Notas de aula de programação em Python
Curso avançado
Topico 1 - Classes

Prof. Louis Augusto
louis.augusto@ifsc.edu.br

Instituto Federal de Santa Catarina
São José

Índice

Sumário

Classes como modelo de objetos

Funções são usadas principalmente quando há partes de códigos que precisam ser repetidas ou reutilizadas com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são utilizadas quando há blocos de funções e de variáveis que se repetem com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são isto, uma sequência de variáveis e funções que são coordenadas entre si, e que fazem sentido serem isoladas de outras partes do código.

Criando uma classe transferimos funcionalidades e variáveis repetitivas para um bloco encapsulado de código. Como estas funções e variáveis fazem sentido entre si este conjunto se chama **objeto**, e a forma com que o Python trabalha com o objeto é uma estrutura de dados que foi chamada de **classe**.

Classes como modelo de objetos

Funções são usadas principalmente quando há partes de códigos que precisam ser repetidas ou reutilizadas com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são utilizadas quando há blocos de funções e de variáveis que se repetem com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são isto, uma sequência de variáveis e funções que são coordenadas entre si, e que fazem sentido serem isoladas de outras partes do código.

Criando uma classe transferimos funcionalidades e variáveis repetitivas para um bloco encapsulado de código. Como estas funções e variáveis fazem sentido entre si este conjunto se chama **objeto**, e a forma com que o Python trabalha com o objeto é uma estrutura de dados que foi chamada de **classe**.

Classes como modelo de objetos

Funções são usadas principalmente quando há partes de códigos que precisam ser repetidas ou reutilizadas com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são utilizadas quando há blocos de funções e de variáveis que se repetem com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são isto, uma sequência de variáveis e funções que são coordenadas entre si, e que fazem sentido serem isoladas de outras partes do código.

Criando uma classe transferimos funcionalidades e variáveis repetitivas para um bloco encapsulado de código. Como estas funções e variáveis fazem sentido entre si este conjunto se chama **objeto**, e a forma com que o Python trabalha com o objeto é uma estrutura de dados que foi chamada de **classe**.

Classes como modelo de objetos

Funções são usadas principalmente quando há partes de códigos que precisam ser repetidas ou reutilizadas com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são utilizadas quando há blocos de funções e de variáveis que se repetem com frequência, e evitam que se tenha de ficar refazendo o código.

Classes são isto, uma sequência de variáveis e funções que são coordenadas entre si, e que fazem sentido serem isoladas de outras partes do código.

Criando uma classe transferimos funcionalidades e variáveis repetitivas para um bloco encapsulado de código. Como estas funções e variáveis fazem sentido entre si este conjunto se chama **objeto**, e a forma com que o Python trabalha com o objeto é uma estrutura de dados que foi chamada de **classe**.

Sumário

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

- Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.
- Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Classes como modelo de objetos

Um objeto na vida real costuma guardar duas características (já traduzido para a linguagem usada para classes):

Atributos: são as distinções de um objeto em relação a outro, como cor, forma, valor, comprimento etc.

Métodos: são as ações que o objeto pode realizar ou mudança de estado, como estar ligado ou desligado, produzindo algo ou não, mudando algo de lugar, ou qualquer alteração em outro objeto ou em si próprio.

Pode-se imaginar um carro como um grande objeto, que é formado por vários objetos menores que podem trocar informações entre si ou não. Por exemplo, o motor do carro, o limpador de pára-brisa, as portas (que podem estar abertas ou fechadas), etc.

Neste caso teríamos um grande objeto chamado carro e vários objetos dentro do carro que podem ou não trocar informações entre si, dependendo da necessidade. Por exemplo, o limpador de pára-brisa pode não precisar se corresponder com a fechadura do porta malas, mas são objetos do carro.

Modelagem de um objeto

Um bom exemplo seria um objeto computador. Imagine um negócio que monte e venda computadores para uso pessoal e empresarial.

Um exemplo de código que a empresa precise poderia ser:

```
#coding: utf-8
marca = input("Digite a marca do computador")
memoria = input("Digite a quantidade de memória")
placaV = input("Digite a placa de vídeo")

print("Seu computador é da marca: ", marca)
print("Seu computador tem memória: ", memoria)
print("Seu computador tem placa de video: ", placaV)
```

Caso este código tenha de ser repetido uma grande quantidade de vezes, poderíamos encapsulá-lo numa função, ou em duas, se não precisassem ser utilizados em conjunto.

Modelagem de um objeto

Um bom exemplo seria um objeto computador. Imagine um negócio que monte e venda computadores para uso pessoal e empresarial.

Um exemplo de código que a empresa precise poderia ser:

```
#coding: utf-8
marca = input("Digite a marca do computador")
memoria = input("Digite a quantidade de memória")
placaV = input("Digite a placa de vídeo")

print("Seu computador é da marca: ", marca)
print("Seu computador tem memória: ", memoria)
print("Seu computador tem placa de video: ", placaV)
```

Caso este código tenha de ser repetido uma grande quantidade de vezes, poderíamos encapsulá-lo numa função, ou em duas, se não precisassem ser utilizados em conjunto.

Modelagem de um objeto

Um bom exemplo seria um objeto computador. Imagine um negócio que monte e venda computadores para uso pessoal e empresarial.

Um exemplo de código que a empresa precise poderia ser:

```
#coding: utf-8
marca = input("Digite a marca do computador")
memoria = input("Digite a quantidade de memória")
placaV = input("Digite a placa de vídeo")

print("Seu computador é da marca: ", marca)
print("Seu computador tem memória: ", memoria)
print("Seu computador tem placa de video: ", placaV)
```

Caso este código tenha de ser repetido uma grande quantidade de vezes, poderíamos encapsulá-lo numa função, ou em duas, se não precisassem ser utilizados em conjunto.

Modelagem de um objeto

Em uma função:

```
def Info_computador():
    marca = input("Digite a marca do computador: ")
    memoria = input("Digite a quantidade de memória: ")
    placaV = input("Digite a placa de vídeo: ")
    print("Seu computador é da marca: ", marca)
    print("Seu computador tem memória: ", memoria)
    print("Seu computador tem placa de video: ", placaV)
Info_computador()
```

Ou em duas funções:

```
def set_computador():
    marca = input("Digite a marca do computador: ")
    memoria = input("Digite a quantidade de memória: ")
    placaV = input("Digite a placa de vídeo: ")
    return marca, memoria, placaV

def get_computador(marca, memoria, placaV):
    print("Seu computador é da marca: ", marca)
    print("Seu computador tem memória: ", memoria)
    print("Seu computador tem placa de video: ", placaV)

marca, memo, placaV = set_computador()
get_computador(marca, memo, placaV)
```

Criação de um objeto

Uma classe é uma forma elegante de reunir numa só estrutura de dados atributos e métodos e ainda poder instanciar a estrutura numa variável.

Se quisermos **somente** guardar variáveis numa estrutura única utilizamos uma classe sem construtora, da seguinte forma:

```
class Computador:
    marca = ''
    memoria = ''
    placaV = ''
```

e podemos instanciar a classe da seguinte forma:

```
A = Computador()
```

Temos agora um objeto `Computador`, que possui 3 atributos, e que foi instanciado à variável `A`.
Podemos alimentar os atributos usando o operador ponto:

```
A.marca = input("Digite a marca do computador: ")
A.memoria = input("Digite a quantidade de memória: ")
A.placaV = input("Digite a placa de vídeo: ")
```

Podemos resgatar os valores dos atributos usando também o operador ponto.

```
print("Marca do computador: ", A.marca)
```

Modelagem de um objeto

É interessante utilizar classes não somente para gravar atributos. Se quisermos construir métodos (que são funções da classe) devemos utilizar uma função construtora da classe, que tem a palavra-chave `__init__` com o parâmetro `self`:

```
class Computador():
    def __init__(self):
        self.marca = ''
        self.memoria = ''
        self.placaV = ''
    def set_computador(self):
        self.marca = input("Digite a marca do computador: ")
        self.memoria = input("Digite a quantidade de memória: ")
        self.placaV = input("Digite a placa de vídeo: ")
    def get_computador(self):
        print("Marca do computador:", self.marca)
        print("Memória do computador:", self.memoria)
        print("Placa de vídeo do computador:", self.placaV)
A = Computador()
A.set_computador()
A.get_computador()
```

É uma boa prática quando se incluem métodos na classe defini-la com parênteses após o nome da classe, usando `class Computador():` e não somente `class Computador:`.

Modelagem de um objeto

Podemos colocar métodos dentro da função construtora, e neste caso o método será chamado já na inicialização do objeto.
Apesar de não ser uma boa prática de programação, atributos podem ser inicializados em qualquer método.
Observe a classe abaixo:

```
class Computador():
    def __init__(self):
        self.marca = ''
        self.set_computador()
    def set_computador(self):
        self.memoria = ''
        self.placaV = ''
        self.marca = input("Digite a marca do computador: ")
        self.memoria = input("Digite a quantidade de memória: ")
        self.placaV = input("Digite a placa de vídeo: ")
    def get_computador(self):
        print("Marca do computador:", self.marca)
        print("Memória do computador:", self.memoria)
        print("Placa de vídeo do computador:", self.placaV)
A = Computador()
A.get_computador()
```

Modelagem de um objeto

No código anterior, o método `set_computador()` é chamado na construtora, e dentro dele são criados dois atributos, `memoria` e `placaV`.

Apesar de não ser uma boa prática, funciona, e qualquer método chamado depois de `set_computador()` conhecerá estes dois atributos.

Recomenda-se, todavia, definir todos os atributos na construtora.

Depois de definir a classe podemos instanciar quantas variáveis forem necessárias, colocá-las numa lista etc. Usando o código do slide 21:

```
Lista_computadores = []
for _ in range(10):
    A = Computador()
    Lista_computadores.append(A)

Lista_computadores[0].set_computador()
Lista_computadores[0].get_computador()
```

Foram criados 10 objetos `Computador()` mas somente o primeiro teve dados inseridos e retornados.

Sumário

Inserindo parâmetros ao objeto

Para inserção de parâmetros na classe já na quando for instanciada, preenchemos os parâmetros na função `__init__`, após o parâmetro `self`.

```
class Computador():
    def __init__(self, marca, memoria, placaVideo):
        self.marca = marca
        self.memoria = memoria
        self.placaV = placaVideo

    def get_computador(self):
        print("Marca do computador:", self.marca)
        print("Memória do computador:", self.memoria)
        print("Placa de vídeo do computador:", self.placaV)

A = Computador("Pichau", "16gB", "Nvidea")
A.get_computador()
```

Deve-se observar que os atributos foram inseridos já quando o objeto é instanciado. Não há mais sentido o atributo `set_computador()`.

Modelagem de um objeto

Apesar de esquisito, o código abaixo funciona:

```
class Computador():
    def __init__(self, *args, **kwargs):
        self.marca = args[0]
        self.memoria = args[1]
        self.placaV = kwargs['placaV']

    def get_computador(self):
        print("Marca do computador:", self.marca)
        print("Memória do computador:", self.memoria)
        print("Placa de vídeo do computador:", self.placaV)

A = Computador("Pichau", "16gB", placaV = "Nvidia")
A.get_computador()
```

Apesar da funcionalidade questionável, nada há de errado. Podem-se usar argumentos dinâmicos na função construtora normalmente.

Sumário

Inserindo métodos ao objeto

O computador pode executar tarefas, como ligar, desligar, exibir suas configurações etc. Para isto podemos inserir métodos.

Inserimos métodos dentro da classe, na mesma identação da função construtora. Para ser um método devemos inserir o parâmetro `self`, e assim o método pode acessar todos os atributos definidos anteriormente à sua chamada, assim como outros métodos.

```
class Computador():
    def __init__(self, marca, memoria, placaV = "Nvidia"):
        self.marca = marca
        self.memoria = memoria
        self.placaV = placaV
        self.ligado = False
    def Acao_computador(self, acao):
        if acao==1:
            self.liga_comp()
        else:
            self.desliga_comp()
    def liga_comp(self):
        self.ligado = True
    def desliga_comp(self):
        self.ligado = False
    def get_estado(self):
        if self.ligado:
            return "ON"
        else:
            return "OFF"
A = Computador("Pichau", "16gB")
A.Acao_computador(2)
print(A.get_estado())
```

Modelagem de um objeto

Um método acessa outros métodos colocando `self` antes do instanciamento. Um método todavia pode ser acessado de fora da classe, como foi no caso `Acao_computador(2)` e `get_estado()`.

Sumário

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:
- Quando se precisa de uma tarefa específica que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer garantir uma unidade atômica que não vai estar correta para reutilização.
- Quando se utiliza uma função diversas vezes ao longo do código.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:
- Quando se precisa de uma tarefa específica que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer garantir uma funcionalidade que talvez venha a estar contida para reutilização.
- Quando se utiliza uma função diversas vezes.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Quando usar classes

Em cada situação num código é aconselhável utilizar uma estratégia para manter ao máximo possível a coesão e o estado humanizado do código.

É aconselhável que todo código em python se inicie com um driver, que geralmente é uma função que evita que existam códigos soltos no programa. Com isso em mente, é razoável manter em:

Driver: Rotinas simples que raramente mudam e, em geral possuem poucas linhas, e controlam externamente o fluxo do programa.

Funções:

- Quando se precisa de uma tarefa específica e que faz sentido ser isolada para ser facilmente identificada.
- Quando se quer guardar uma funcionalidade que sabe-se estar correta para reutilização.
- Quando se utiliza uma funcionalidade mais de uma vez.

Classes: Um programa que possui partes bem definidas, descritas por várias funções e variáveis que se comunicam, e formam um conjunto bem definido, de forma que há sentido em serem encapsulados.

Sumário

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.

Este arquivo pode possuir várias características:

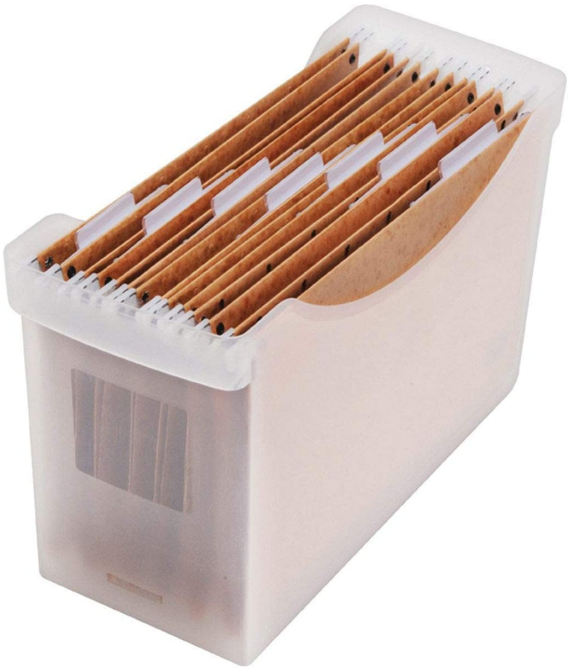
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Para receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

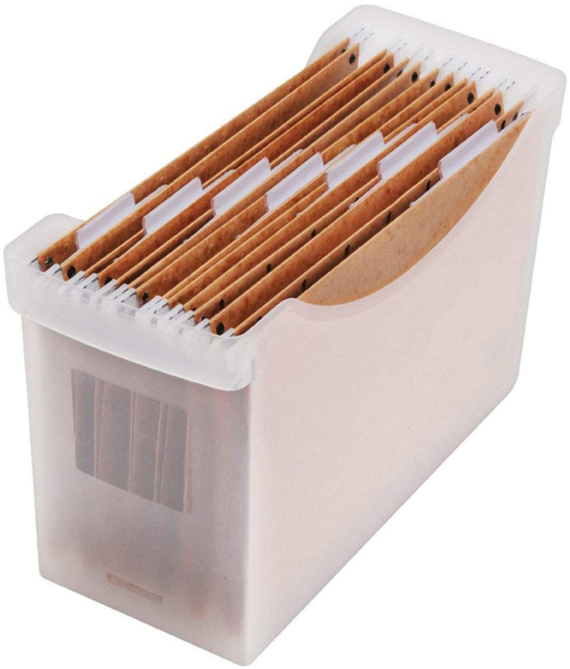
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Para receber uma pasta nova;
  - Para remover uma pasta;
  - Para transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

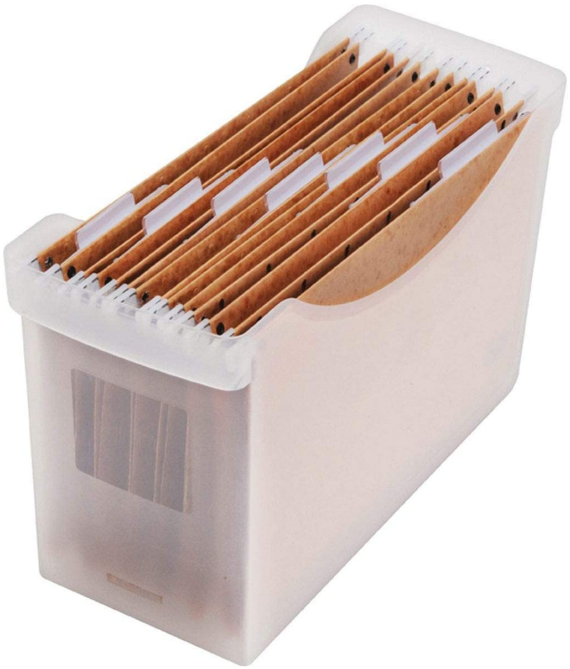
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

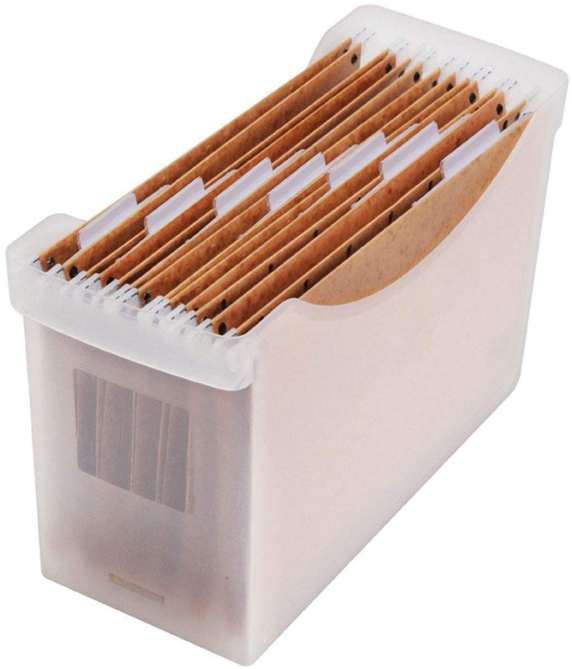
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

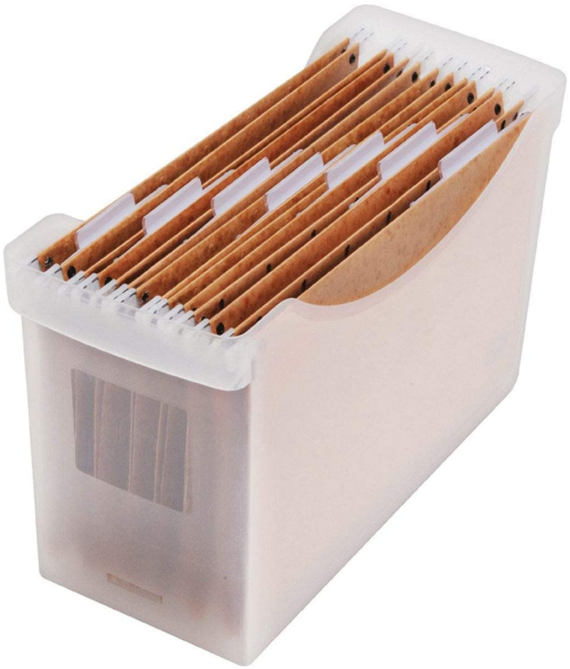
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

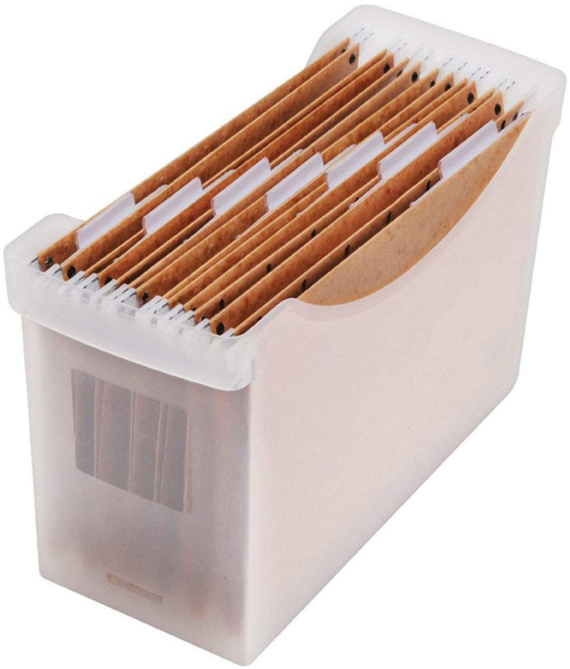
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

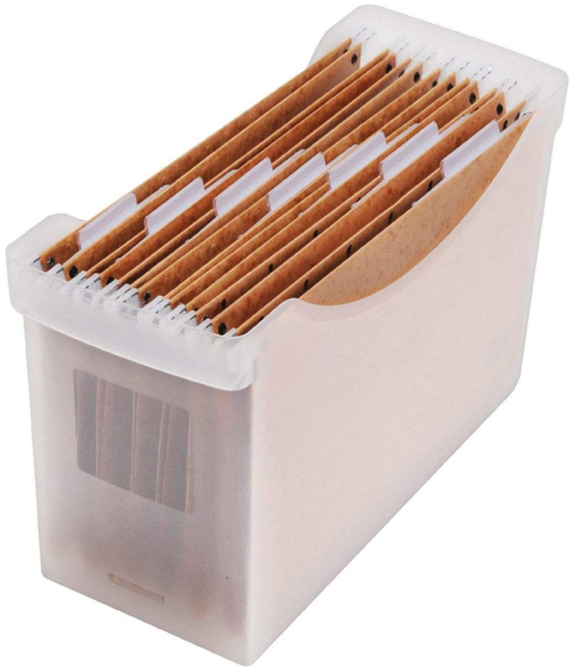
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

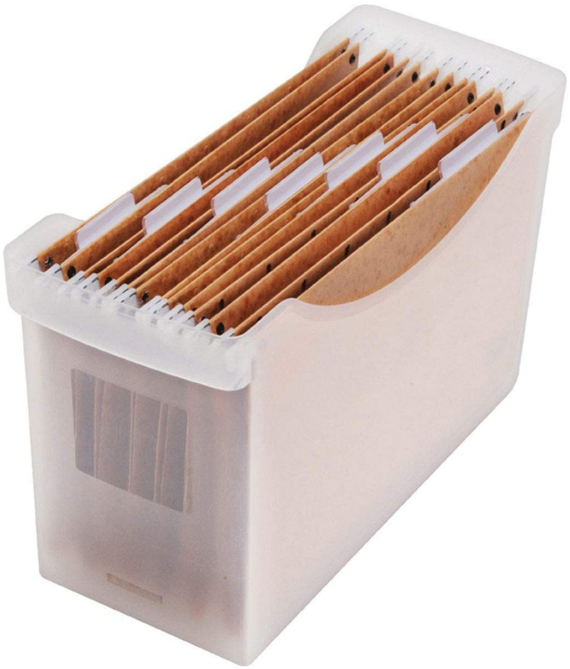
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

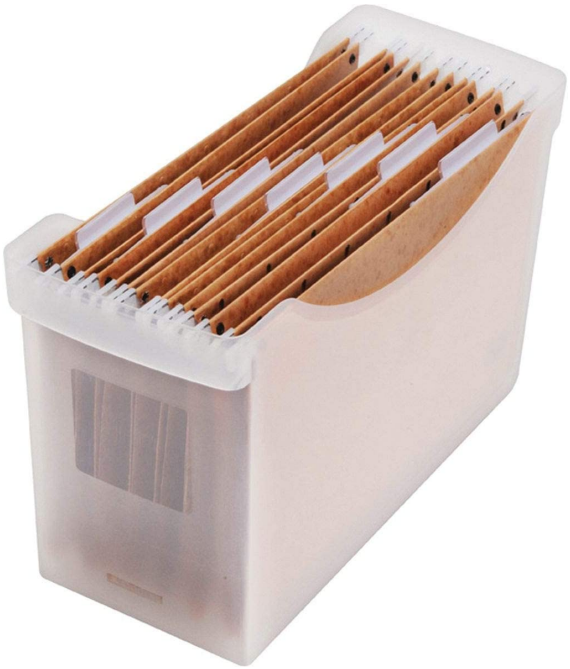
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

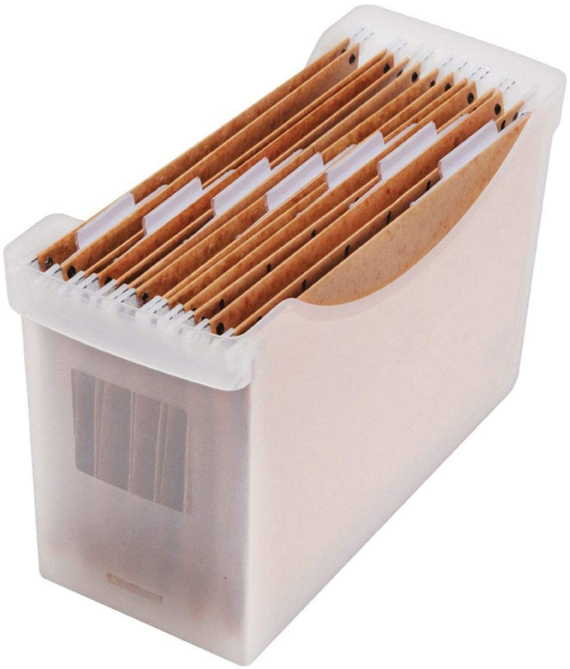
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

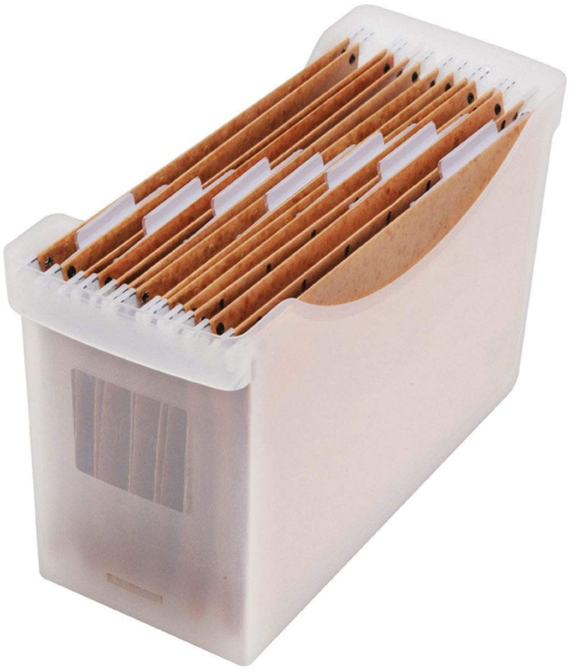
- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Modelagem de um objeto

Vamos modelar uma arquivo de pastas suspensas que guarda informações sobre pessoas.



Este arquivo pode possuir várias características:

- Pode ter uma cor;
- Pode pertencer a um departamento.
- Pode ter um limite máximo de pastas que pode conter.
- Pode realizar algumas ações sobre uma pasta:
  - Pode receber uma pasta nova;
  - Pode remover uma pasta;
  - Pode transferir uma pasta para outro arquivo.

Este é um exemplo de objeto que possui atributos e pode realizar ações.

Modelamos este objeto com uma classe, que é uma estrutura que possui atributos (variáveis) e métodos (funções).

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos. Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Arquivo de pastas suspensas

Supondo que no arquivo de pastas suspensas vamos guardar dados sobre pessoas, como uma pessoa também tem atributos e pode realizar ações, vamos definir um objeto pessoa.

Programa 1

Vamos inicialmente definir uma pessoa somente com atributos, sem métodos.
Os atributos devem ser: nome, idade, sexo.
Crie uma quantidade de pessoas, guarde as informações numa lista e depois mostre ao usuário as pessoas.

Programa 2

Crie agora um arquivo de pastas suspensas que guarde as pessoas, este arquivo pode receber uma pessoa ou excluir uma pessoa, e pode mostrar ao usuário uma pessoa específica ou todas as pessoas no arquivo.
Considere para o objeto Pessoa, que se for uma mulher maior que 30 anos, sua idade será confidencial e não poderá ser mostrada ao usuário.

Sumário

Caixa acoplada

Vamos codificar o objeto abaixo, um vaso sanitário e uma caixa acoplada.

O processo é simples: acabou o serviço, aperte um botão e a água desce da caixa acoplada levando o que tem ali do vaso sanitário para a rede de esgoto.

Caixa acoplada

Vamos codificar o objeto abaixo, um vaso sanitário e uma caixa acoplada.

O processo é simples: acabou o serviço, aperte um botão e a água desce da caixa acoplada levando o que tem ali do vaso sanitário para a rede de esgoto.

Caixa acoplada

Vamos codificar o objeto abaixo, um vaso sanitário e uma caixa acoplada.

O processo é simples: acabou o serviço, aperte um botão e a água desce da caixa acoplada levando o que tem ali do vaso sanitário para a rede de esgoto.

Caixa acoplada

Vamos codificar o objeto abaixo, um vaso sanitário e uma caixa acoplada.

O processo é simples: acabou o serviço, aperte um botão e a água desce da caixa acoplada levando o que tem ali do vaso sanitário para a rede de esgoto.

Caixa acoplada

Vamos codificar o objeto abaixo, um vaso sanitário e uma caixa acoplada.

O processo é simples: acabou o serviço, aperte um botão e a água desce da caixa acoplada levando o que tem ali do vaso sanitário para a rede de esgoto.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.

A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Queremos entender o funcionamento da caixa acoplada.
A caixa acoplada é formada de:

- Uma alavanca de acionamento: o botão pressionado no fim do serviço.
- Uma comporta de vedação: uma tampa de borracha cuja função é abrir para a água descer e fechar para encher a caixa de água.
- Uma válvula de alimentação: responsável por encher de água.
- Uma boia: cuja função é fechar a válvula de alimentação assim que o nível da água estiver completo.

Cada objeto tem uma função específica:

Alavanca de acionamento: abrir a comporta de vedação, permitindo que a água saia da caixa.

Comporta de vedação: se fechar depois que toda a água saiu da caixa acoplada para o vaso.

Válvula de alimentação: Encher de água a caixa até que a bóia chegue ao nível completo.

Caixa acoplada

Faça um programa que simule a operação de uma caixa acoplada.
Para isto use os objetos:

Caixa Acoplada: Contém os outros objetos, tem como atributo o nível de água, e tem como métodos acionar o botão e acionar a entrada de água, controlando até o nível máximo.

Comporta de vedação: Tem como ações abrir, fechar e retornar o estado.

Alavanca de acionamento: Tem como método acionamento da alavanca e como atributo um contador de utilização.

Válvula de alimentação: Tem como método encher de água a caixa e como atributo a vazão.

Caixa acoplada

Faça um programa que simule a operação de uma caixa acoplada.
Para isto use os objetos:

Caixa Acoplada: Contém os outros objetos, tem como atributo o nível de água, e tem como métodos acionar o botão e acionar a entrada de água, controlando até o nível máximo.

Comporta de vedação: Tem como ações abrir, fechar e retornar o estado.

Alavanca de acionamento: Tem como método acionamento da alavanca e como atributo um contador de utilização.

Válvula de alimentação: Tem como método encher de água a caixa e como atributo a vazão.

Caixa acoplada

Faça um programa que simule a operação de uma caixa acoplada.
Para isto use os objetos:

Caixa Acoplada: Contém os outros objetos, tem como atributo o nível de água, e tem como métodos acionar o botão e acionar a entrada de água, controlando até o nível máximo.

Comporta de vedação: Tem como ações abrir, fechar e retornar o estado.

Alavanca de acionamento: Tem como método acionamento da alavanca e como atributo um contador de utilização.

Válvula de alimentação: Tem como método encher de água a caixa e como atributo a vazão.

Caixa acoplada

Faça um programa que simule a operação de uma caixa acoplada.
Para isto use os objetos:

Caixa Acoplada: Contém os outros objetos, tem como atributo o nível de água, e tem como métodos acionar o botão e acionar a entrada de água, controlando até o nível máximo.

Comporta de vedação: Tem como ações abrir, fechar e retornar o estado.

Alavanca de acionamento: Tem como método acionamento da alavanca e como atributo um contador de utilização.

Válvula de alimentação: Tem como método encher de água a caixa e como atributo a vazão.

Caixa acoplada

Faça um programa que simule a operação de uma caixa acoplada.
Para isto use os objetos:

Caixa Acoplada: Contém os outros objetos, tem como atributo o nível de água, e tem como métodos acionar o botão e acionar a entrada de água, controlando até o nível máximo.

Comporta de vedação: Tem como ações abrir, fechar e retornar o estado.

Alavanca de acionamento: Tem como método acionamento da alavanca e como atributo um contador de utilização.

Válvula de alimentação: Tem como método encher de água a caixa e como atributo a vazão.