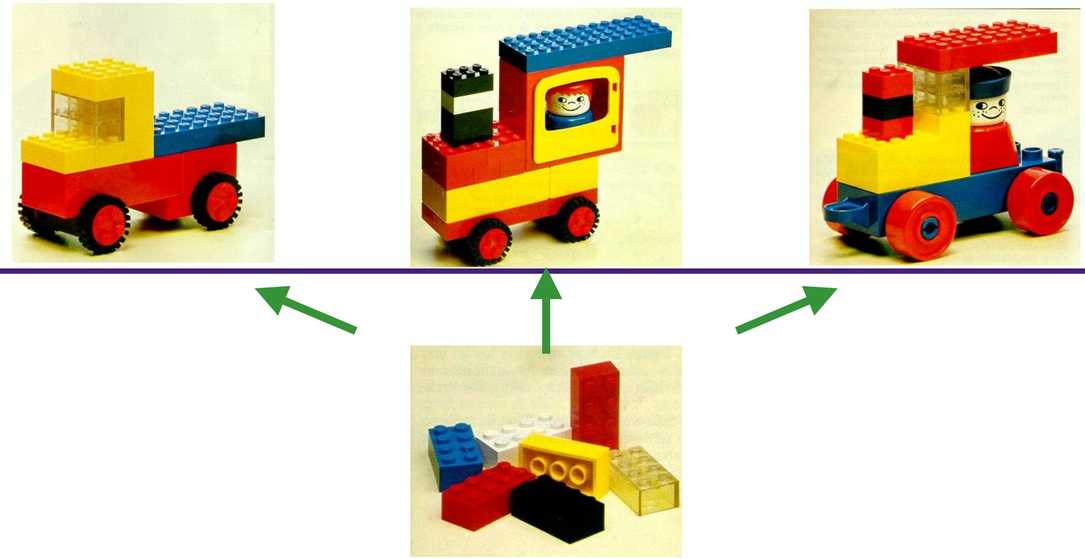


Modularidade de Software

5 requisitos fundamentais de modularidade

Componibilidade

- Favorece a construção de elementos de software que podem ser livremente combinados

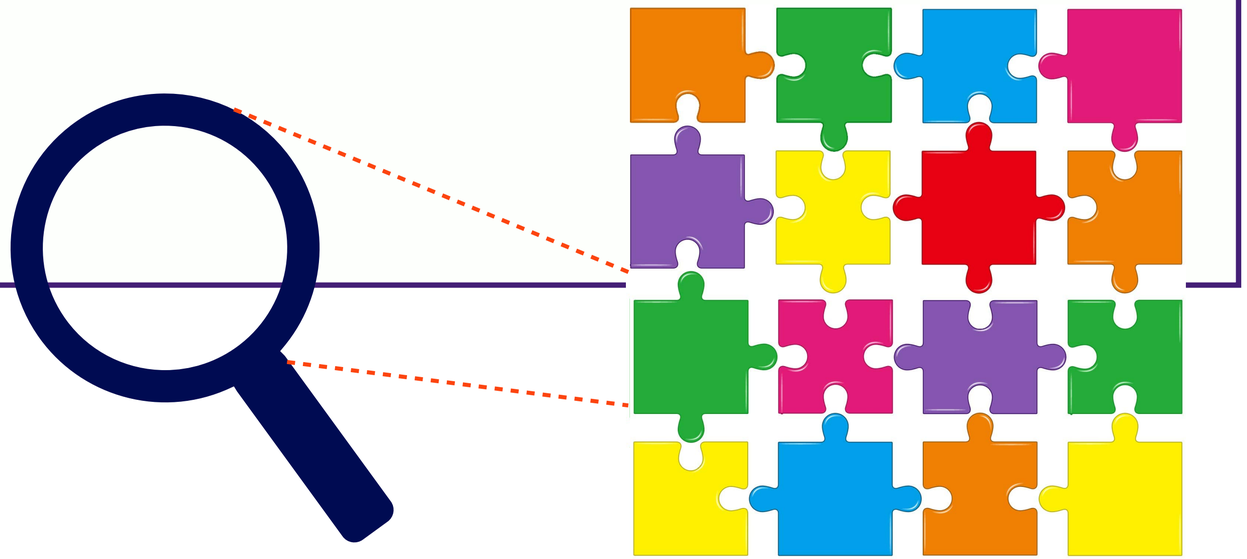


Modularidade de Software

5 requisitos fundamentais de modularidade

Compreensibilidade

- Favorece a construção software compostos de módulos que podem ser compreendidos por humanos sem o conhecimentos de outros módulos

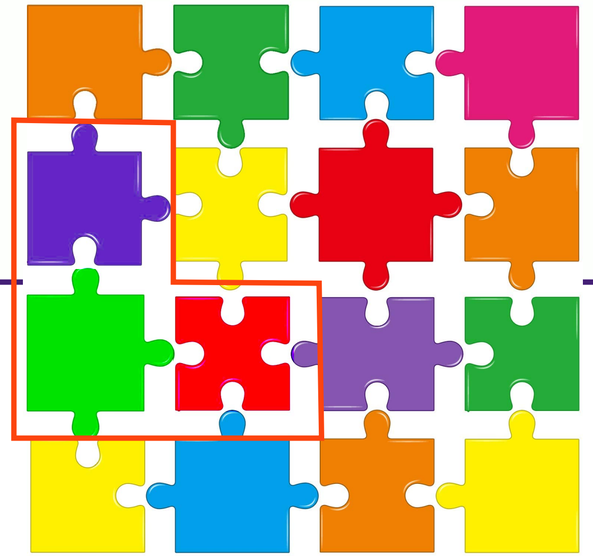


Modularidade de Software

5 requisitos fundamentais de modularidade

Continuidade

- Uma pequena mudança de especificação afetar apenas um ou poucos módulos

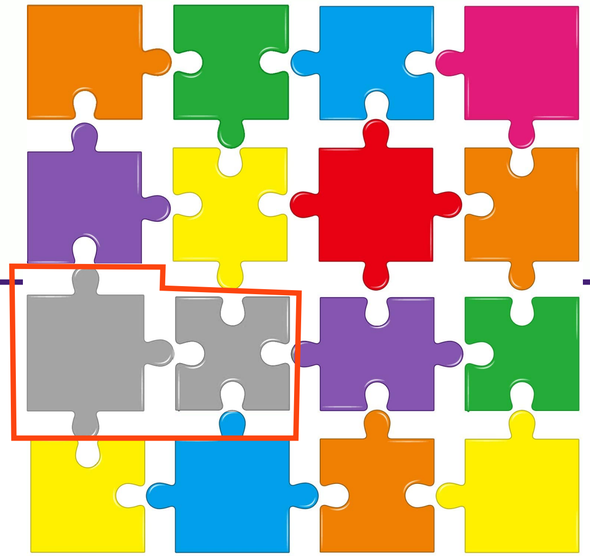


Modularidade de Software

5 requisitos fundamentais de modularidade

Proteção

- Uma efeito anormal durante a execução afetar apenas um ou poucos módulos vizinhos



Modularidade de Software

Estruturação de programas

Assembly ARM

```
start:
    MOV R0, #3   @ Put the value 3 into R0
    MOV R1, #0   @ Put the value 0 into R1

loop:
    CMP R0, R1   @ Compare R1 to R0 (effectively R0 minus R1)
    BEQ     done  @ If they are equal (Z=1) branch to done
label
    ADDGT R1, #1  @ If R0 is greater than R1 add 1 to R1
    SUBLT R1, #1  @ If R0 is less than R1 subtract 1 from R1
    B loop        @ Branch back and re-run loop

done:
    @ do other stuff
```

C

```
int r0 = 3;
int r1 = 0;


while(r0 != r1) {
  if(r0 > r1) {
    r1++;
  }
  if (r1 > r0) {
    r1--;
  }
}
```

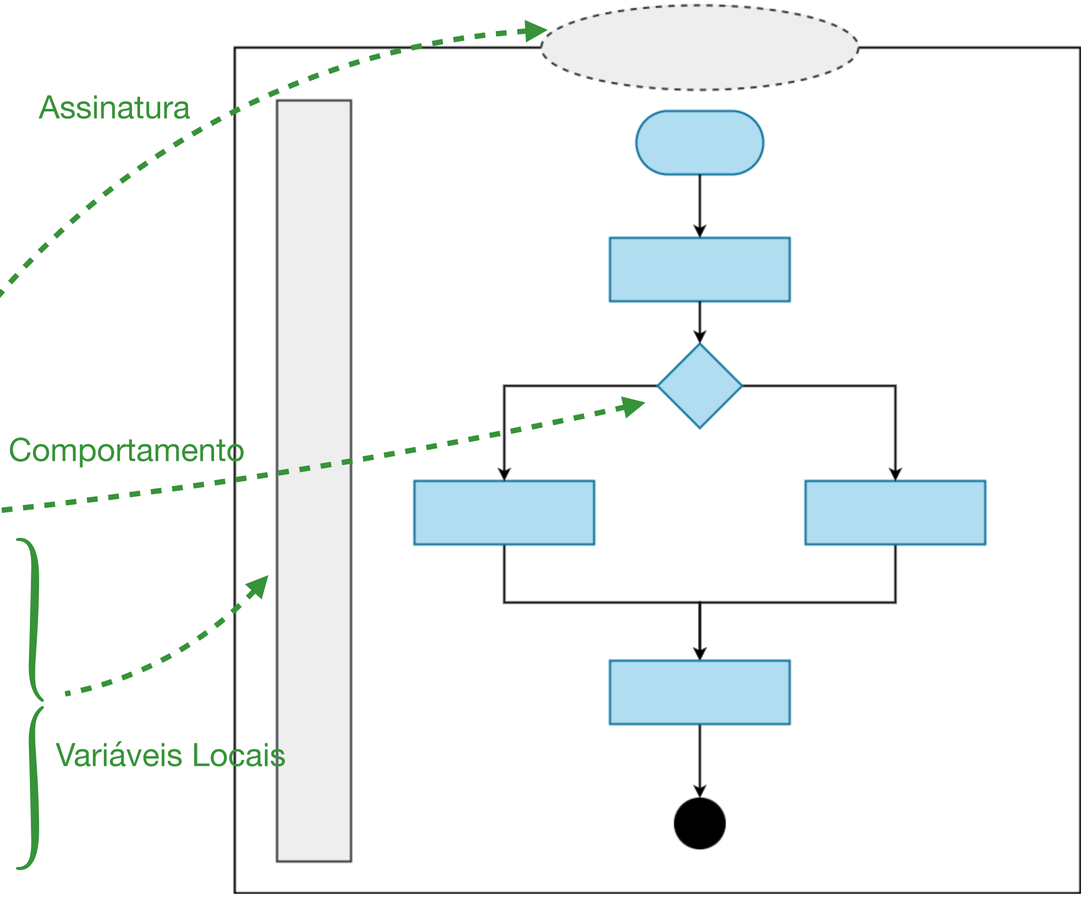
Modularidade de Software

Estruturação de programas

Programação procedural

O procedimento é a unidade de modularidade

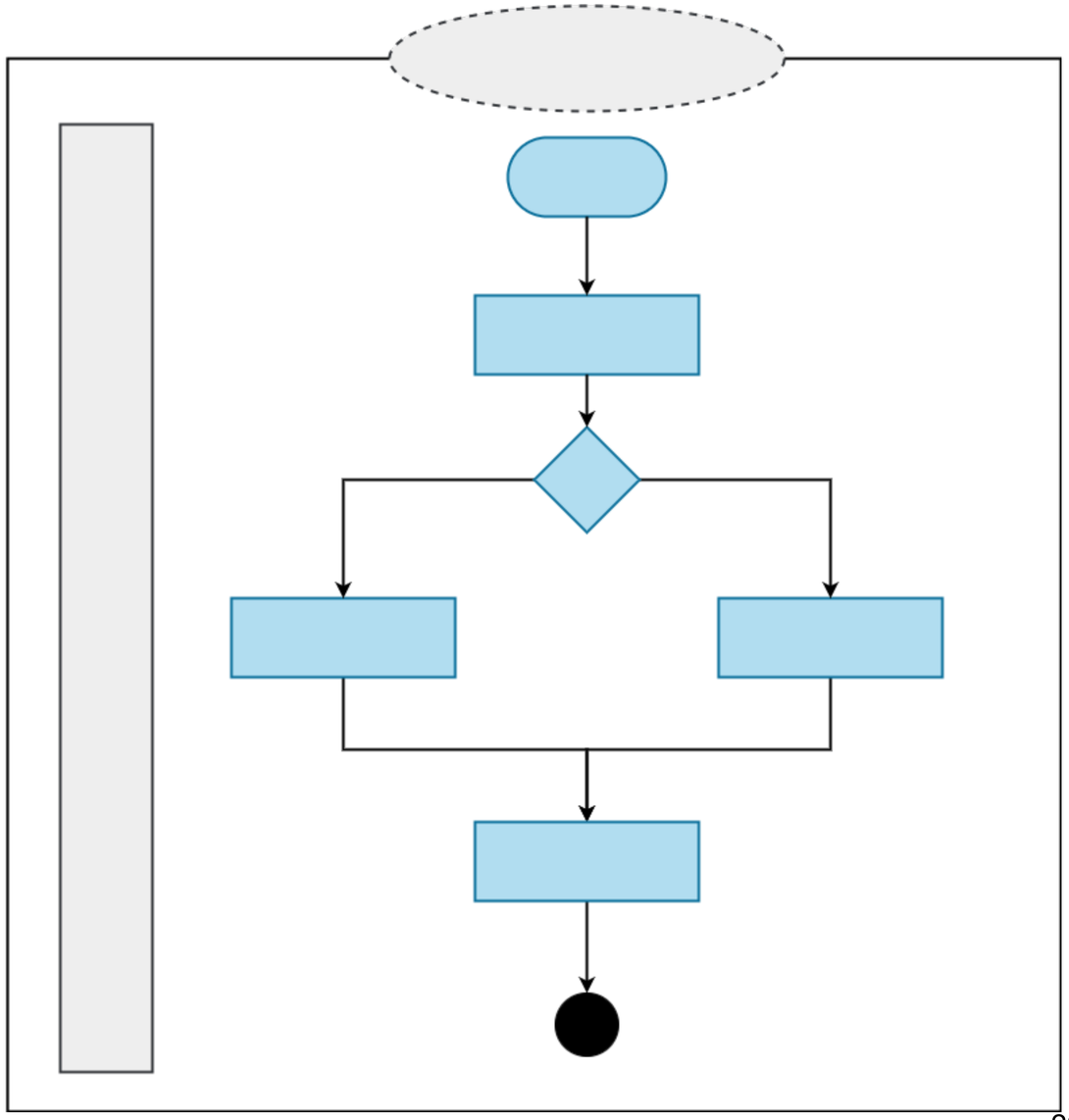
```c
float example(int r0, int r1) {
  if(r0 > r1) {
    r1++;
  } else {
    r1--;
  }
  return r0 + r1;
}
```



Modularidade de Software

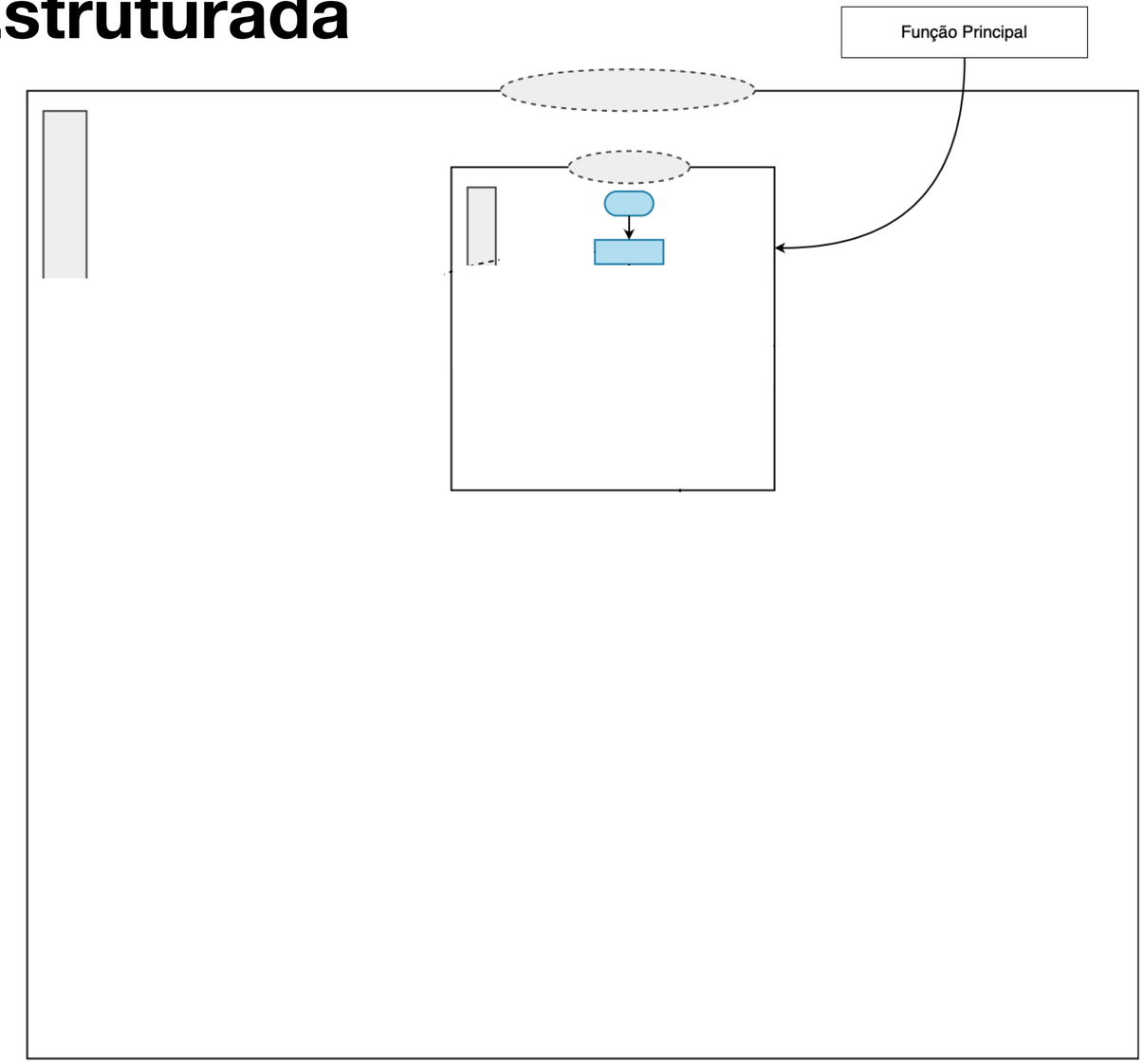
Programação estruturada

- Introduziu estruturas de controle
  - `If, if-else, while, do-while`
- Algumas vantagens
  - Organização
  - Indireção explícita
  - Prova de corretude
  - Análise de complexidade



Modularidade de Software

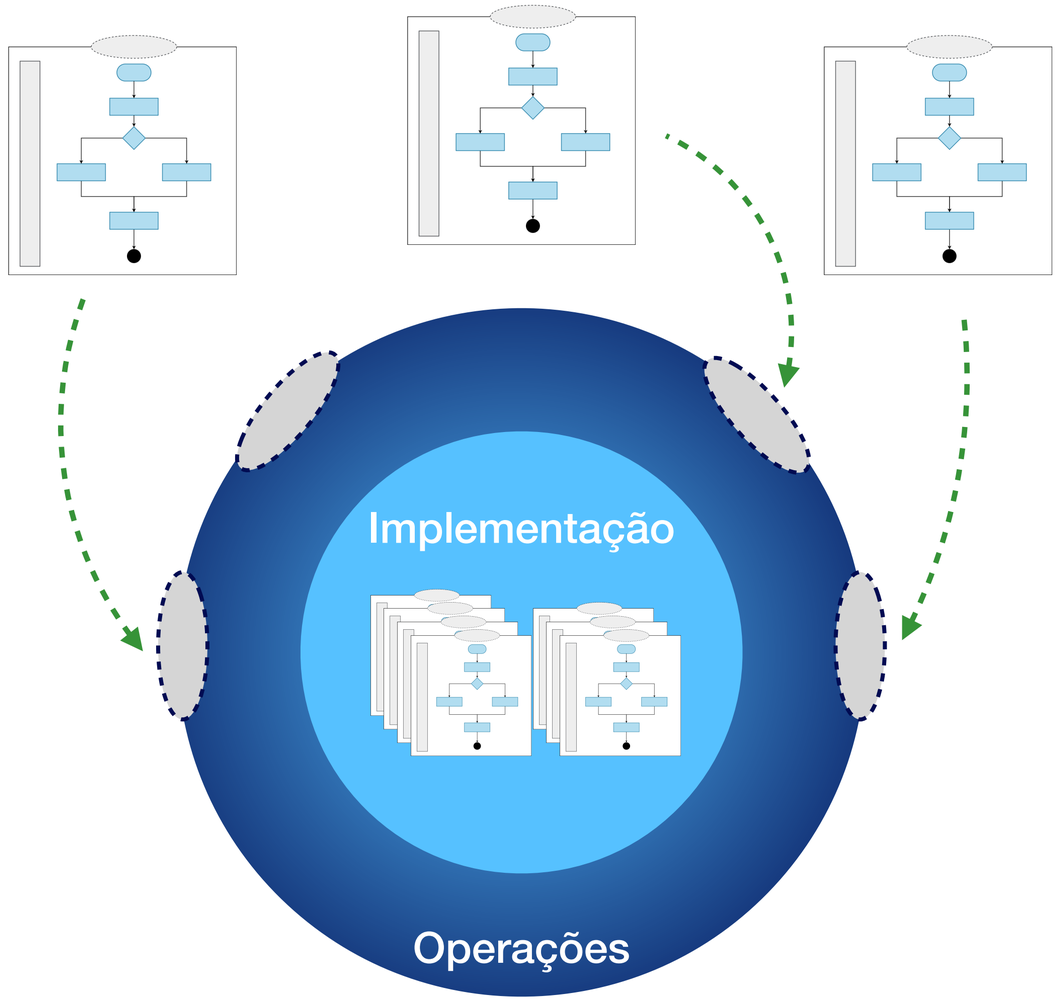
Programação Estruturada



Modularidade de Software

Tipo abstrato de dados

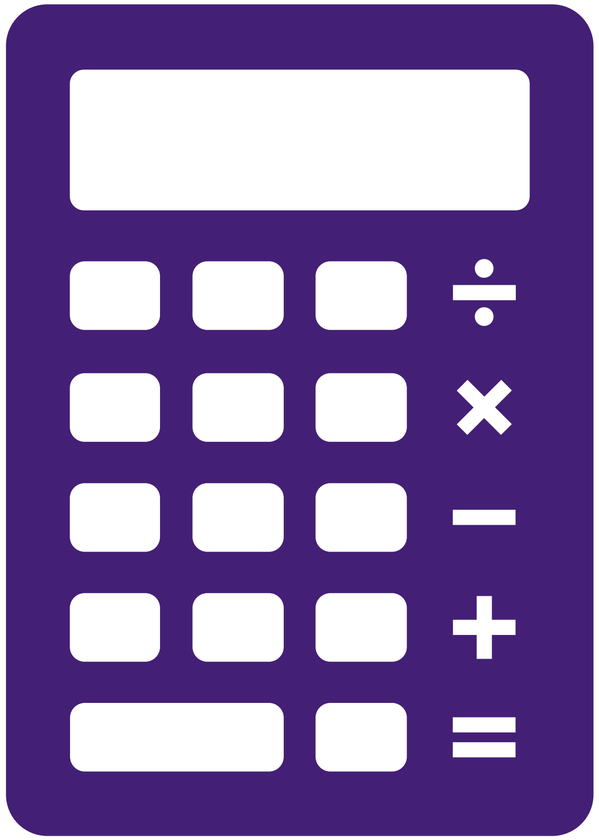
- Especificação:
  - Conjuntos de dados
  - Operações
    - Implementação escondida
- Interface visível



Exemplo prático de modularização

Calculadora

Exemplo prático usando a linguagem C



Exemplo prático de modularização

- Calculadora v1
  - Apenas a função principal
- Calculadora v2
  - Operações divididas em Funções
- Calculadora v3
  - Operações separadas um arquivo diferente do que contém a função principal

Orientação a Objetos e modularização

Orientação a Objetos e modularização

Programação procedural

- Procedimento é a unidade modular
- Limitação : não existe uma forma simples de criar conexão forte entre dados e funções

sequence.h

```
typedef struct sequence *Sequence;

Sequence seq_create(int init);
Sequence seq_create_step(int init, int step);
void seq_destroy(Sequence);
int seq_next(Sequence);
```

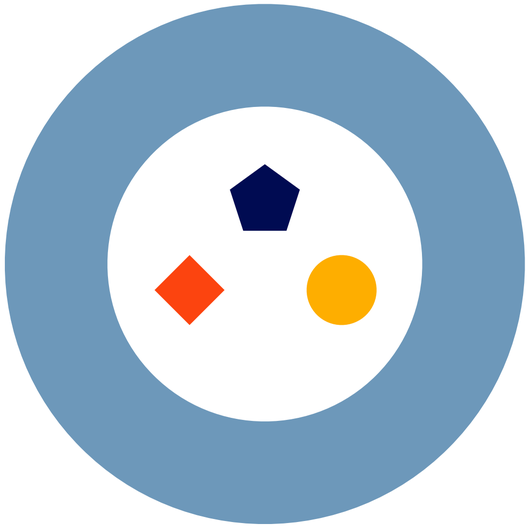
sequence.c

```
struct sequence {
  int next;
  int step;
}
```

Orientação a Objetos e modularização

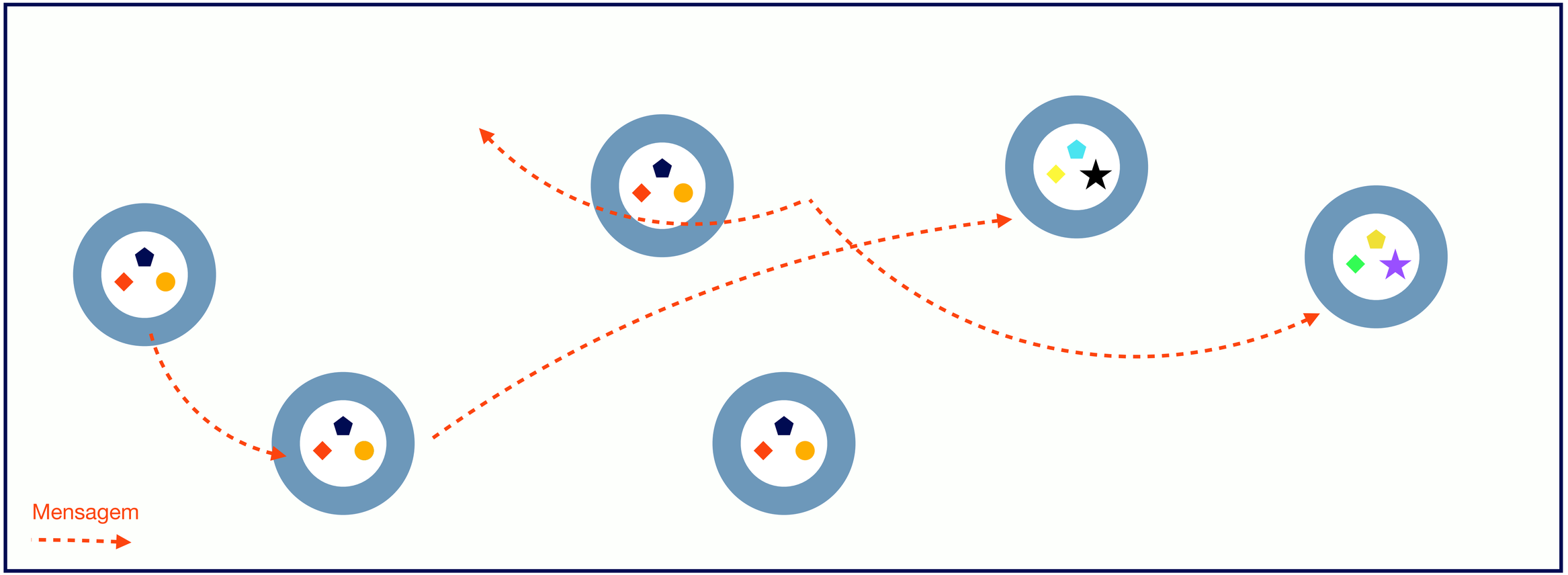
Programação Orientada a Objetos

- Introduz o conceito de objeto como unidade modular
- Objeto é composto:
  - Estados/Dados
  - Comportamento/Ações



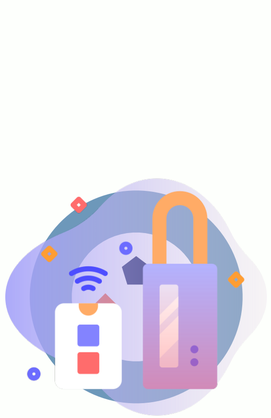
Orientação a Objetos e modularização

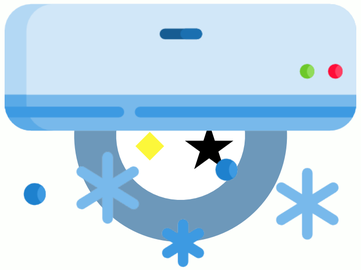
Software orientado a objetos

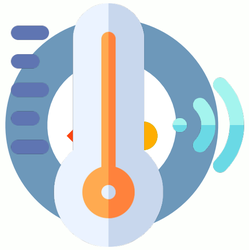


Orientação a Objetos e modularização

Software orientado a objetos







Orientação a Objetos e modularização

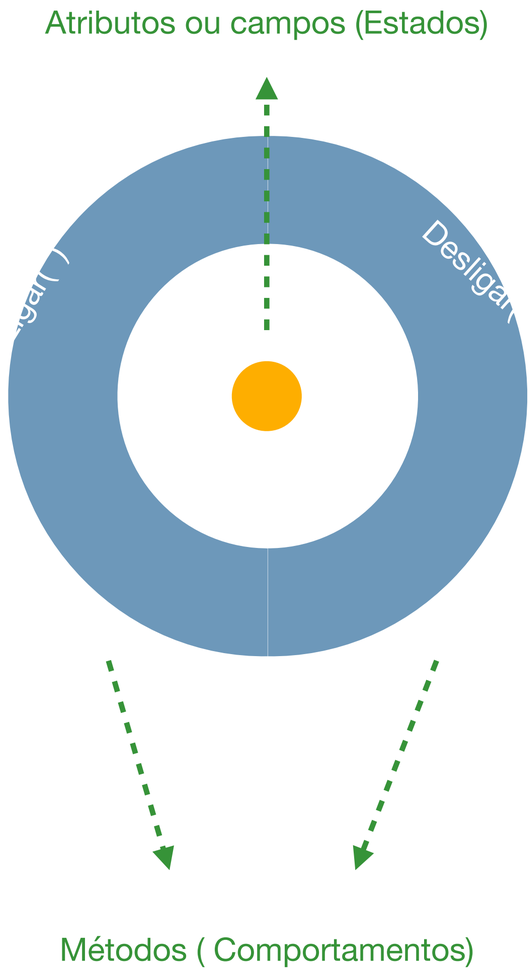
- Estados / Dados :
  - Ligada
  - Desligada
- Comportamento / Ações:
  - Ligar
  - Desligar

Orientação a Objetos e modularização

Mudanças de estado

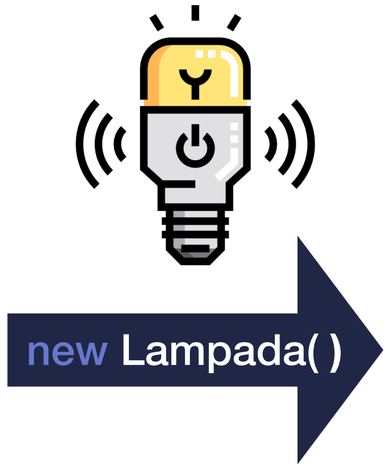
Ciclo de vida do objeto

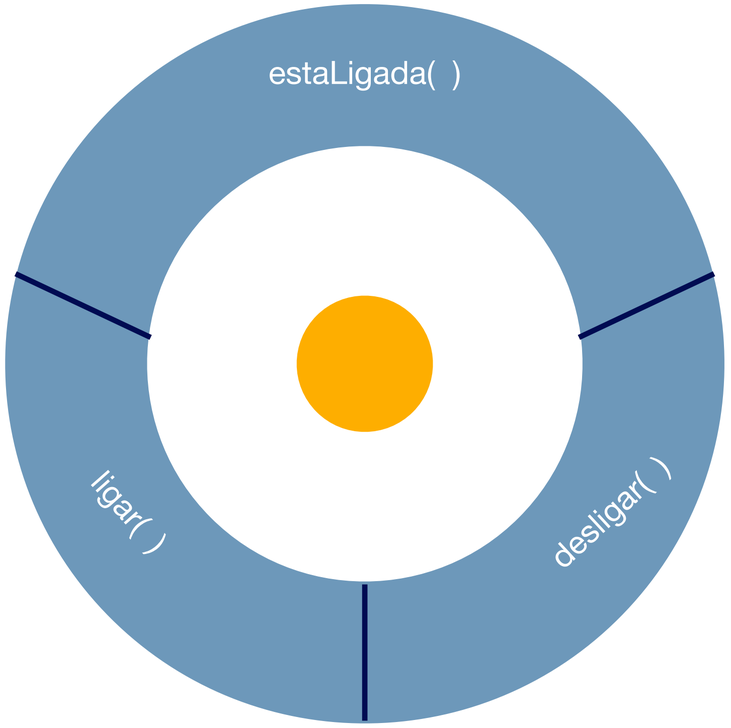
Orientação a Objetos e modularização



Orientação a Objetos e modularização

```
class Lampada {
  boolean ligada;

  public Lampada() {
    ligada = true;
  }

  public void ligar() {
    ligada = true;
  }

  public void desligar() {
    ligada = false;
  }

  public boolean estaLigada() {
    return ligada;
  }
}
```





Orientação a Objetos e modularização

Vantagens da programação orientada a objetos

- Modularidade intrínseca
- Incentiva o raciocínio modular
- Ocultamento de informação
- Reusabilidade de código
- Facilidade em “plugar” e depurar

Reflexões para a próxima aula :

1. Como você implementaria uma **calculadora** com a funcionalidade de **memória : mc , m+, m- e mr**
2. Como você implementaria essa mesma **calculadora** usando **POO (Java)**

Por hoje é só