Service : Level

Types : bool, int, enum Nature{EMPTY, DIRTY, METAL}

Observators:

```
const height:[Level] -> int
const width:[Level] -> int
editing:[Level] -> bool
caseExiste:[Level] * int * int -> bool
nature:[Level] * int * int -> Nature
    pre : nature(L,x,y) require caseExiste(L,x,y)
exitX : [Level] -> int
    pre exitX(L) require ¬(editing(L))
exitY : [Level] -> int
    pre exitY(L) require ¬(editing(L))
entranceX: [Level] -> int
    pre entranceX(L) require ¬(editing(L))
entranceY: [Level] -> int
    pre entranceY(L) require ¬(editing(L))
```

Constructors:

```
init:int * int -> [Level]
    pre init(h , w) require (h >= 5)^(w >= 4)
```

Operators :

```
setNature:[Level] * int * int * Nature -> [Level]
    pre setNature(L, x, y, Nat) require
        caseExiste(L, x, y)
        ^ editing(L)

goPlay:[Level] * int * int * int * int -> [Level]
    pre goPlay(L, eX, eY, qX, qY ) require
        editing(L)
        ^ (∀  ∉ {0,  height(L) -1} et ∉ {0 ...  width(L) -1} nature(L,i,j) = METAL)
        ^ (∀  ∉ {0 ...  height(L) -1} et ∉ {0 ,  width(L) -1} nature(L,i,j) = METAL)
        ^ nature(L,eX,eY -1) = nature(L,eX,eY) = nature(L,eX,eY+1) = EMPTY
        ^ nature(L,qX,qY -1) = METAL ^ nature(L,qX,qY) = EMPTY ^ nature(L,qX,qY+1) = EMPTY
        ^  ¬ ((eX == qX) && (eY == qY) )


remove:[Level] * int * int -> [Level]
    pre remove(L, x, y) require
        ¬(editing(L))
        ^ caseExiste(x, y)
        ^ nature(L, x, y) = DIRTY

build:[Level] * int * int -> [Level]
    pre build(L, x, y) require
        ¬(editing(L))
        ^ caseExiste(x, y)
        ^ nature(L, x, y) = EMPTY
        ^ ¬(x = entranceX(L) && (y = entranceY(L) || y = entranceY(L) -1 && y = entranceY(L) +1))
        ^ ¬(x = exitX(L) && (y = exitY(L) || y = exitY(L) +1))
```

Observations :

```
[invariant]
    height(L) > 5
    width(L) > 4
    caseExiste(L, n1, n2) min=  0 <= n1 < height(L) && 0 <= n1 < width(L)
```

[init]

height(init(h , w)) = h
width(init(h , w)) = w
editing(init(h , w)) = true
∀*i* ∈ *{0 ... w -1}*;∀*j* ∈ *{0 ... h -1}*
nature(init(h , w), i, j) = EMPTY

exitX(init(h , w)) = null
exitY(init(h , w)) = null
entranceX(init(h , w)) = null
entranceY(init(h , w)) = null

[setNature]

editing(setNature(L, x, y, Nat)) = true

nature(setNature(L, x, y, Nat), x, y) = Nat
∀ *x1*∈ *{0 ... w -1}*;∀*y1* ∈ *{0 ... h -1}; x != x1 || y != y1*
nature(setNature(L, x, y, Nat), x1, y1) = nature(L, x1, y1)@pre

exitX(setNature(L, x, y, Nat)) = null
exitY(setNature(L, x, y, Nat)) = null
entranceX(setNature(L, x, y, Nat)) = null
entranceY(setNature(L, x, y, Nat)) = null

[goPlay]

editing(goPlay(L, eX, eY, qX, qY)) = false
∀*x* ∈ *{0 ... w -1}*;∀*y* ∈ *{0 ... h -1}*
nature(goPlay(L, eX, eY, qX, qY),x,y) = nature(L,x,y)@pre
entranceX(goPlay(L,eX, eY, qX, qY)) = eX
entranceY(goPlay(L,eX, eY, qX, qY)) = eY
exitX(goPlay(L, eX, eY, qX, qY))= qX
exitY(goPlay(L, eX, eY, qX, qY))= qY

[remove]

editing(remove(L, i, j)) = false
nature(remove(L,i,j),i,j) = EMPTY
∀*x* ∈ *{0 ... w -1}*;∀*y* ∈ *{0 ... h -1}; x!=i || y!=j*
nature(remove(L,i,j),x,y) = nature(L,x,y)@pre
exitX(remove(L, i, j)) = exitX(L)@pre
exitY(remove(L, i, j)) = exitY(L)@pre
entranceX(remove(L, i, j)) = entranceX(L)@pre
entranceY(remove(L, i, j)) = entranceY(L)@pre

[build]

editing(build(L, i, j)) = false
nature(build(L,i,j)) = DIRTY
∀*x* ∈ *{0 ... w -1}*;∀*y* ∈ *{0 ... h -1}; x!=i || y!=j*
nature(build(L,i,j),x,y) = nature(L,x,y)@pre
exitX(build(L, i, j)) = exitX(L)@pre
exitY(build(L, i, j)) = exitY(L)@pre
entranceX(build(L, i, j)) = entranceX(L)@pre
entranceY(build(L, i, j)) = entranceY(L)@pre