Desenvolvimento de Sistemas Orientados a Objetos I

Estruturando o Sistema Orientado a Objetos
PARTE - II

Jean Carlo Rossa Hauck, Dr.

jean.hauck@ufsc.br

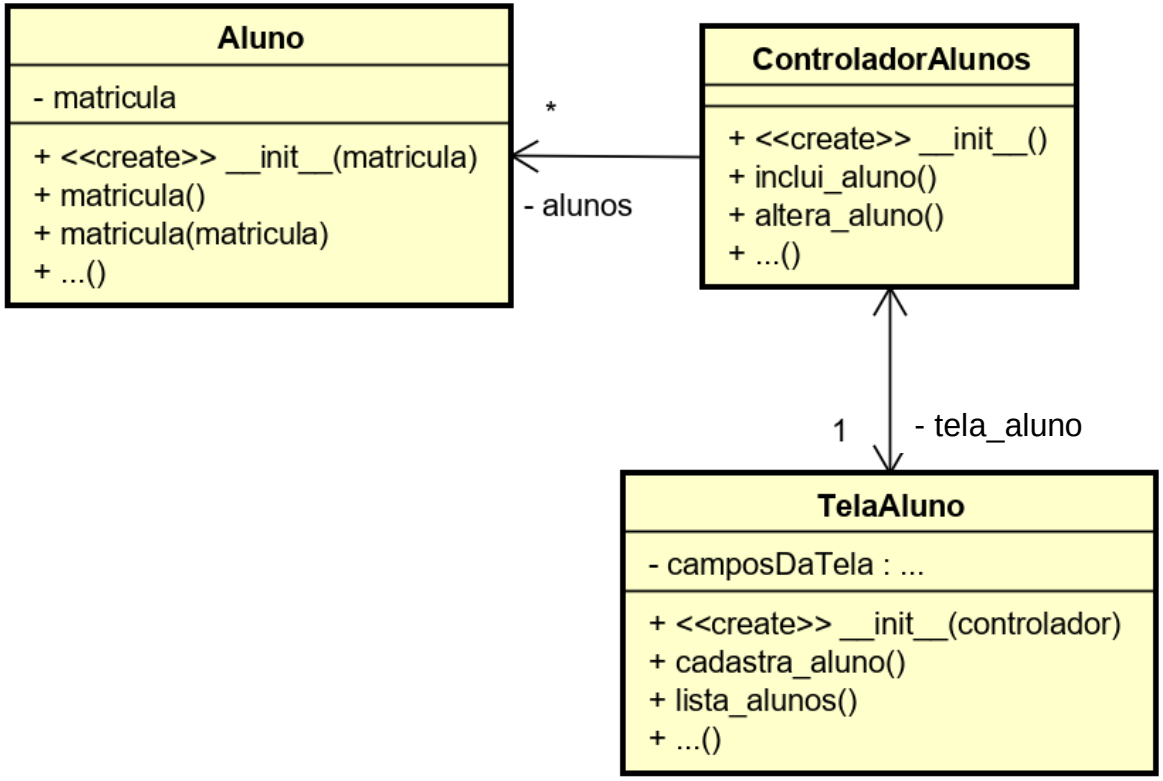
http://www.inf.ufsc.br/~jeanhauck

Conteúdo Programático

- Conceitos e mecanismos da programação orientada a objetos
- Práticas de Desenvolvimento de Software
  - Arquitetura em camadas e padrões de projeto

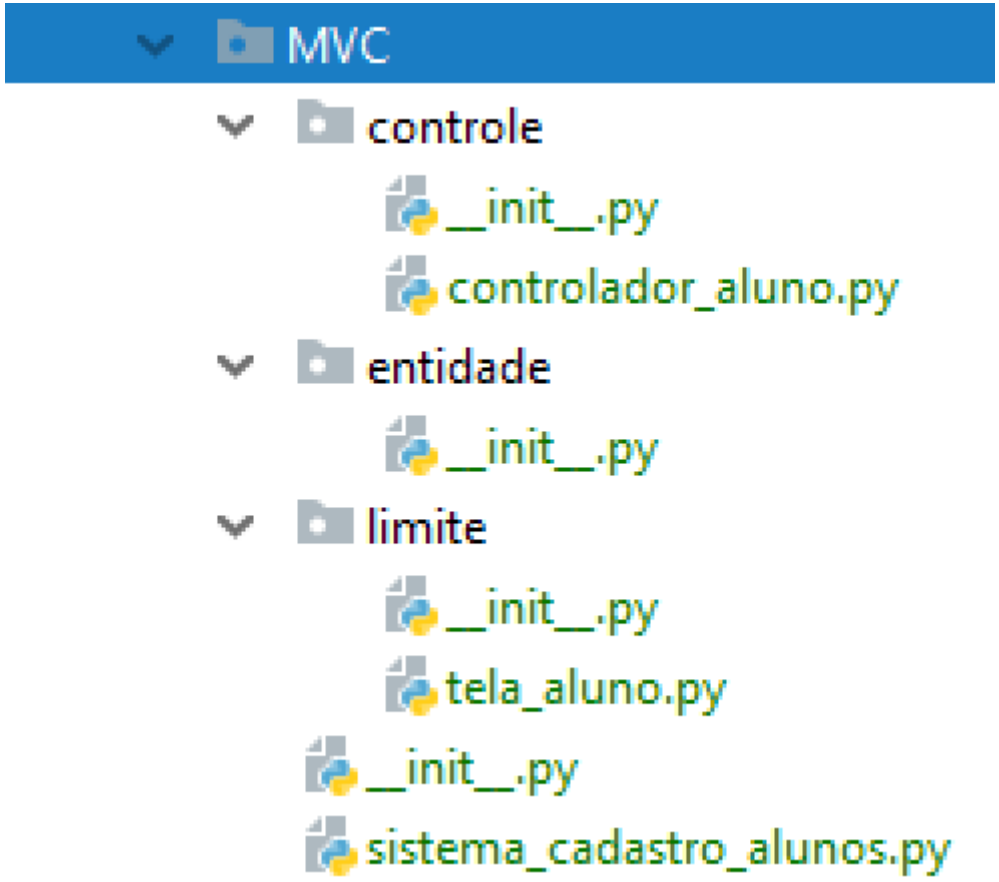
Então ... na prática

- Para cada Entidade, tipicamente mais duas classes:



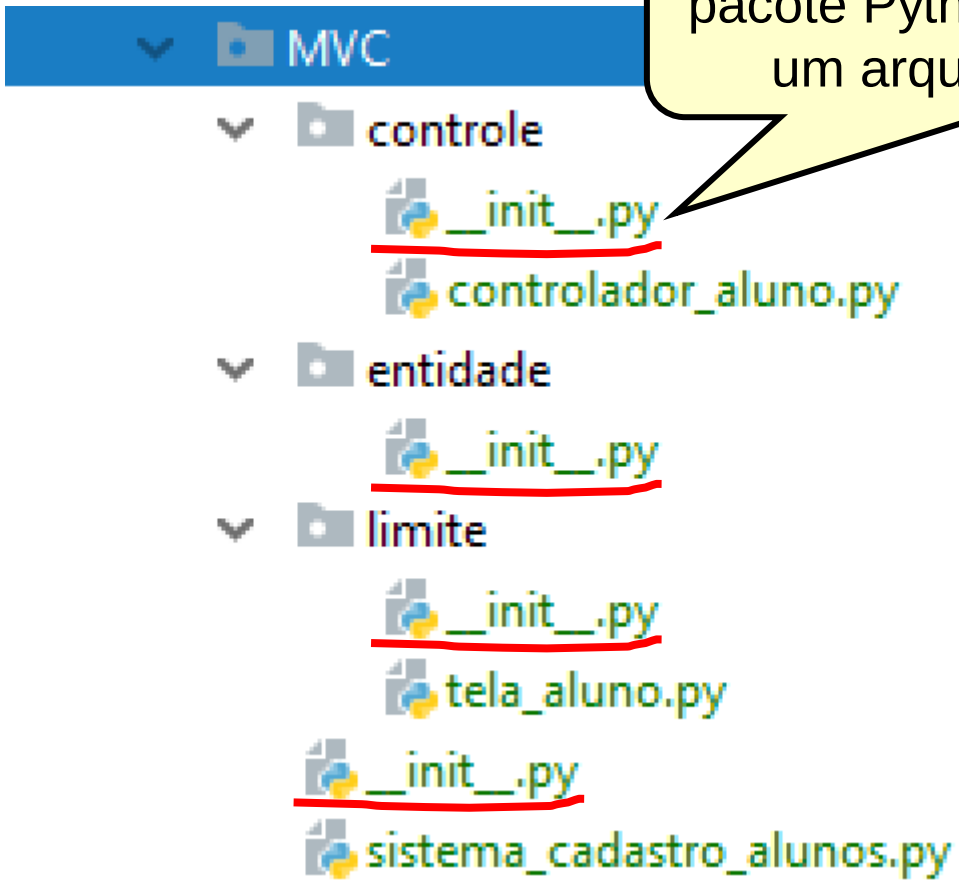
Então ... na prática

- Sugestão de organização dos pacotes:



Então ... na prática

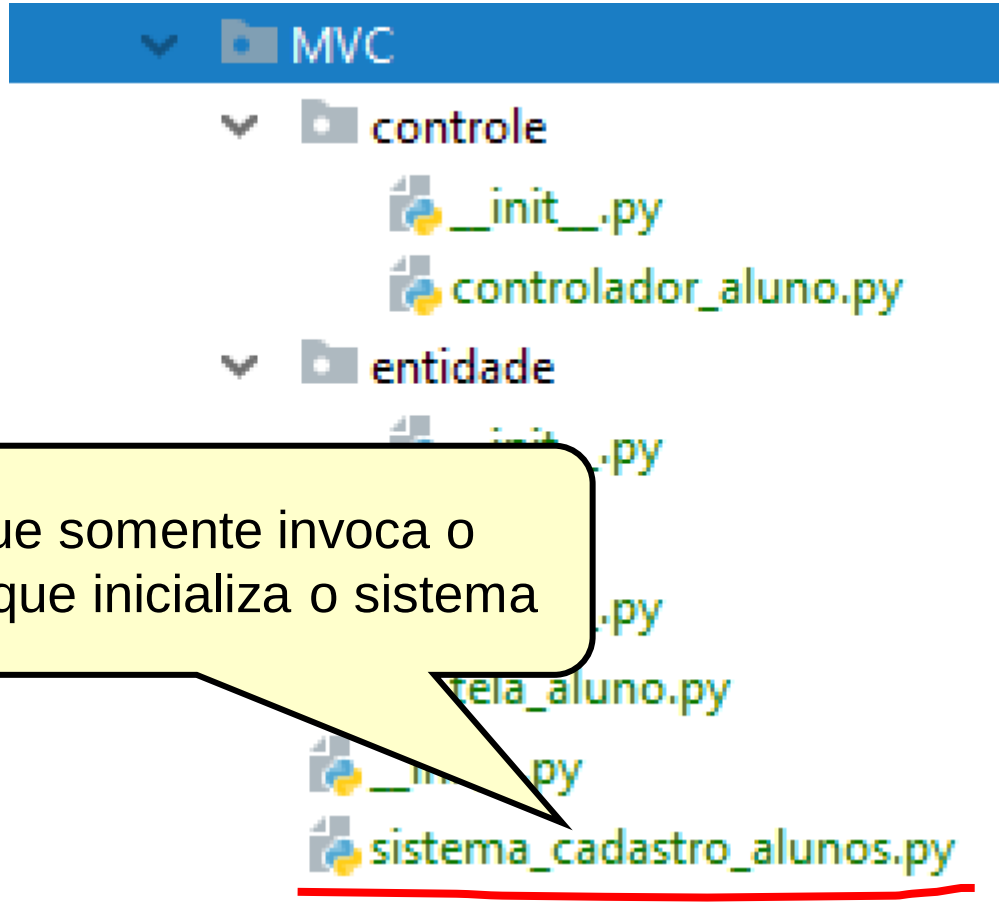
- Sugestão de organização dos pa



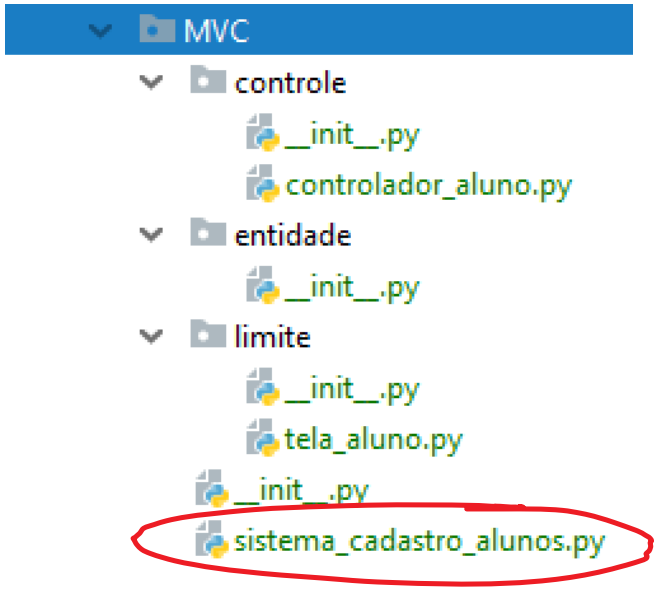
Para que uma pasta seja um pacote Python ela deve conter um arquivo **__init__.py**

Então ... na prática

- Sugestão de organização dos pacotes:

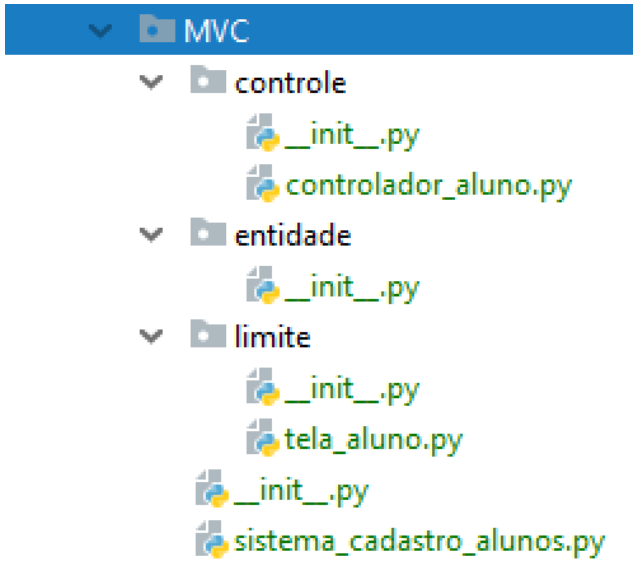


Arquivo de inicialização




```python
from controle.controlador_aluno import ControladorAluno

if __name__ == "__main__":
    ControladorAluno().inicia()
```

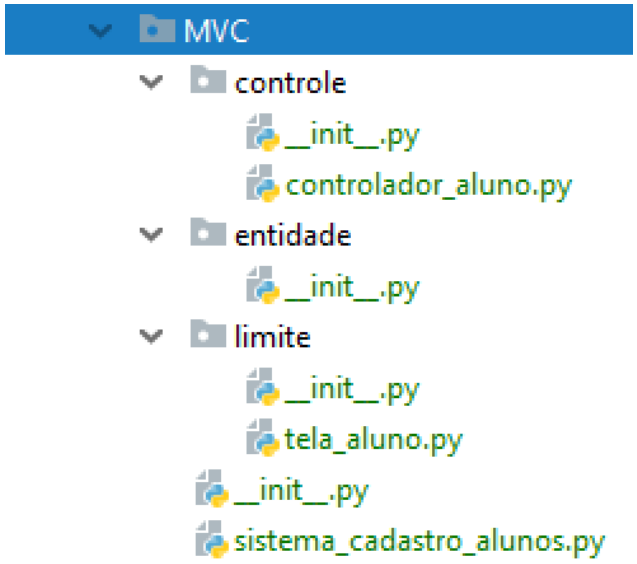
Arquivo de inicialização




```
from controle.controlador_aluno import ControladorAluno

if __name__ == "__main__":
    ControladorAluno().inicia()
```

Instancia o controlador

Arquivo de inicialização

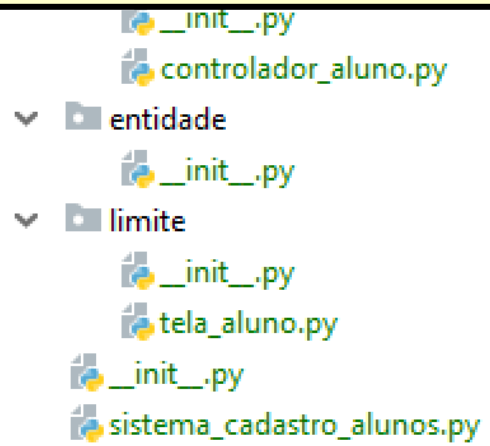



```
from controle.controlador_aluno import ControladorAluno

if __name__ == "__main__":
    ControladorAluno().inicia()
```

Invoca operação que inicia o programa

Arquivo de inicialização

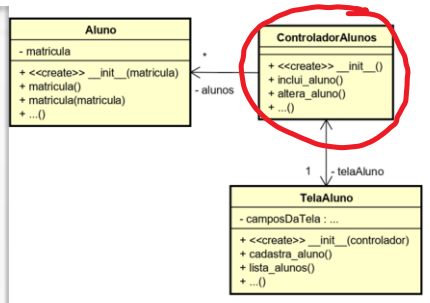
Testa se o arquivo está sendo executado como um programa principal




```
from ce      e.controlador_aluno import ControladorAluno

if __name__ == "__main__":
    ControladorAluno().inicia()
```

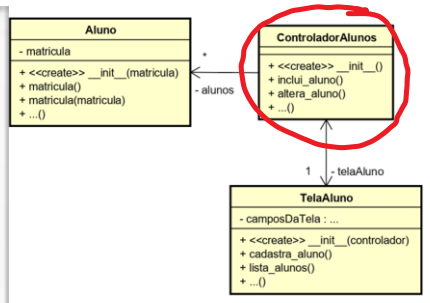
Código do Controlador

```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```



Código do Controlador

```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```

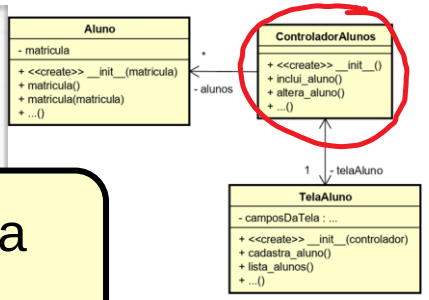


Construtor instancia TelaAluno e passa **ele mesmo** como parâmetro

Código do Controlador

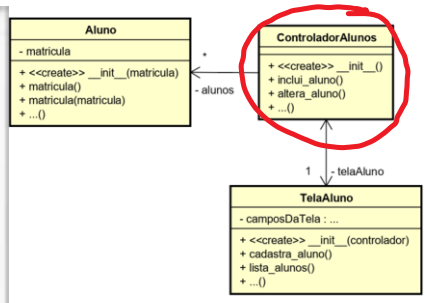
```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```

Método inicia() manda abrir a tela inicial



Código do Controlador

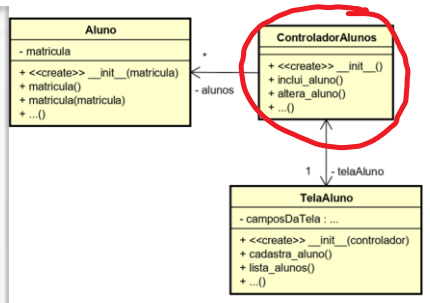
```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```



Diversos métodos normais para Incluir, Alterar, Excluir, Listar, etc.

Código do Controlador

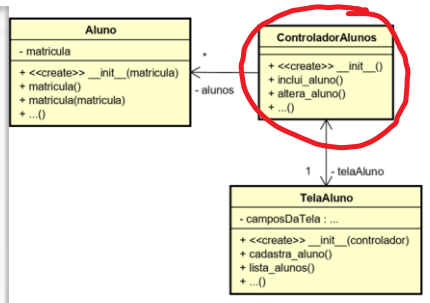
```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alun
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```



Invoca a tela_aluno para solicitar escolha de opções pelo usuário

Código do Controlador

```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```

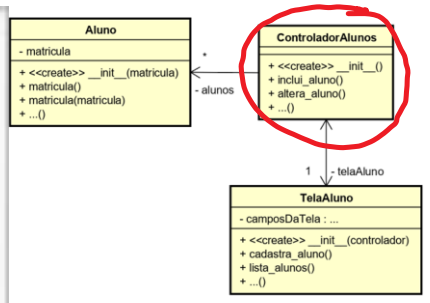


{chave: valor}

chave: opção escolhida
valor: método a ser executado

Código do Controlador

```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```

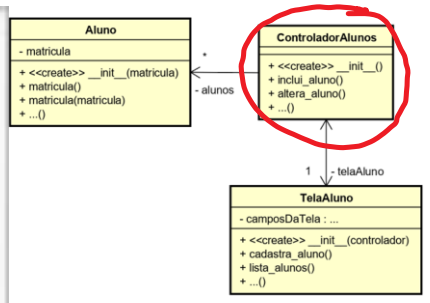


{chave: valor}

chave: opção escolhida
valor: método a ser executado

Código do Controlador

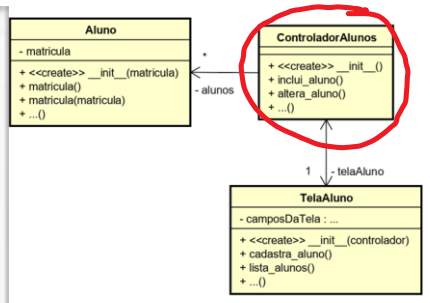
```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: s
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```



Manda a tela solicitar a opção do usuário

Código do Controlador

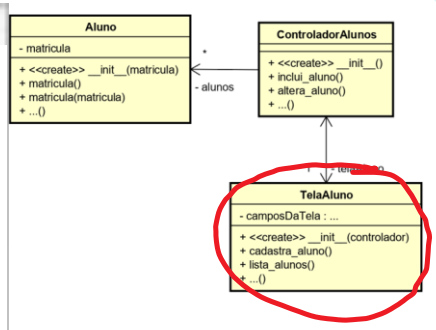
```
import ...


class ControladorAluno:

    def __init__(self):
        self.__tela_aluno = TelaAluno(self)
        self.__alunos = []

    def inicia(self):
        self.abre_tela_inicial()

    def inclui_aluno(self):...

    def altera_aluno(self):...

    def exclui_aluno(self):...

    def lista_aluno(self):...

    def finalizar(self):
        # manda a tela_aluno mostrar a lista dos alunos
        exit(0)

    def abre_tela_inicial(self):
        switcher = {0: self.finalizar, 1: self.inclui_aluno, 2: self.altera_aluno,
                    3: self.exclui_aluno, 4: self.lista_aluno}
        while True:
            opcao = self.__tela_aluno.mostra_tela_opcoes()
            funcao_escolhida = switcher[opcao]
            funcao_escolhida()
```



De acordo com a opção escolhida, executa o método desejado

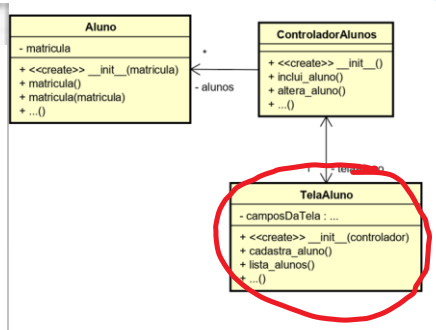
Código da Tela

```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(mensagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numerico inteiro valido")
                if inteiros_validos:
                    print("Valores validos: ", inteiros_validos)

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```



Código da Tela

```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(mensagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor
                if inteiros_validos:
                    print("Valores validos: ", inteiros

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```

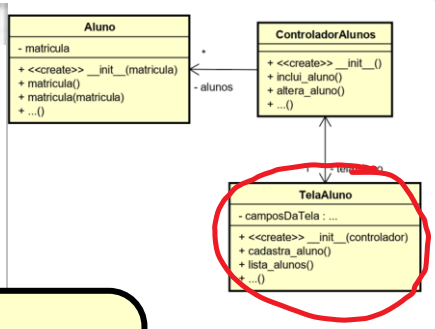


Método invocado pelo controlador, que retorna a opção escolhida pelo usuário

Código da Tela

```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(me
            try:
                inteiro = int(valor_lid
                if inteiros_validos and
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numerico inteiro valido")
                if inteiros_validos:
                    print("Valores validos: ", inteiros_validos)

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```

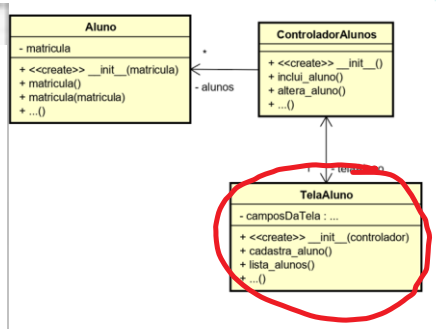
Método para tratar a entrada de dados



Código da Tela

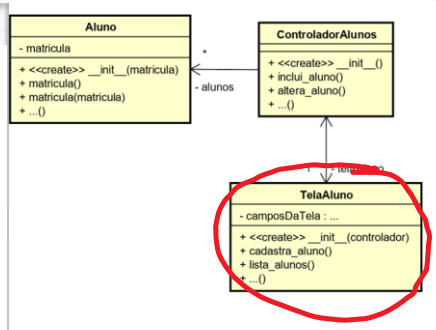
```
class TelaAluno:

    def __init__(self, con
        self.__controlador

    def le_num_inteiro(sel
        while True:
            valor_lido = in                    sagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numerico inteiro valido")
                if inteiros_validos:
                    print("Valores validos: ", inteiros_validos)

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```

Tratamento da exceção gerada pelo int(valor_lido)



Código da Tela

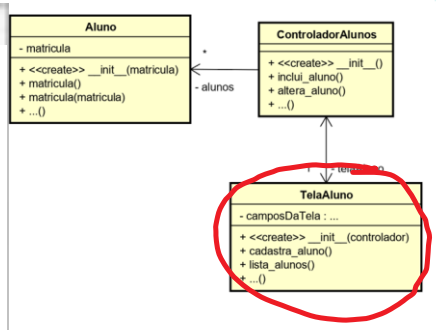
```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(mensagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numer
                if inteiros_validos:
                    print("Valores validos: ", inteiros_valid

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```



Também é possível disparar uma Exceção intencionalmente

Código da Tela

```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(mensagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numerico inteiro valido")
                if inteiros_validos:
                    print("Valores validos: ", inteiros_validos)

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4,
        return opcao
```

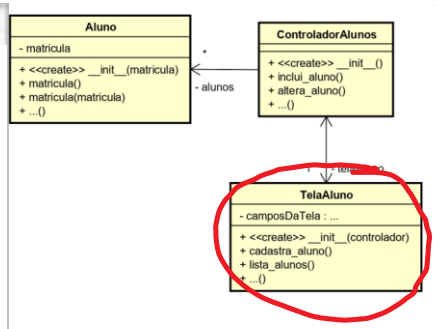


E tratar a exceção
(para os dois casos)

Será estudado em breve!

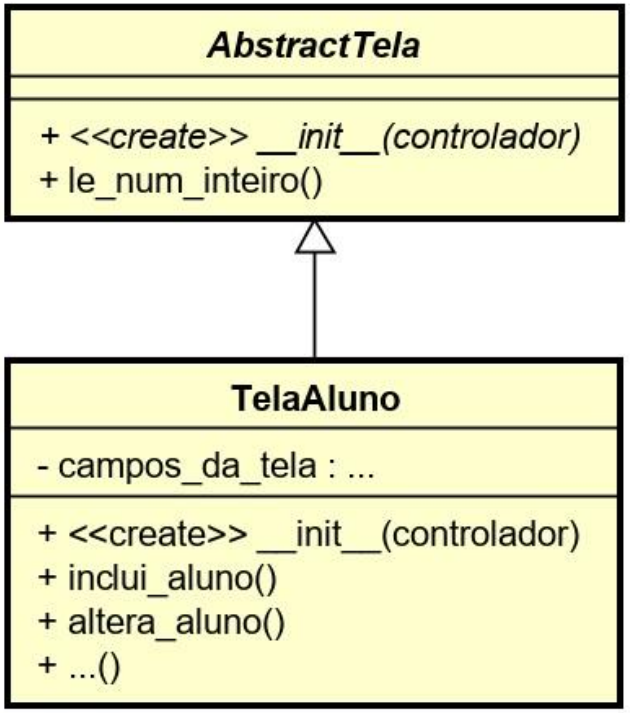
Código da Tela

```
class TelaAluno:

    def __init__(self, controlador):
        self.__controlador = controlador

    def le_num_inteiro(self, mensagem: str = "", inteiros_validos: [] = None):
        while True:
            valor_lido = input(mensagem)
            try:
                inteiro = int(valor_lido)
                if inteiros_validos and inteiro not in inteiros_validos:
                    raise ValueError
                return inteiro
            except ValueError:
                print("Valor incorreto: Digite um valor numerico inteiro valido")
                if inteiros_validos:
                    print("Valores validos: ", inteiros_validos)

    def mostra_tela_opcoes(self):
        print("--------- CADASTRO ALUNOS ---------")
        print("1 - Incluir")
        print("2 - Alterar")
        print("3 - Excluir")
        print("4 - Listar")
        print("0 - Voltar")
        opcao = self.le_num_inteiro("Escolha a opcao: ", [1, 2, 3, 4, 0])
        return opcao
```



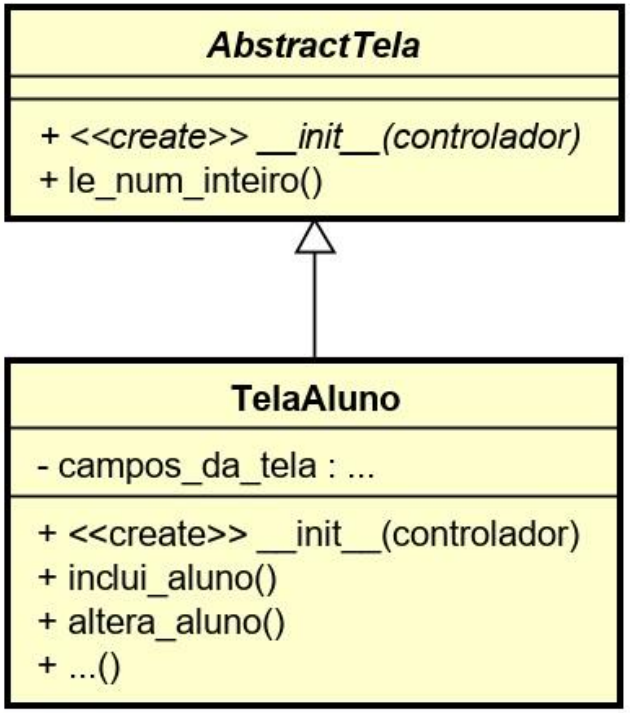
Este código será utilizado por outras telas?

Herança utilizada para Telas



Que tal então uma hierarquia de telas?

Herança utilizada para Telas



Que tal então uma hierarquia de telas?

Pode ser interessante também nos Controladores que tenham características comuns entre eles!

Licença

Atribuição-Uso-Não-Comercial-Compartilhamento pela Licença 2.5 Brasil

Você pode:
- copiar, distribuir, exibir e executar a obra
- criar obras derivadas

Sob as seguintes condições:
Atribuição — Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.
Uso Não-Comercial — Você não pode utilizar esta obra com finalidades comerciais.
Compartilhamento pela mesma Licença — Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.

Para ver uma cópia desta licença, visite http://creativecommons.org/licenses/by-nc-sa/2.5/br/ ou mande uma carta para Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.