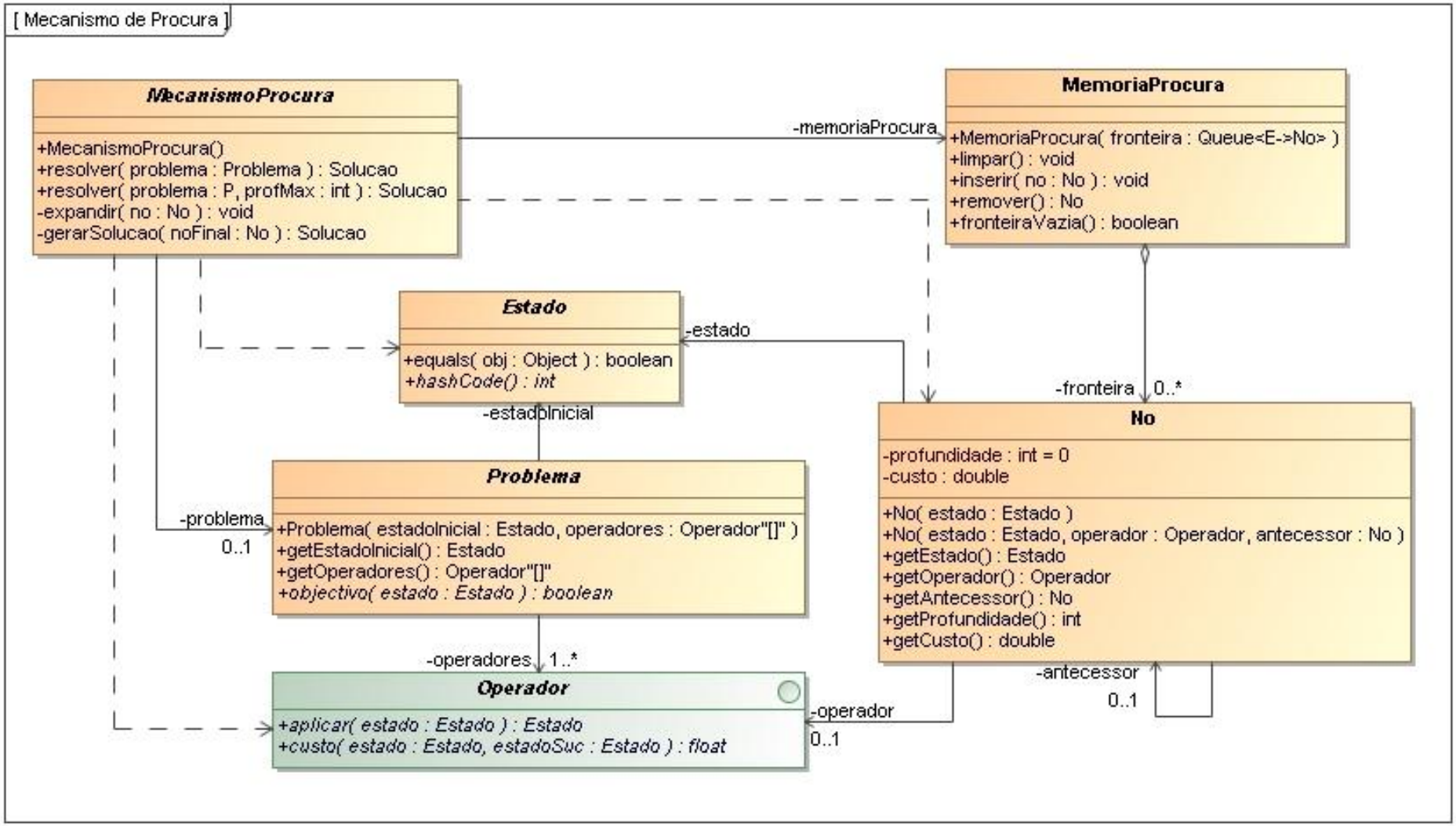
MECANISMO DE PROCURA



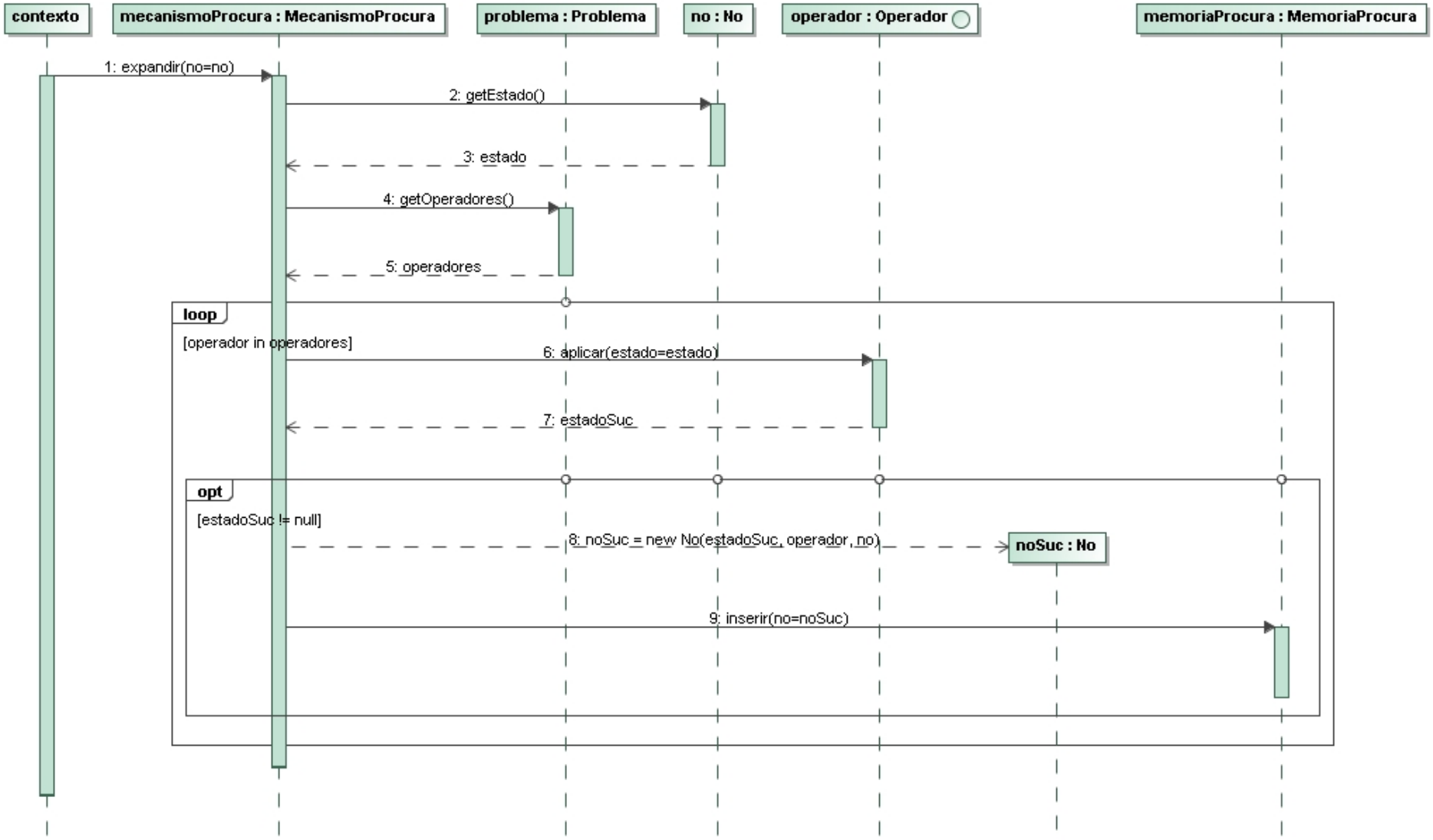
RESOLVER PROBLEMA

```
function resolver(problema : Problema) : Solucao
    problema = problema
    memoria_procura.limpar()
    no_inicial = No(problema.estado_inicial)
    memoria_procura.inserir(no_inicial)
    while not memoria_procura.fronteira_vazia:
        no = memoria_procura.remover()
        if problema.objectivo(no.estado):
            return gerar_solucao(no)
        else:
            expandir(no)
```

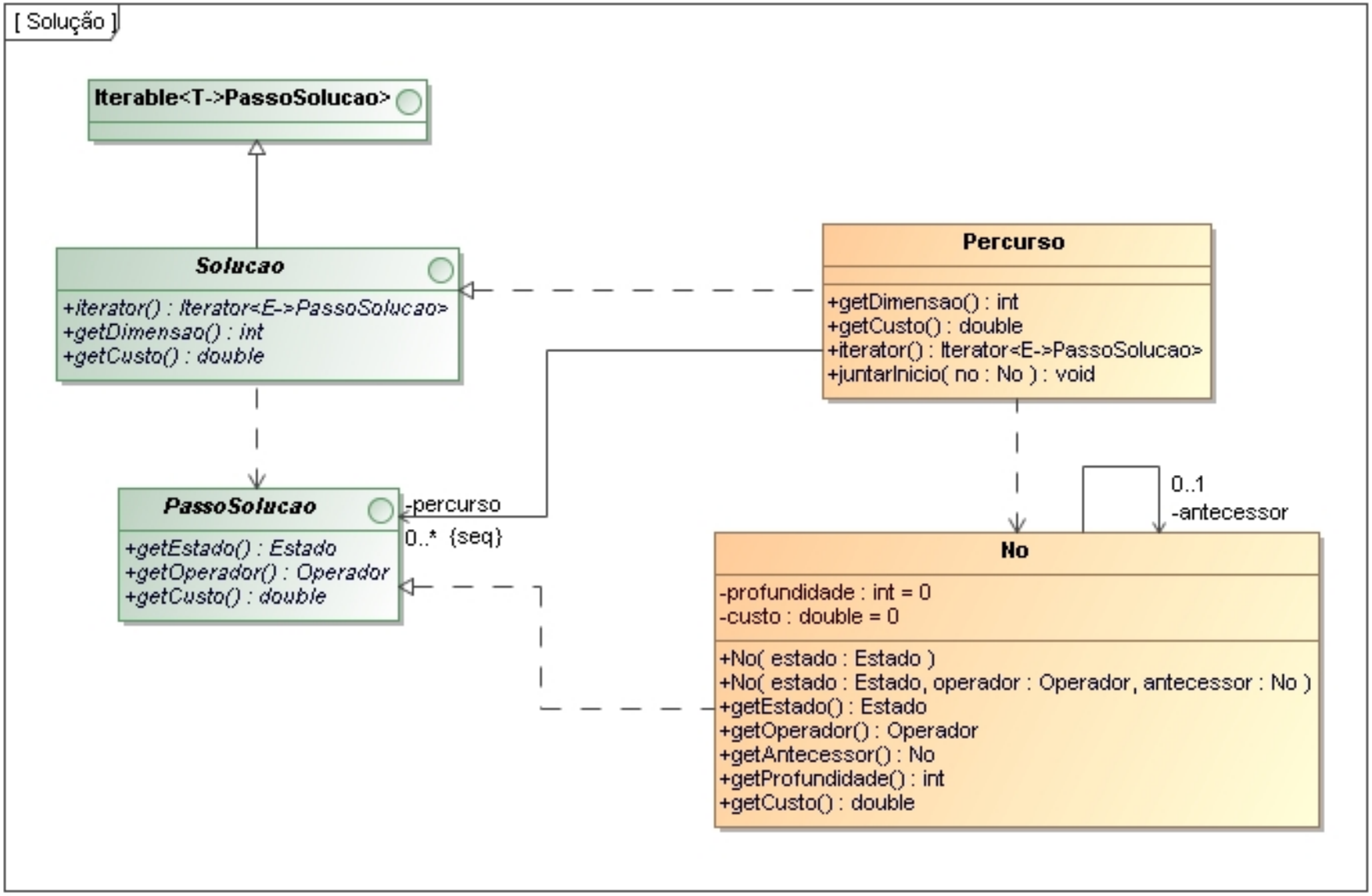
RESOLVER PROBLEMA COM LIMITAÇÃO DE PROFUNDIDADE

```
function resolver(problema : Problema, prof_max : int) : Solucao
    problema = problema
    memoria_procura.limpar()
    no_inicial = No(problema.estado_inicial)
    memoria_procura.inserir(no_inicial)
    while not memoria_procura.fronteira_vazia:
        no = memoria_procura.remover()
        if problema.objectivo(no.estado):
            return gerar_solucao(no)
        else:
            if no.profundidade < prof_max):
                expandir(no)
```

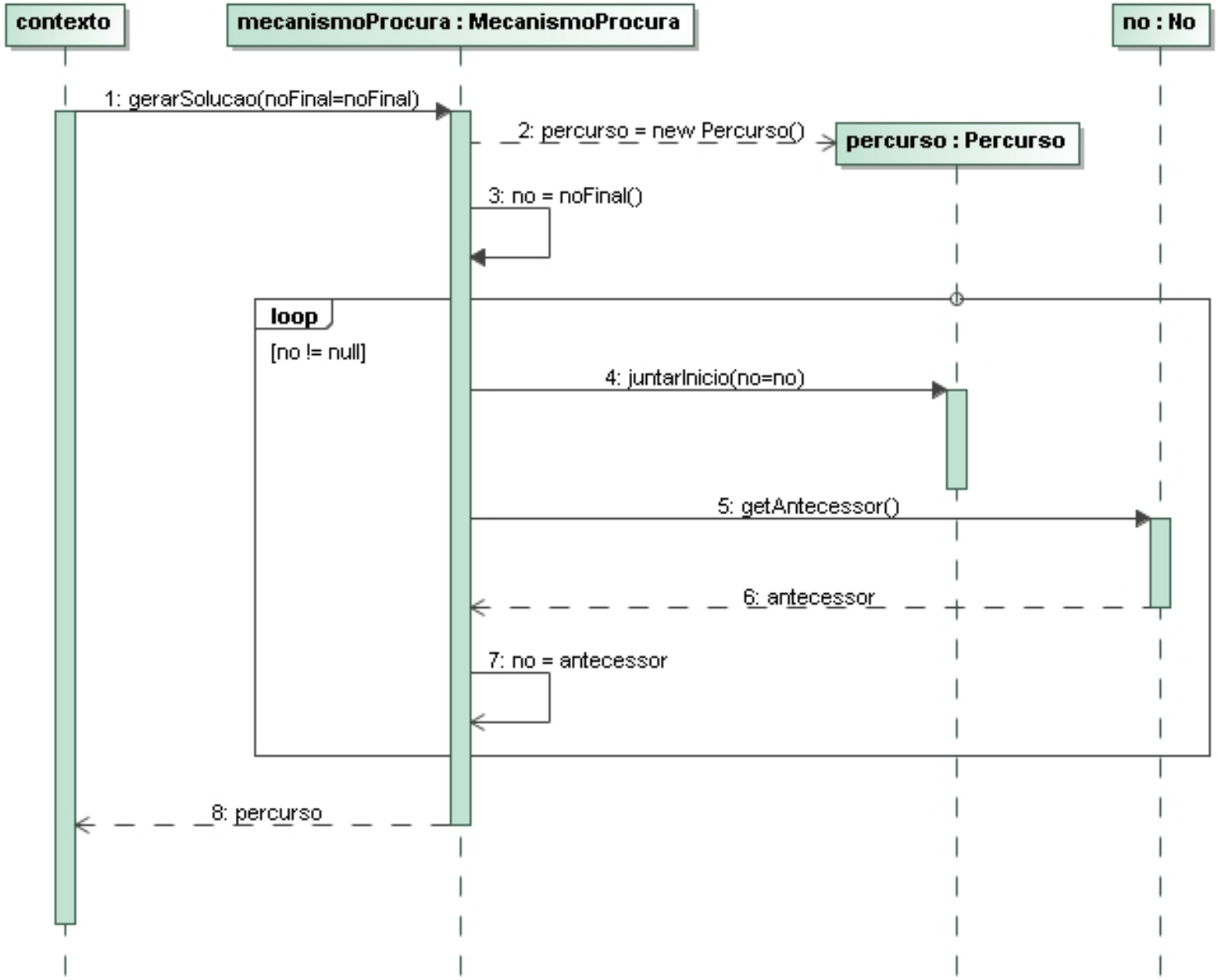
EXPANDIR NÓ


contexto
mecanismoProcura : MecanismoProcura
problema : Problema
no : No
operador : Operador
memoriaProcura : MemoriaProcura
1: expandir(no=no)
2: getEstado()
3: estado
4: getOperadores()
5: operadores
loop
[operador in operadores]
6: aplicar(estado=estado)
7: estadoSuc
opt
[estadoSuc != null]
8: noSuc = new No(estadoSuc, operador, no)
noSuc : No
9: inserir(no=noSuc)

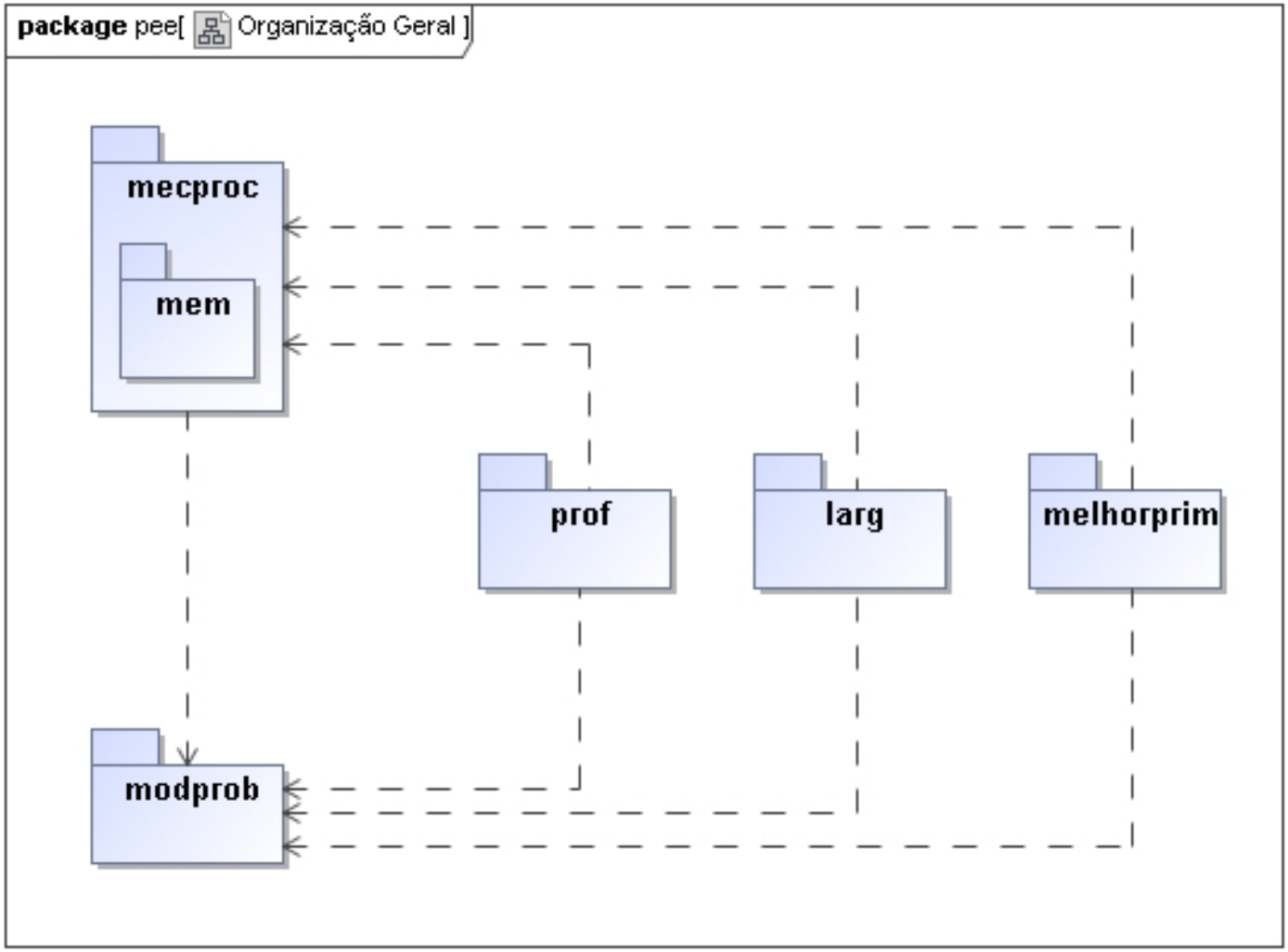
SOLUÇÃO DE UMA PROCURA



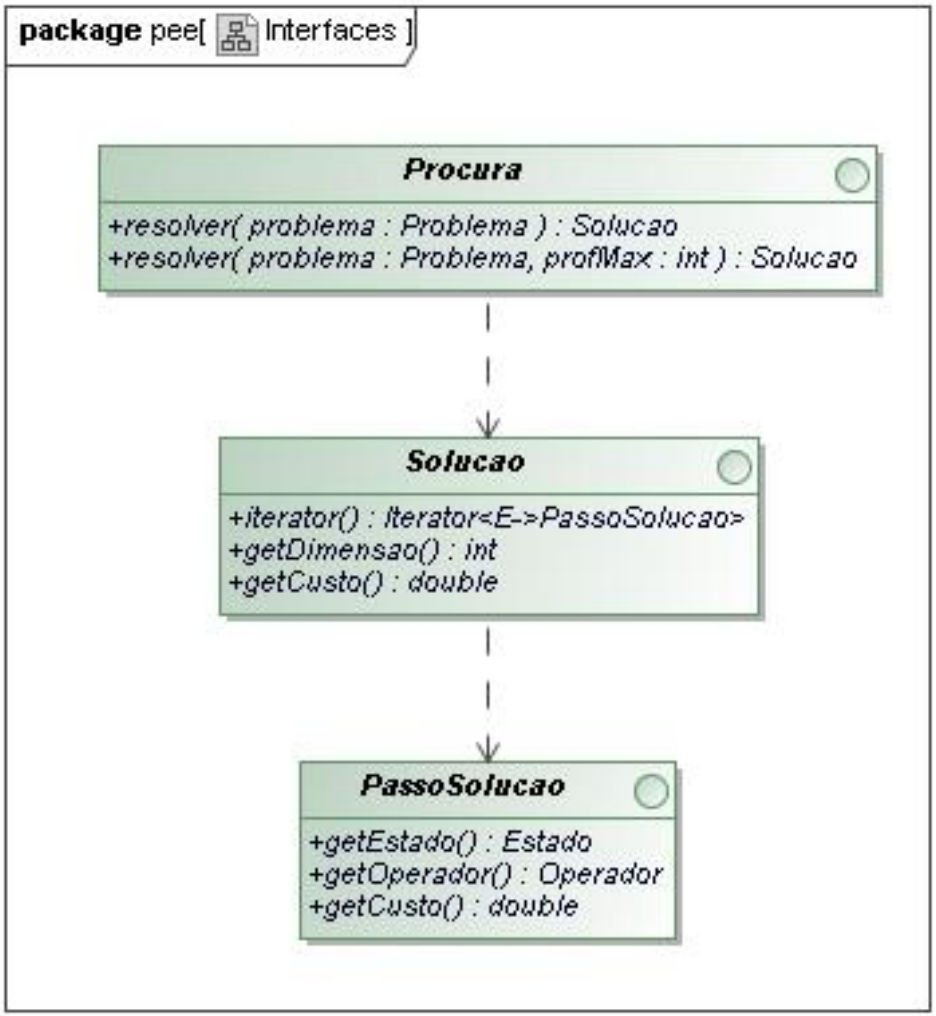
GERAR SOLUÇÃO



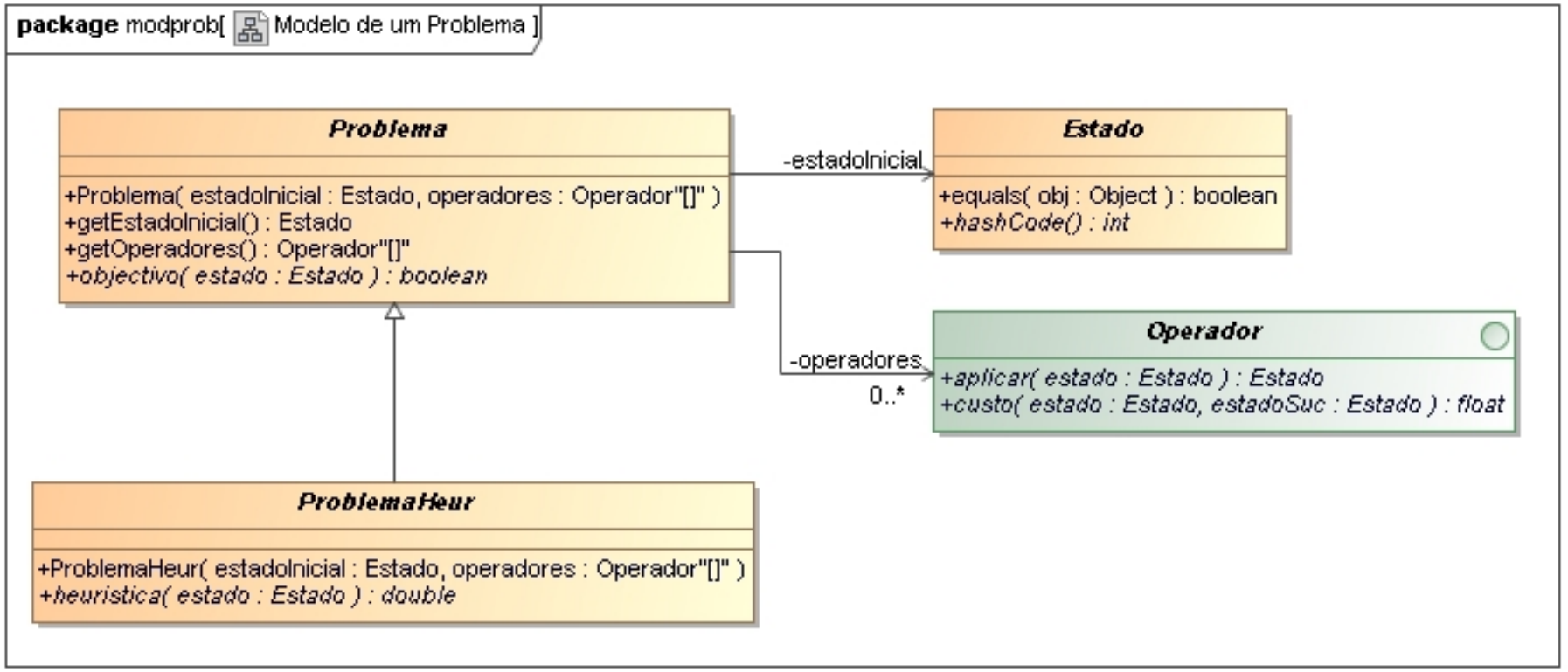
MÓDULO PEE – ORGANIZAÇÃO GERAL



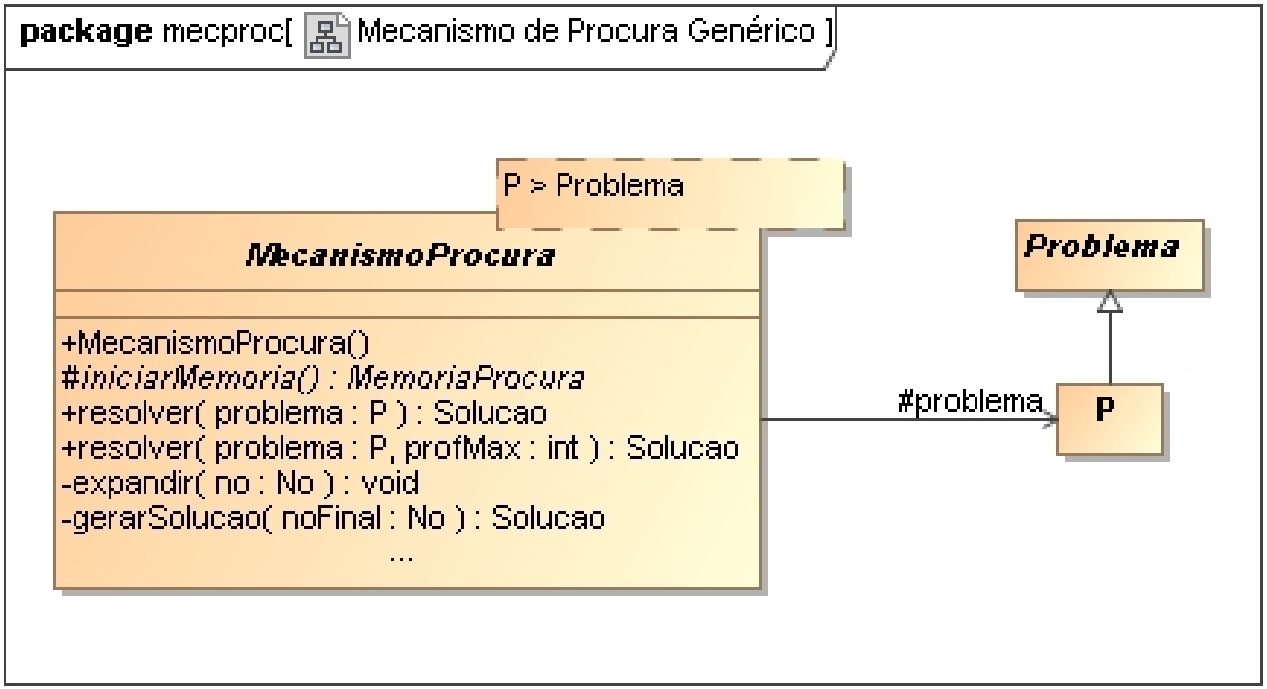
MÓDULO PEE – INTERFACES



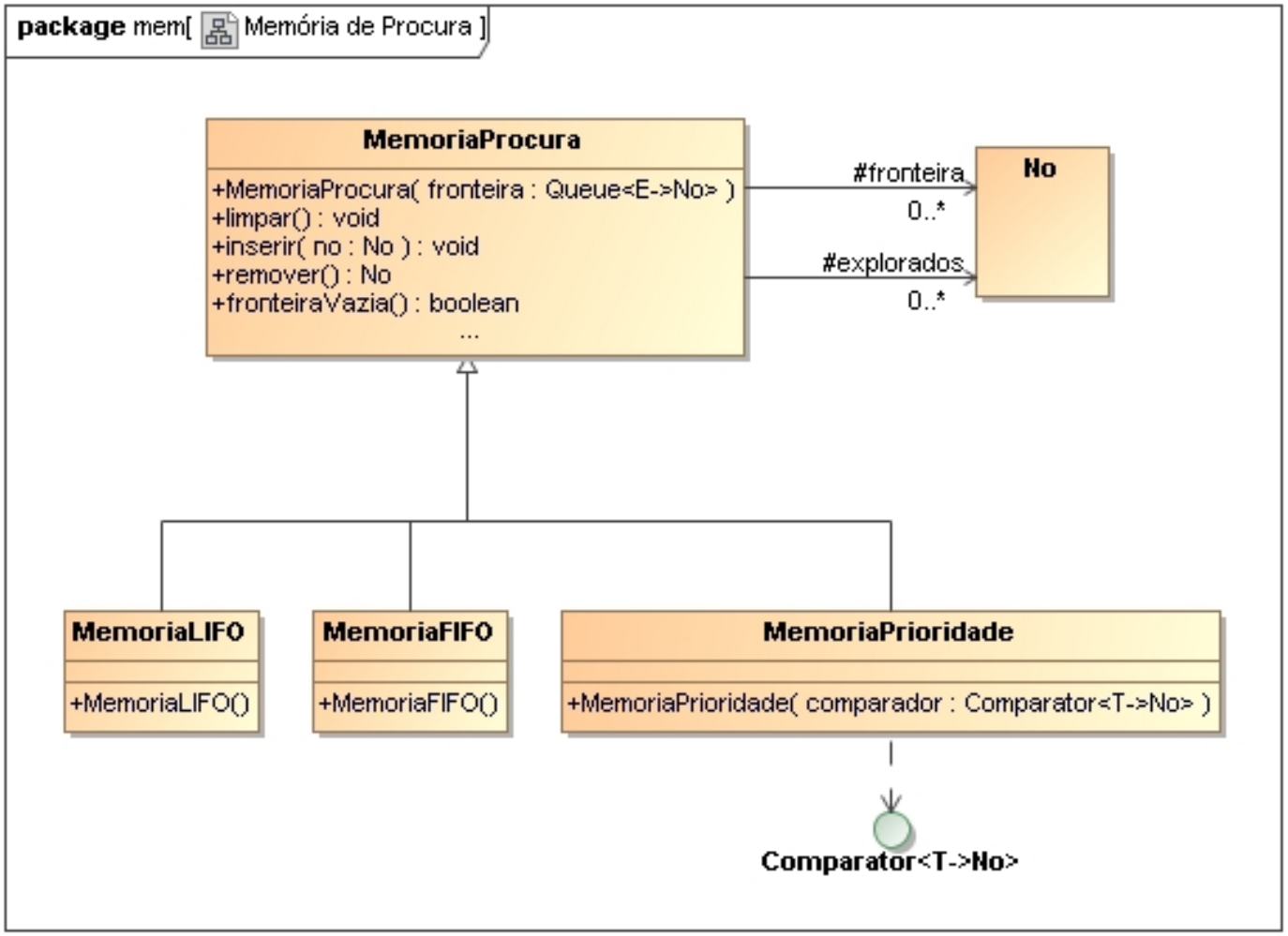
MODELO DE UM PROBLEMA



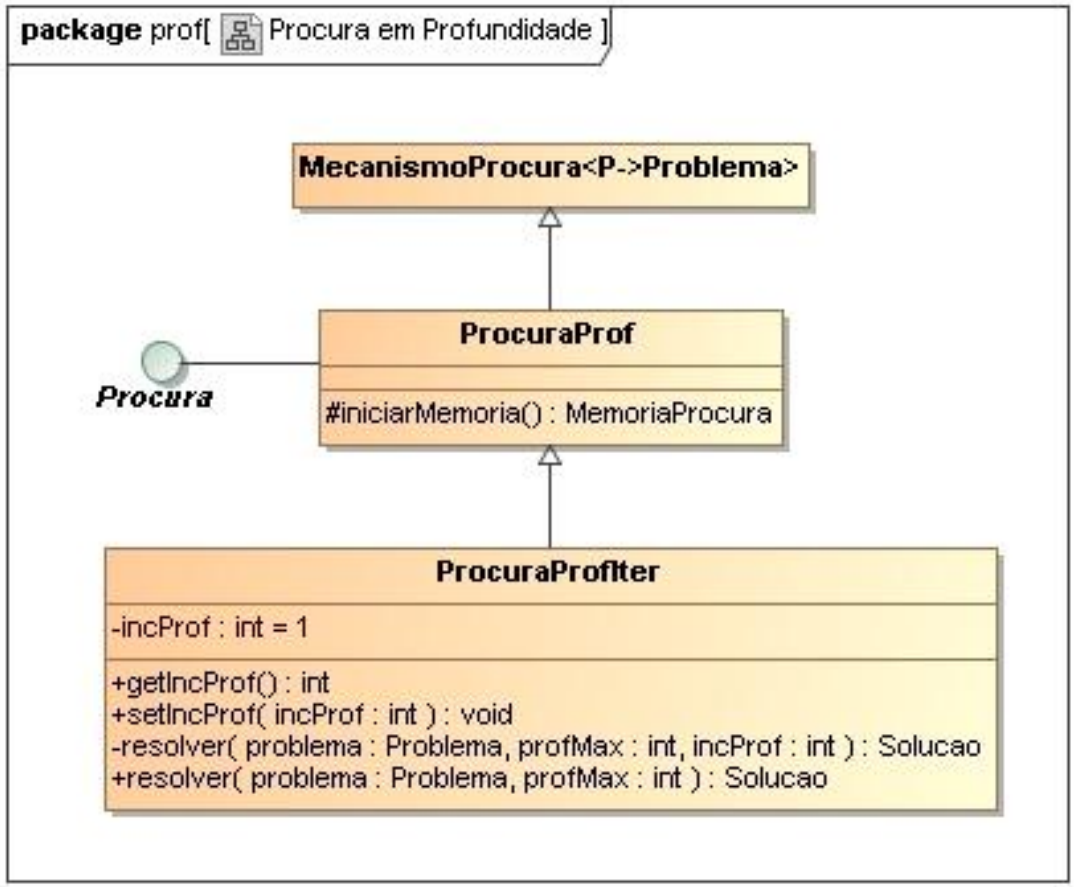
MECANISMO DE PROCURA GENÉRICO PARA DIFERENTES TIPOS DE PROBLEMA



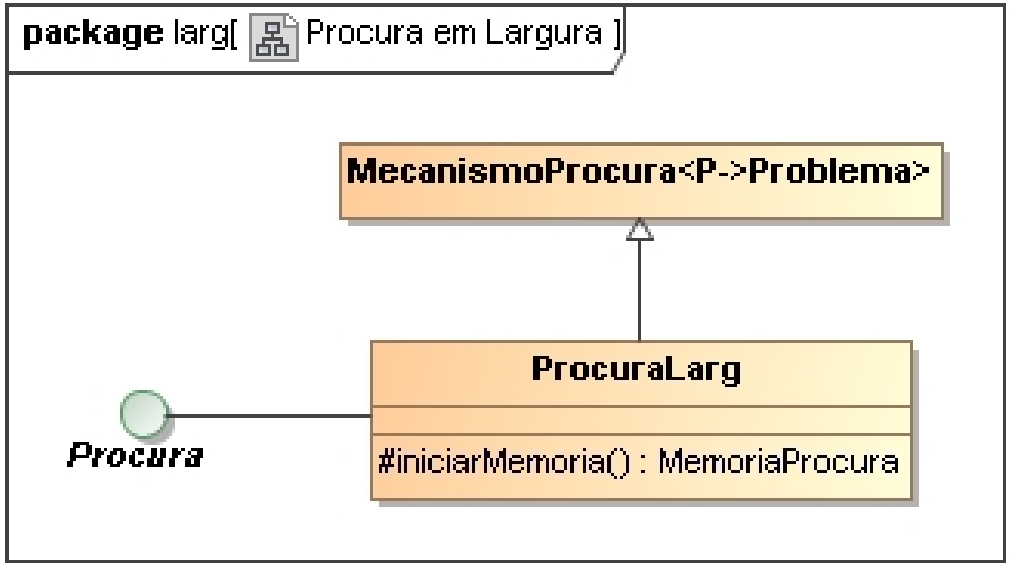
MEMÓRIA DE PROCURA



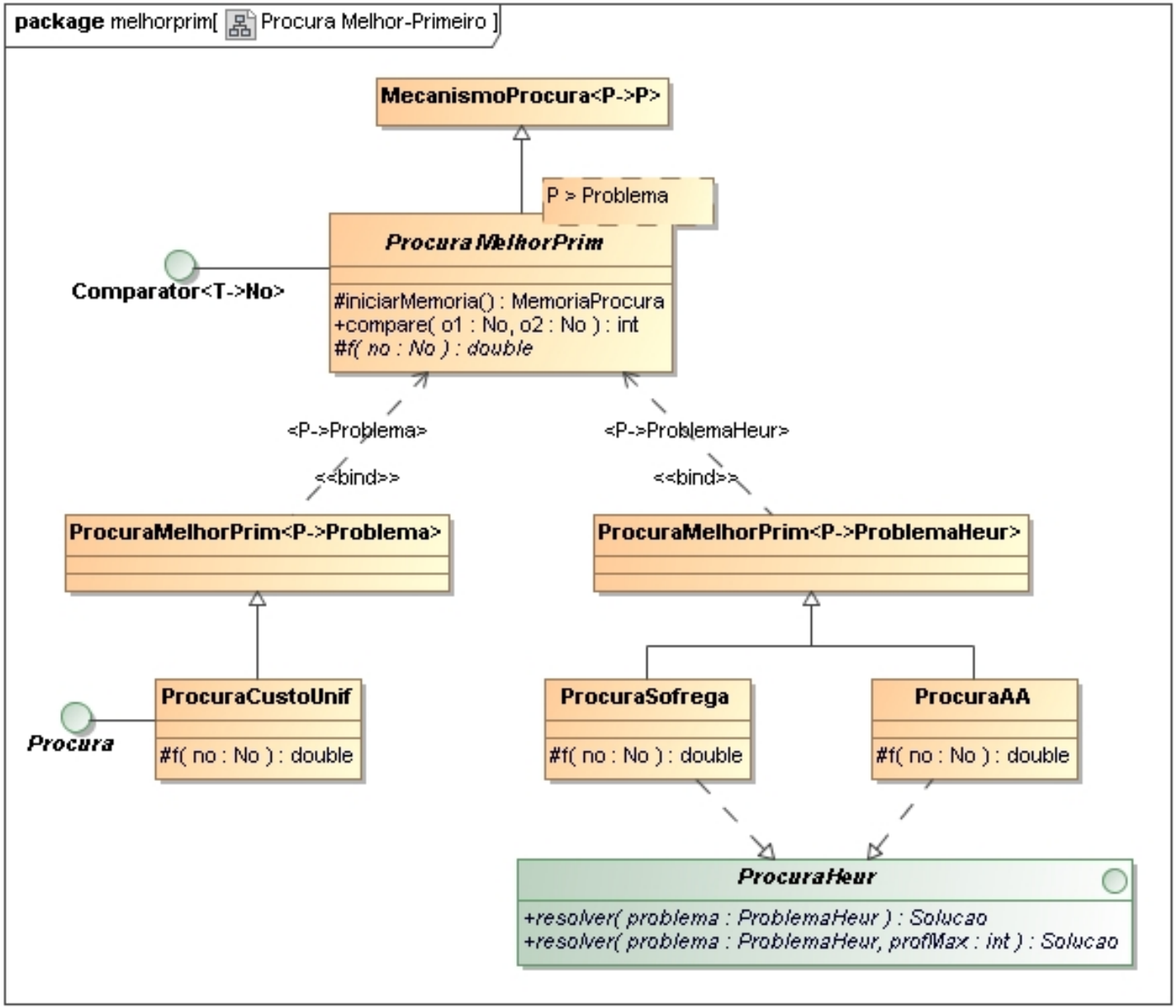
PROCURA EM PROFUNDIDADE



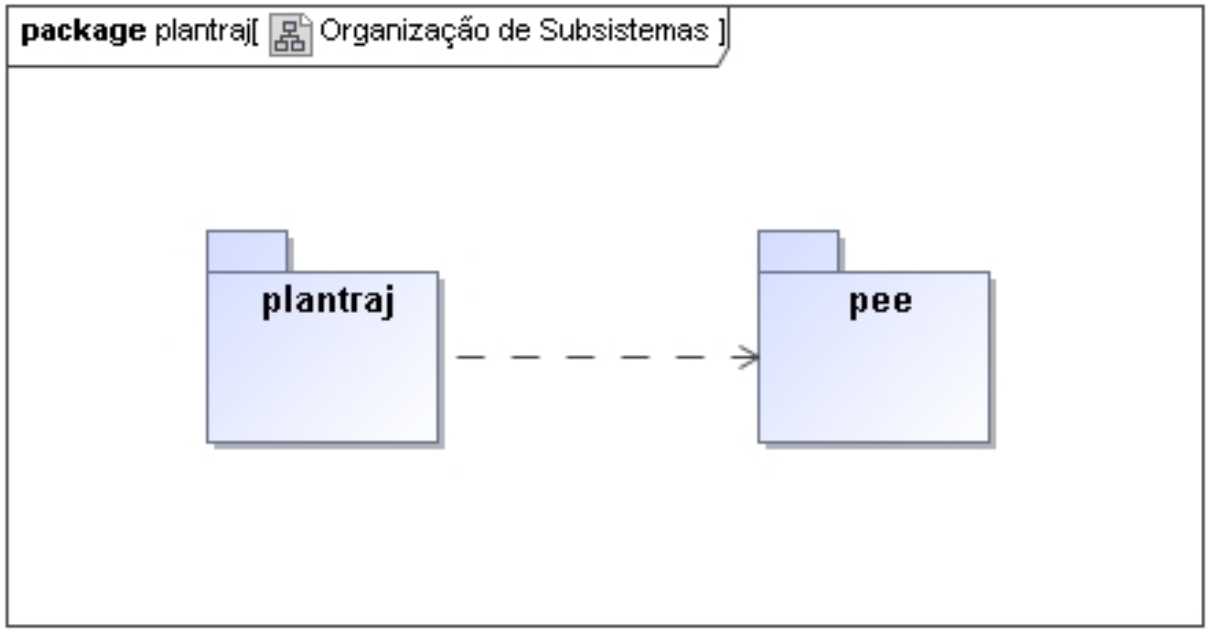
PROCURA EM LARGURA



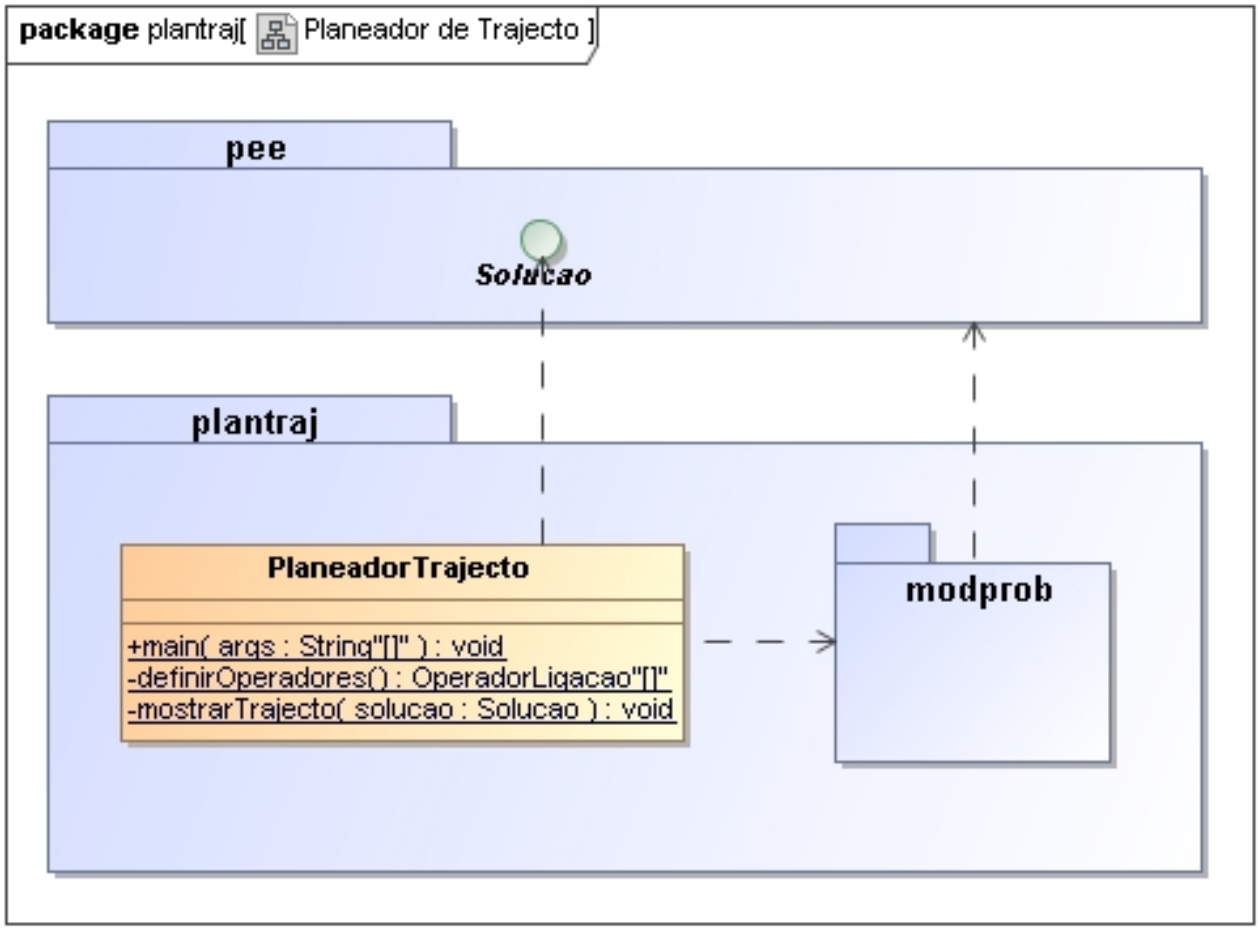
PROCURA MELHOR-PRIMEIRO



PLANEADOR DE TRAJECTO



PLANEADOR DE TRAJECTO - DETALHE



MODELO DO PROBLEMA DE PLANEAMENTO DE TRAJECTO