Funções Estáticas e Dependências

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Material

- Baixe o material disponibilizado
  - “Material para antes da aula”
- Entenda o funcionamento da classe Console criada

Relembrando

- O que é um **dado estático**?

Relembrando

- O que é um **dado estático**?
  - Um dado que pertence a classe, e não aos objetos individuais
    - É “compartilhado” entre todos os objetos instanciados da classe

Função estática

- Da mesma forma, uma função estática **pertence a classe**, e não aos objetos
  - Pode ser acessada diretamente através da classe
  - Não precisa de um objeto instanciado para se ter acesso

Classe Console

- Note que a função membro *imprimirDados* da classe Console não acessa nenhum dado membro da classe Console
  - Essa é uma função utilitária da classe
- Nesse caso, faz sentido transformar essa função membro em estática
  - Não precisaremos instanciar objetos para usar a função

Funções estáticas

- Basta adicionar o modificador *static* no protótipo da função no .hpp
- O cpp não é alterado
- Para acessar uma função estática, utilizamos o operador :: , da mesma forma que fizemos para os dados estáticos

Console.hpp

```
#ifndef CONSOLE_HPP
#define CONSOLE_HPP

#include "Disciplina.hpp"

class Console{
	public:
		static void imprimirDadosDisciplina(Disciplina& disciplina);

};
#endif
```

Main.cpp

```cpp
#include <iostream>

#include "Pessoa.hpp"
#include "Disciplina.hpp"
#include "Console.hpp"

int main(){
	Pessoa prof1{"João", 40};
	Disciplina dis1{"C++"};
	dis1.setProfessor(&prof1);

	dis1.adicionarConteudoMinistrado("Ponteiros", 4);
	dis1.adicionarConteudoMinistrado("Referencias", 2);

	Console::imprimirDadosDisciplina(dis1);

	return 0;
}
```

Podemos fazer isso?

Console.hpp

```
#ifndef CONSOLE_HPP
#define CONSOLE_HPP

#include <string>

#include "Disciplina.hpp"

class Console{
        public:
                static void imprimirDadosDisciplina(Disciplina& disciplina);
                //...
        private:
                std::string cabecalho;

};
#endif
```

Console.cpp

```
#include "Console.hpp"

#include<iostream>

void Console::imprimirDadosDisciplina(Disciplina& disciplina){
        std::cout << cabecalho << std::endl;


        std::cout << "Disicplina: "<< disciplina.getNome() << std::endl;
        //...
}
```

UDESC

Podemos fazer isso?

Console.hpp

```
#ifndef CONSOLE_HPP
#define CONSOLE_HPP

#include <string>

#include "Disciplina.hpp"

class Console{
        public:
                static void imprimirDadosDisciplina(Disciplina& disciplina);
                //...
        private:
                std::string cabecalho;

};
#endif
```

Console.cpp

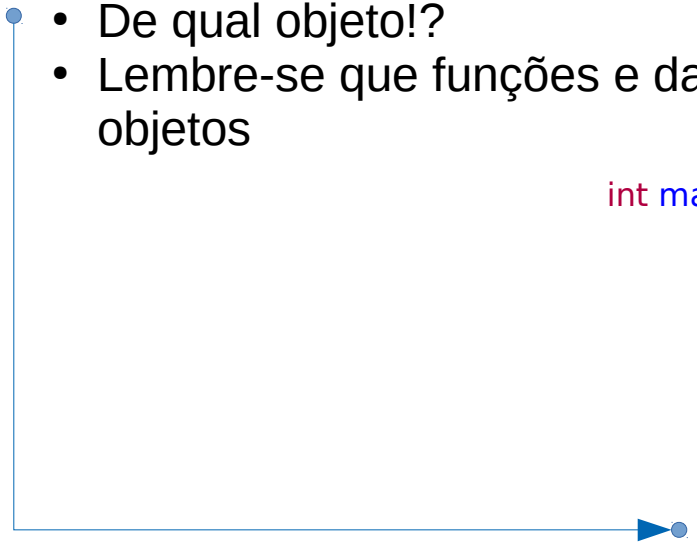
```
#include "Console.hpp"

#include<iostream>

void Console::imprimirDadosDisciplina(Disciplina& disciplina){
        std::cout << cabecalho << std::endl;


        std::cout << "Disicplina: "<< disciplina.getNome() << std::endl;
        //...
}
```

Uma função estática não pode acessar um dado membro não estático!!!

UDESC

Funções Estáticas

- Uma função estática não pode acessar um dado membro não estático!!!
  - Estamos tentando acessar o dado membro de um objeto
    - De qual objeto!?
    - Lembre-se que funções e dados estáticos pertencem a classe, e não aos objetos



```
int main(){
        Pessoa prof1{"João", 40};
        Disciplina dis1{"C++"};
        dis1.setProfessor(&prof1);


        dis1.adicionarConteudoMinistrado("Ponteiros", 4);
        dis1.adicionarConteudoMinistrado("Referencias", 2);


        Console::imprimirDadosDisciplina(dis1);


        return 0;
}
```

Funções Estáticas

- Funções estáticas **podem** acessar dados estáticos da classe

Funções Estáticas

- O ponteiro ***this*** **não existe** para uma função estática
  - *This* aponta para o objeto atual, mas a função estática não pertence a nenhum objeto

Dependências

- Note que a classe Console não possui nenhum dado membro disciplina
- Mas a classe Disciplina é **necessária** para se executar a sua função membro imprimirDadosDisciplina
- Isso se trata de uma **Dependência**
- Em uma dependência, uma classe depende da outra para executar uma tarefa
  - Mas não possui nenhuma referência interna (no formato de um dado membro) ligando-a a essa classe

Dependência UML

- No diagrama de Classes UML, a dependência é anotada como uma seta tracejada

Exercícios

1. Pesquise mais sobre o relacionamento de dependência
   - Alguns links interessantes
     - https://www.ibm.com/support/knowledgecenter/pt-br/SSCLKU_7.5.5/com.ibm.xtools.modeler.doc/topics/cdepend.html
     - Seção 10.5 do livro “Pressman, R.;Maxim, B. Engenharia de Software: uma abordagem Profissional. McGraw Hill Brasil, 2016. 8 ed. ISBN 9788580555349.”
       - A seção está disponível gratuitamente em books.google.com
2. Atualize o diagrama de classes UML do sistema
   - Adicione a classe Console e seu relacionamento de dependência com Disciplina
   - Não esqueça de marcar a função imprimirDadosDisciplina como estática
     - Assim como dados estáticos, as funções estáticas são marcadas com sublinhado no diagrama de classe

Referências

- DEITEL, P.; DEITEL, H. **C++ how to Program**. [S.l.]: Pearson, 2017. ISBN 9780134448237
- STROUSTRUP, B. **The C++ Programming Language**. Pearson Education, 2013. ISBN 9780133522853.