19.1 - Usando enumeração a modo antigo

Antigamente, criava-se constantes dentro da classe p/ definir o status dela.

isso era um Problema, pois ele gerava muitos Problemas. Um deles é o type safe, ou seja, ele não é seguro, não garante quais são os valores finitos que podemos atribuir ao status um status valido. Outro Problema é que usar constantes pode gera Ambiguidade, pq podemos gerar constantes tambem p/ outro atributo e acabarmos confundindo.

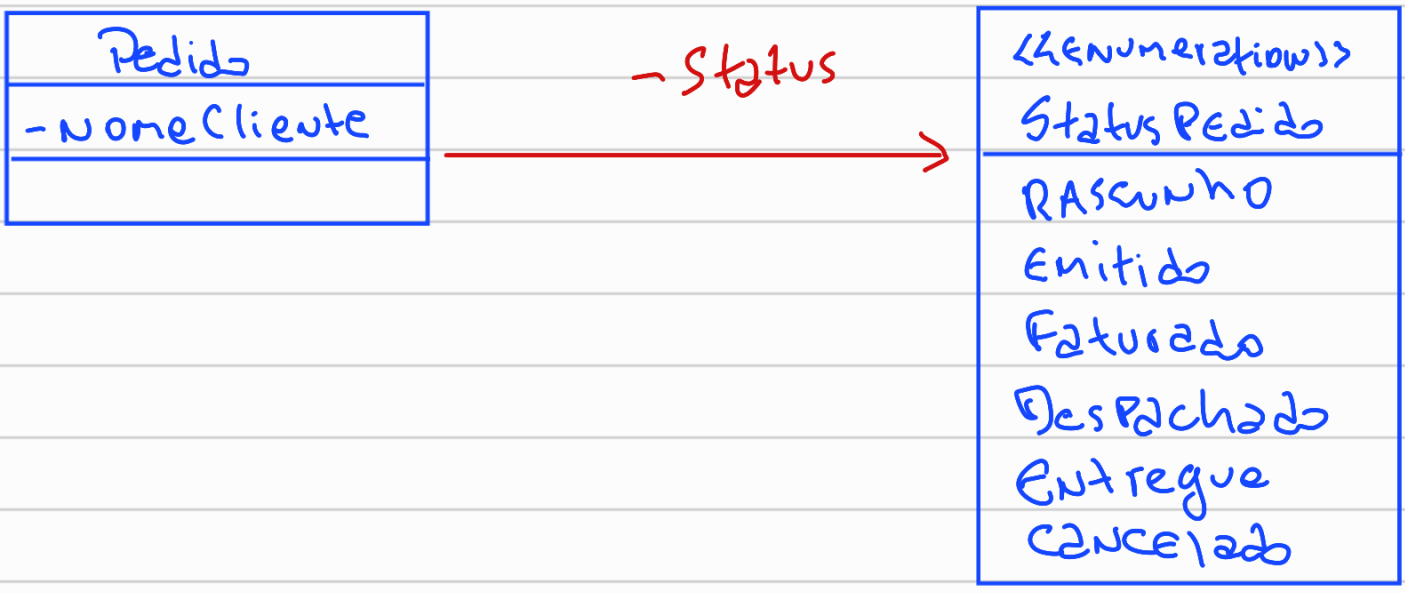
19.2 - Criando tipo enum

A partir do Java 5, podemos usar o tipo enum para representar uma enumeração com uma lista finita de constantes.

O tipo enum, além de resolver os Problemas da aula anterior, também é uma ferramenta poderosissima, permitindo que adicionemos atributos, métodos, implemente interface, além de serem comparaveis com compareto e equals

19.3 - Diagrama de classes: Enumeração

Como representar com Diagrama de classe um enum?



19.4 - Usando métodos do tipo Enum

Em tempo de execução, nossa enumeração possui um valor inteiro único, um identificador. Para saber qual numero é esse, basta chamar o metodo .ordinal() e para pegar o nome do Enum em formato de String, basta usar o método .name ou .toString

Exemplo:

StatusPedido.EMITIR.ordinal()

ordinal() → valor numerico

EMITIR → Enum

ou StatusPedido.EMITIR.name()
ou StatusPedido.EMITIR.toString()

StatusPedido → classe Enum

name() / toString() → nome do enum em string

EMITIR → Enum

StatusPedido.EMITIR.values().For

↙ iterar na classe do enum

StatusPedido.EMITIR.valueOF("Emitir")

↙ valida/retorna se existe um enum com esse nome.

Para acessar um enum por um índice, usamos o método .value()[int]

exemplo

```
int numero = 1;
StatusPedido status = StatusPedido.values()[numero];
Sout(status);
```

19.5 - Declarando e inicializando propriedades e Construtores

Uma enumeração só pode ter construtor Private (Default).

Também podemos gerar métodos getters e setters, com visibilidade public.

O construto só pode ser usado por ele mesmo, sendo passado nos enums

19.6 - Implementando métodos.

Adicionamos métodos normais como qualquer outra classe.

19.7 - implemetando métodos abstratos

Cenário: Sempre que adicionamos um status no enum, precisamos mexer no método do enum que tem uma regra de Negócio.

Solução: No próprio enum, criar um método abstrato que será implementado pela constante.

19.8- Boas Praticas: Substitua parâmetros booleanos por enums.

Passar true ou false no codigo não é legal, o ideal é passar um enum (nem que sejam duas constants), pois o codigo fica mais legivel

19.9 - Desafio Enumerações

O que eu aprendi?

1) Criar construtor do enum

```
MeuEnum( ) {

}
```

2) Criar atributo e gerar o getter
   - Normal

3) Atribuir ao construtor o valor do atributo
   - Normal

4) Adicionar na constante do enum o param do construtor

```
MEU ENUM {

    Valor_1 (valor)
    Valor_2 (valor)

    ~ construtor ~
    ~ getter ~
```