

O professor começou a aula reforçando a importância da leitura dos diários de bordos (relatos) entregues semanalmente. *"Esteja atento aos relatos" - Lucio Geronimo Valentim*

## 1. CALCULADORA

Ao decorrer da aula, o professor nos instruiu a fim de obter um melhor resultado quando estamos modelando e codificando.

### DICAS PARA DIAGRAMA UML CALCULADORA

- Devemos abstrair elementos da calculadora em objetos
- Devemos atribuir corretamente as responsabilidades de cada classe e atributo
- Em nosso display, devemos ter a menor quantidade de funções possíveis que outros objetos podem utilizar para se comunicar com o mesmo.
- Analisar que são os teclados que mandam mensagens para a cpu, e não o contrario.

### DICAS PARA CÓDIGO CALCULADORA

- Devemos escolher nome de variáveis a fim de melhorar a legibilidade de nosso código.
- Utilizar try catch a fim de capturar e tratar erros que podem ocorrer durante a execução do algoritmo.
- Somente utilizar números inteiros, desconsiderar números do tipo float, com objetivo de simplificar o algoritmo (pois focaremos no quesito orientação a objetos).

## 2. ORIENTAÇÃO A OBJETOS

Quando estamos trabalhando com o paradigma orientado a objetos, devemos seguir o princípio de modularidade, para isso utilizamos alguns critérios:

- **Decomposição:** decompor nosso problema em subproblemas.
- **Composição:** utilizar pequenas partes (subproblemas) a fim de obter a solução.
- **Entendimento:** os módulos de nosso programa devem ser coerentes ao problema.
- **Continuidade:** não precisamos propagar muitas alterações em nosso software.
- **Proteção:** ao ocorrer problemas em um módulo, esse problema não se propaga em grande escala.

Seguimos também alguns princípios:

- **Linguística Modular:** devemos respeitar as unidades sintáticas que a linguagem utilizada suporta (C/C++ suporta ".h e .cpp"; Java suporta pacotes).
- **Poucas Interfaces:** devemos ter uma quantidade limitada de canais de comunicações.
- **Pequenas Interfaces:** a troca de informação dever ser a mínima possível
- **Interfaces explícitas:** deve ser fácil perceber quem se comunica com quem.

Interface é como as coisas se ligam e se comunicam. Exemplo: *"para uma garrafa plastica e tampinha se comunicar, utilizam a interface rosca", "em um cabo usb, para as duas entradas se camunicam, via canais".*

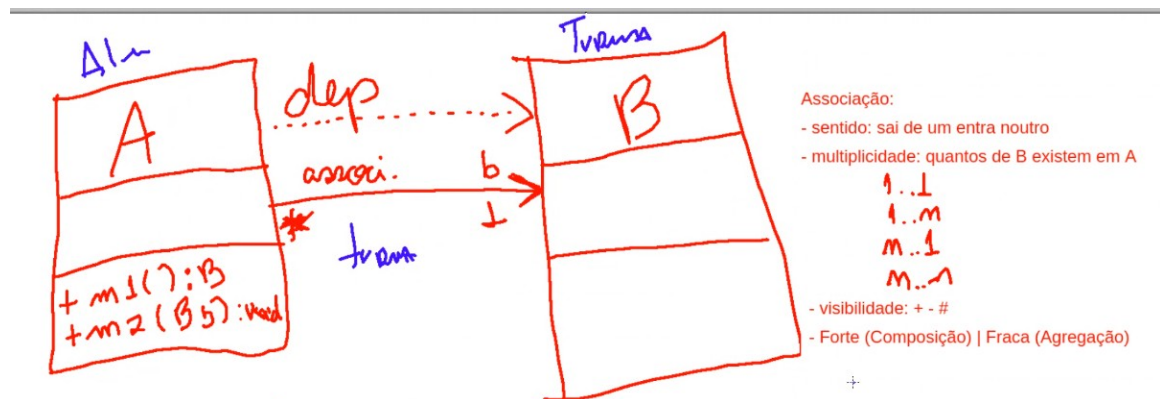
*Em C++ um conjunto de métodos é oque cria uma interface.*

## Associação

Considere uma classe **A** e uma classe **B**.

- **Dependencia:** Quando a classe A depende de B em algum de seus métodos. Caso mudarmos algo em B, esta mudança poderia afetar o comportamento ou estado de A.
- **Associação:** Caso A tenha algum atributo do tipo B, ou seja, define se objetos de uma classe são conectados com objetos de outra classe.
- **Agregação:** Significa que um objeto contém o outro. Desta forma, um pode existir independente do outro.
- **Composição:** Temos uma composição de B em A, quando B está em A. Porém, a classe B so existe enquanto A existir.

Veja na Figura 1 abaixo uma imagem que exemplifica tipos de associações.

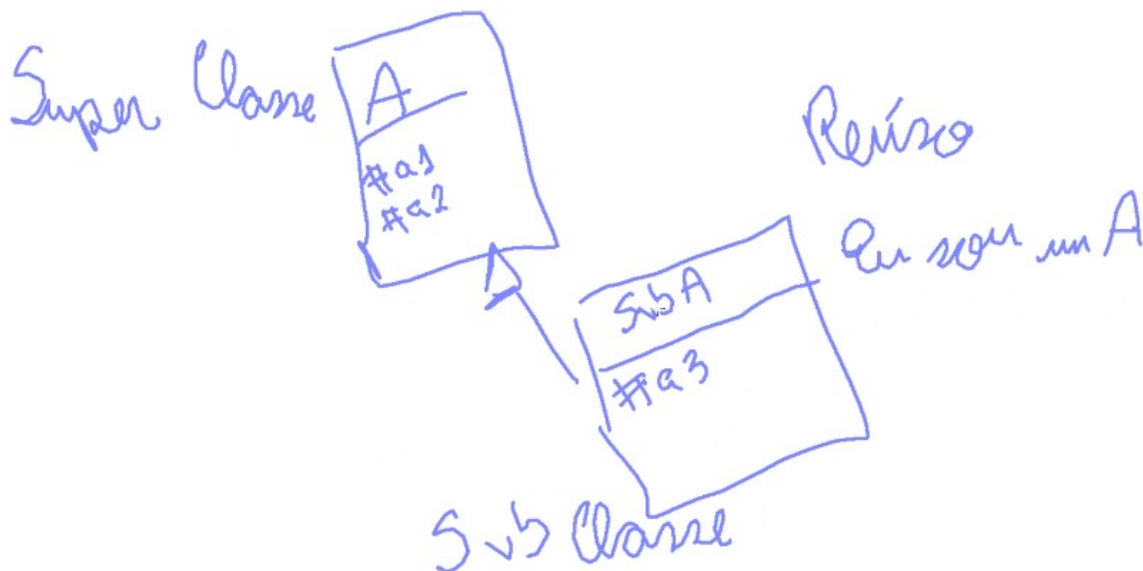


**Figura 1.** Exemplos de associações.

**Fonte:** Professor Dr. Lucio Geronimo Valentin

## Herança

A explicação para herança foi dada considerando uma classe e subclasse, onde a subclasse (classe que herda da superclasse) herda atributos e métodos de sua superclasse (classe de quem se herda). Veja na Figura 2 abaixo uma ilustração representativa.



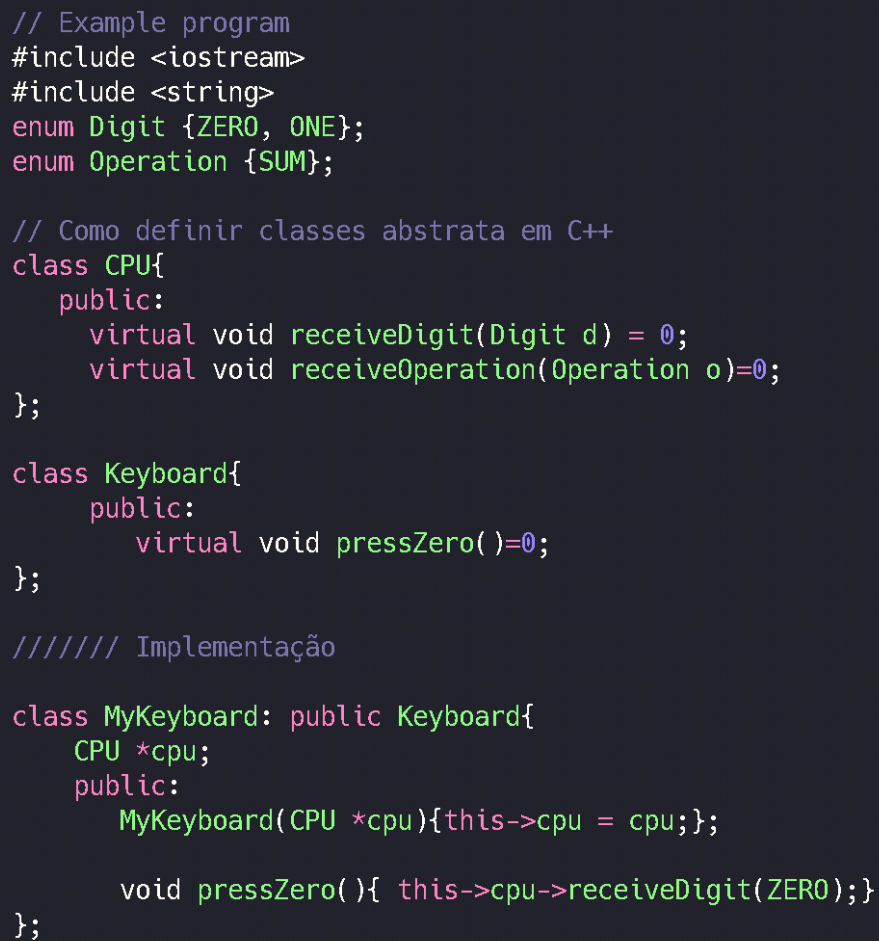
**Figura 2.** Ilustração representativa sobre herança.

**Fonte:** Professor Dr. Lucio Geronimo Valentin

## 3. ATIVIDADES ASSINCRONAS

1. Resenhar capítulo 3 e 5 - Livro Clinio Borges  
([https://moodle.utfpr.edu.br/pluginfile.php/902601/mod\\_resource/content/0/CppBorgesClinio.pdf](https://moodle.utfpr.edu.br/pluginfile.php/902601/mod_resource/content/0/CppBorgesClinio.pdf))
2. Implementar em dupla uma interface em calc.h modularizada e simplificada.

Veja na Figura 3 abaixo o código exemplo do professor disponibilizado durante o encontro síncrono.



```
// Example program
#include <iostream>
#include <string>
enum Digit {ZERO, ONE};
enum Operation {SUM};

// Como definir classes abstrata em C++
class CPU{
public:
    virtual void receiveDigit(Digit d) = 0;
    virtual void receiveOperation(Operation o)=0;
};

class Keyboard{
public:
    virtual void pressZero()=0;
};

//////// Implementação

class MyKeyboard: public Keyboard{
    CPU *cpu;
public:
    MyKeyboard(CPU *cpu){this->cpu = cpu;};

    void pressZero(){ this->cpu->receiveDigit(ZERO);}
};
```

**Figura 3.** Código interface CPU e KeyBoard.

**Fonte:** Professor Dr. Lucio Geronimo Valentin