Service : Lemming
use : GameEng, ActivityLemming
types : Direction{GAUCHE, DROITE}, ClasseType{TOMBEUR, MARCHEUR, CREUSEUR, GRIMPEUR, CONSTRUCTEUR, EXPLOSEUR, STOPPEUR, FLOTTEUR, PELLETEUR, MINEUR}

Observators :

const gameEng : [Lemming] -> GameEng
x : [Lemming] -> int
y : [lemming] -> int
direction : [Lemming] -> Direction
classeType : [Lemming] -> ClasseType
enChute : [lemming] -> int
estFlotteur : [Lemming] -> boolean
estExploseur : [Lemming] -> boolean
estGrimpeur : [Lemming] -> boolean
attenteConstruction : [lemming] -> int
attenteExplosion : [lemming] -> int
nbDalle : [lemming] -> int
nbCasse : [lemming] -> int

Constructors :

init : GamingEng * int * int -> [GameEng]
pre : init(G, x, y) require
G != null
^ Level::caseExiste(GameEng::level(G), x,y)
^ Level::nature(GameEng::level(G), x, y) = EMPTY
^ Level::nature(GameEng::level(G), x, y +1) = EMPTY

Operators :

step : [Lemming] -> Lemming
setClasseType : [Lemming] * ClasseType -> Lemming
changeDirection : [Lemming] -> Lemming
setFlotteur : [Lemming] * boolean -> Lemming
setExploseur: [Lemming] * boolean -> Lemming
setEstGrimpeur: [Lemming] * boolean -> Lemming

Observations :

[Invariant]
Level::nature(GameEng::level(gameEng(L)), x, y +1) = EMPTY
Level::nature(GameEng::level(gameEng(L)), x, y) = EMPTY

[init]
gameEng(init(G, i,j)) = G
x(init(G, i,j)) = i
y(init(G, i,j)) = j
classeType(init(G, i,j)) = MARCHEUR
direction(init(G, i,j)) = DROITE
enChute(init(G, i,j)) = 0
estFlotteur(init(G, i,j)) = false
estExploseur(init(G, i,j)) = false
estGrimpeur(init(G, i,j)) = false
attenteConstruction(init(G, i,j)) = 0
attenteExplosion(init(G, i,j)) = 0
nbDalle(init(G, i,j)) = 0
nbCasse(init(G, i,j)) = 0

[setClasseType]
x(setClasseType(L, T)) = x(L)@pre
y(setClasseType(L, T)) = y(L)@pre
direction(setClasseType(L, T)) = direction(L)@pre
classeType(setClasseType(L, T)) = T

```
        enChute(setClasseType(L, T)) = 0
        estFlotteur(setClasseType(L, T)) = estFlotteur(L)@pre
        estExploseur(setClasseType(L, T)) = estExploseur(L)@pre
        estGrimpeur(setClasseType(L, T)) = estGrimpeur(L)@pre
        attenteConstruction(setClasseType(L, T)) = 0
        attenteExplosion(setClasseType(L, T))  = 0
        nbDalle(setClasseType(L, T)) = 0
        nbCasse(setClasseType(L, T)) = 0

[changeDirection]
        x(changeDirection(L)) = x(L)@pre
        y(changeDirection(L)) = y(L)@pre
        if direction(G) = GAUCHE then
                direction(changeDirection(L)) = DROITE
        else
                direction(changeDirection(L)) = GAUCHE
        classeType(changeDirection(L)) = classeType(L)@pre
        enChute(changeDirection(L)) = enChute(L)@pre
        estFlotteur(changeDirection(L)) = estFlotteur(L)@pre
        estExploseur(changeDirection(L)) = estExploseur(L)@pre
        estGrimpeur(changeDirection(L)) = estGrimpeur(L)@pre
        attenteConstruction(changeDirection(L)) = attenteConstruction(L)@pre
        attenteExplosion(changeDirection(L))  = attenteExplosion(L)@pre
        nbDalle(changeDirection(L)) = nbDalle(L)@pre
        nbCasse(changeDirection(L)) = nbCasse(L)@pre

[setEstGrimpeur]
        x(setEstGrimpeur(L), bol) = x(L)@pre
        y(setEstGrimpeur(L), bol) = y(L)@pre
        direction(setEstGrimpeur(L), bol) = direction(L)@pre
        classeType(setEstGrimpeur(L), bol) = classeType(L)@pre
        enChute(setEstGrimpeur(L), bol) = enChute(L)@pre
        estFlotteur(setEstGrimpeur(L), bol) = estFlotteur(L)@pre
        estExploseur(setEstGrimpeur(L), bol) = estExploseur(L)@pre
        estGrimpeur(setEstGrimpeur(L), bol) = bol
        attenteConstruction(setEstGrimpeur(L), bol) = attenteConstruction(L)@pre
        attenteExplosion(setClasseType(L, T))  = attenteExplosion(L)@pre
        nbDalle(setClasseType(L, T)) = nbDalle(L)@pre
        nbCasse(setClasseType(L, T)) = nbCasse(L)@pre

[step]
        if classeType(L) = MARCHEUR then :
                if obstacle(L, x(L), y(L)-1)  then :
                        classeType(step(L)) = classeType(L)
                        if obstacle(L, x(L)+1,y(L)+1) || ( obstacle(L, x(L)+1, y(L)) && obstacle(x(L)+1,y(L)+2) ) then :
                                direction(step(L)) = direction(changeDirection(L))
                                x(step(L)) = x(L)@pre
                                y(step(L)) = y(L)@pre
                        else
                                direction(step(L)) = direction(L)@pre
                                x(step(L)) = x(L)@pre +1
                                if obstacle(L, x(L)+1, y(L))  then :
                                        y(step(L)) = y(L)@pre +1
                                else
                                        y(step(L)) = y(L)@pre
                else
                        classeType(step(L)) = TOMBEUR
                estFlotteur(step(L)) = estFlotteur(L)@pre
                estExploseur(step(L)) = estExploseur(L)@pre
                attenteConstruction(step(L)) = attenteConstruction(L)@pre
                attenteExplosion(step(L))  = attenteExplosion(L)@pre
                nbDalle(step(L)) = nbDalle(L)@pre
                nbCasse(step(L)) = nbCasse(L)@pre
```

```
else if classeType(L) = TOMBEUR then :
        x(step(L)) = x(L)@pre
        direction(step(L)) = direction(L)@pre
        if(obstacle(L, x(L), y(L)-1)) then :
                y(step(L)) = y(L)@pre
                classeType(step(L)) = MARCHEUR
                if enChute(L) >= 8 then :
                        L ∉ GameEng::lemmings(gameEng(L))
                estFlotteur(step(L)) = false
        else
                y(step(L)) = y(L)@pre -1
                classeType(step(L)) = classeType(L)@pre
                enChute(step(L)) = enChute(L)@pre +1
        estFlotteur(step(L)) = estFlotteur(L)@pre
        estExploseur(step(L)) = estExploseur(L)@pre
        attenteConstruction(step(L)) = attenteConstruction(L)@pre
        attenteExplosion(step(L))  = attenteExplosion(L)@pre
        nbDalle(step(L)) = nbDalle(L)@pre
        nbCasse(step(L)) = nbCasse(L)@pre

else if classeType(L) = CREUSEUR then :
        x(step(L)) = x(L)@pre
        direction(step(L)) = direction(L)@pre
        if Level::nature(GameEng::level(gameEng(L)), x(L), y(L)-1) = DIRTY
                y(step(L)) = y(L)@pre +1
                classeType(step(L)) = classeType(L)@pre

        else if Level::nature(GameEng::level(gameEng(L)), x(L), y(L)-1) = METAL then :
                y(step(L)) = y(L)@pre
                classeType(step(L)) = MARCHEUR
        else
                y(step(L)) = y(L)@pre
                classeType(step(L)) = TOMBEUR
        estFlotteur(step(L)) = estFlotteur(L)@pre
        estExploseur(step(L)) = estExploseur(L)@pre
        attenteConstruction(step(L)) = attenteConstruction(L)@pre
        attenteExplosion(step(L))  = attenteExplosion(L)@pre
        nbDalle(step(L)) = nbDalle(L)@pre
        nbCasse(step(L)) = nbCasse(L)@pre

else if (estGrimpeur(step(L))) then :
        if obstacle(L,x(L)+1,y(L)) && obstacle(L,x(L)+1,y(L)+1) &&
                ¬obstacle(L, x(L), y(L)+2) then :
                direction(step(L)) = direction(L)@pre
                classeType(step(L)) = classeType(L)@pre
                y(step(L)) = y(L)@pre +1
                x(step(L)) = x(L)@pre
        else
                if obstacle(L,x(L),y(L) + 2) then :
                        classeType(step(L)) = TOMBEUR
                        direction(step(L)) = direction(L)@pre
                        y(step(L)) = y(L)@pre
                else if obstacle(L,x(L)+1,y(L)) && ¬obstacle(L,x(L)+1,y(L)+1)
                        classeType(step(L)) = MARCHEUR
                        y(step(L)) = y(L)@pre +1
                        x(step(L)) = x(L)@pre +1

else if (classeType(L) = STOPPEUR) then :
        direction(L) = direction@pre
        if ¬obstacle(L, x(L), y(L)-1)  then :
                classeType(step(L)) = TOMBEUR
        else
                classeType(step(L)) = STOPPEUR
        y(L) = y(L)@pre
```

```
            x(L) = x(L)@pre
            direction(L) = direction(L)@pre
            estFlotteur(step(L)) = estFlotteur(L)@pre
            estExploseur(step(L)) = estExploseur(L)@pre
            attenteConstruction(step(L)) = attenteConstruction(L)@pre
            attenteExplosion(step(L))  = attenteExplosion(L)@pre
            nbDalle(step(L)) = nbDalle(L)@pre
            nbCasse(step(L)) = nbCasse(L)@pre

else if (classeType(L) = PELLETEUR) then :
            if obstacle(L, x(L), y(L)-1)  then :
                        if Level::nature(GameEng::level(gameEng(L)), x(L), y(L)) = DIRTY &&
                          Level::nature(GameEng::level(gameEng(L)), x(L), y(L)+1) = DIRTY &&
                          Level::nature(GameEng::level(gameEng(L)), x(L), y(L)+2) = DIRTY &&
                          nbCasse(L) < 12 then :
                                                classeType(step(L)) = PELLETEUR
                                                Level::nature(GameEng::level(gameEng(L)), x(L) +1, y(L)) = EMPTY
                                                Level::nature(GameEng::level(gameEng(L)), x(L) +1, y(L)+1) = EMPTY
                                                Level::nature(GameEng::level(gameEng(L)), x(L) +1, y(L)+2) = EMPTY
                                                nbCasse(step(L)) = nbCasse(L)@pre + 3
                        else
                                    classeType(step(L)) = MARCHEUR
                                    nbCasse(step(L)) = 0
            else
                        classeType(step(L)) = TOMBEUR
                        nbCasse(step(L)) = nbCasse(L)@pre
            estFlotteur(step(L)) = estFlotteur(L)@pre
            estExploseur(step(L)) = estExploseur(L)@pre
            attenteConstruction(step(L)) = attenteConstruction(L)@pre
            attenteExplosion(step(L))  = attenteExplosion(L)@pre
            nbDalle(step(L)) = nbDalle(L)@pre

else if (estExploseur(step(L)) = true) then :
            classeType(step(L)) = classeType(L)@pre
            if attenteExplosion(step(L))>= 5
                        L ∉  GameEng::lemmings(gameEng(L))
                        ∀( x,y) x∈ {  x(L)-2, ..., x(L)+2} et y∈ {  y(L)-1, ..., y(L)+1}
                                    if Level::nature(GameEng::level(gameEng(L)), x, y) != METAL
                                                Level::nature(GameEng::level(gameEng(L)), x, y) = EMPTY
                                    else
                                                Level::nature(GameEng::level(gameEng(L)), x, y) = METAL
                        ∀  ∉  gameEng::lemmings(gameEng(L))
                                    if Lemming::x(l) = x && Lemming::y(l) = y then
                                                l ∉  GameEng::lemmings(gameEng(L))
                                    else
                                                l ∈  GameEng::lemmings(gameEng(L))
            else
                        attenteExplosion(step(L)) = attenteExplosion(L)@pre +1
            estFlotteur(step(L)) = estFlotteur(L)@pre
            estExploseur(step(L)) = estExploseur(L)@pre
            attenteConstruction(step(L)) = attenteConstruction(L)@pre
            nbDalle(step(L)) = nbDalle(L)@pre
            nbCasse(step(L)) = nbCasse(L)@pre

else if (estFlotteur(step(L)) = true) then :
            estFlotteur(step(L)) = estFlotteur(L)@pre
            classeType(step(L)) = classeType(L)@pre
            direction(step(L)) = direction(L)@pre
            y(L) = y(L)@pre
            x(L) = x(L)@pre
            attenteExplosion(step(L)) = attenteExplosion(L)@pre
            estExploseur(step(L)) = estExploseur(L)@pre
            attenteConstruction(step(L)) = attenteConstruction(L)@pre
            nbDalle(step(L)) = nbDalle(L)@pre
            nbCasse(step(L)) = nbCasse(L)@pre
```

```
else if (classeType(L) = CONSTRUTEUR) then :
        if obstacle(L, x(L), y(L)-1)  then :
        ∀( x,y)∈ {(x(L)+1, y(L)),(x(L)+2, y(L)),(x(L)+3, y(L)),(x(L)+1, y(L)+1),(x(L)+2, y(L)+1),(x(L)+2, y(L)+2)}
                if  attenteConstruction(step(L)) = 3 then
                        if Level::nature(GameEng::level(gameEng(L)), x, y) = EMPTY && nbDalle(step(L)) < 12 then :
                                Level::nature(GameEng::level(gameEng(L)), x(L)+2, y(L)) = DIRTY
                                Level::nature(GameEng::level(gameEng(L)), x(L)+3, y(L)) = DIRTY
                            Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)+1) = DIRTY
                                x(L) = x(L)@pre +1
                                y(L) = y(L)@pre +1
                                classeType(step(L)) = CONSTRUCTEUR
                                attenteConstruction(step(L)) = 0
                                nbDalle(step(L)) = nbDalle(L)@pre + 3
                        else
                                classeType(step(L)) = MARCHEUR
                else
                        attenteConstruction(step(L)) = attenteConstruction(step(L))@pre +1
        else
                classeType(step(L)) = TOMBEUR
        attenteExplosion(step(L)) = attenteExplosion(L)@pre
        estExploseur(step(L)) = estExploseur(L)@pre
        nbCasse(step(L)) = nbCasse(L)@pre

else if (classeType(L) = MINEUR) then :
        if obstacle(L, x(L), y(L)-1)  then :
                if ¬obstacle(L, x(L)+1, y(L)) &&
                        Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)+1) != METAL &&
                        Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)+2) != METAL then :
                                classeType(L) = MINEUR
                                Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)+1) = EMPTY
                                Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)+2) = EMPTY
                                y(L) = y(L)@pre +1
                                x(L) = x(L)@pre +1
                else
                        if Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)) != METAL &&
                                Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)-1) != METAL
                                classeType(L) = MINEUR
                                Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)) = EMPTY
                                Level::nature(GameEng::level(gameEng(L)), x(L)+1, y(L)-1) = EMPTY
                                y(L) = y(L)@pre +1
                                x(L) = x(L)@pre -1


                        else
                                classeType(step(L)) = MARCHEUR
        else
                classeType(step(L)) = TOMBEUR
        estFlotteur(step(L)) = estFlotteur(L)@pre
        estExploseur(step(L)) = estExploseur(L)@pre
        attenteConstruction(step(L)) = attenteConstruction(L)@pre
        attenteExplosion(step(L))  = attenteExplosion(L)@pre
        nbDalle(step(L)) = nbDalle(L)@pre
        nbCasse(step(L)) = nbCasse(L)@pre
```