Level

- **Service**: Level
- **Types**: int, bool, enum Nature{DIRT, METAL, EMPTY}
- **Observators**:
  - height: Level $\rightarrow$ int
  - width: Level $\rightarrow$ int
  - editing: Level $\rightarrow$ bool
  - nature: Level * int * int $\rightarrow$ Nature
- **Constructors**:
  - init: $\rightarrow$ Level
- **Operators**:
  - setNature: Level * int * int * Nature $\rightarrow$ Level
    - pre setNature(L, x, y, n) require $0 \leq x \leq$ width(L) $\wedge$ $0 \leq y \leq$ height(L) $\wedge$ editing(L)
  - goPlay: Level $\rightarrow$ Level
  - remove: Level * int * int $\rightarrow$ Level
    - pre remove(L, x, y) require $0 \leq x \leq$ width(L) $\wedge$ $0 \leq y \leq$ height(L) $\wedge$ nature(L, x, y) = DIRT $\wedge$ not(editing(L))
  - build: Level * int * int $\rightarrow$ Level
    - pre build(L, x, y) require $0 \leq x \leq$ width(L) $\wedge$ $0 \leq y \leq$ height(L) $\wedge$ nature(L, x, y) = EMPTY $\wedge$ not(editing(L))
- **Observations**:
  - [invariants]
  - [init]
  - [goPlay]
  - [remove]
  - [build]

GameEng

- **Service**: GameEng
- **Types**: int, bool, enum Nature {DIRT, METAL, EMPTY}, Lemming, Set
- **Observators**:
  - colony : GameEng $\rightarrow$ Set
  - getLemm : GameEng * int $\rightarrow$ Lemming
  - sizeColony : GameEng $\rightarrow$ int
  - spawned : GameEng $\rightarrow$ int

- spawnSpeed : GameEng $\rightarrow$ int
- level : GameEng $\rightarrow$ Level
- tours : GameEng $\rightarrow$ int
- nbSauves : GameEng $\rightarrow$ int
- score : GameEng $\rightarrow$ score
  - pre score(G) require gameOver(G)
- gameOver : GameEng $\rightarrow$ bool
- obstacle : GameEng * int * int $\rightarrow$ bool
  - pre obstacle(G, x, y) require $0 \leq$ x $<$ Level::height(level(G)) $\wedge$ $0 \leq$ y $<$ Level::height(level(G))

- **Constructors**:
  - init : Level, int, int $\rightarrow$ GameEng
    - pre init(lvl, sc, ss) require sc $> 0 \wedge$ ss > 0

- **Operators**:
  - addLemming : GameEng * Lemming $\rightarrow$ GameEng
    - pre addLemming(G, l) require spawned(G) $<$ sizeColony(G)
  - killLemming: GameEng * int $\rightarrow$ GameEng
    - pre killLemming(G, ln) require $0 \leq$ ln $<$ sizeColony(G)
  - saveLemming: GameEng * int $\rightarrow$ GameEng
    - pre saveLemming(G, ln) require $0 \leq$ ln $<$ sizeColony(G)
  - step: GameEng $\rightarrow$ GameEng
  - loadLevel: GameEng * Level * int * int $\rightarrow$ GameEng
    - pre loadLevel(G, lvl, sc, ss) require sc $> 0 \wedge$ ss > 0

- **Observations**:
  - [invariants]
    - gameOver() min= |colony()| == 0
    - score() min= nbSauves() / tours()
    - $0 \leq$ spawned() $<$ sizeColony()
    - $0 \leq$ nbSauves() $<$ sizeColony()
    - obstacle(G,x,y)) min = Level::nature(x,y)!=EMPTY;
  - [init]
    - sizeColony(init(G,sc,ss))=sc
    - spawnSpeed(init(G,sc,ss))=ss
    - spawned(init(G,sc,ss))=0
    - tours(init(G,sc,ss))=0
    - nbSauves(init(G,sc,ss))=0
  - [addLeming]
    - sizeColony(addLeming(G,L,numero))=sizeColony(G)

    - spawnSpeed(addLeming(G,L,numero)=spawnSpeed(G)
    - spawned(addLeming(G,L,numero))=spawned(G)
    - tours(addLeming(G,L,numero)=tours(G)
    - nbSauves(addLeming(G,L,sc,ss))=0
  - [killLeming]
    - sizeColony(killLeming(G,L,numero))=sizeColony(G)
    - spawnSpeed(killLeming(G,L,numero)=spawnSpeed(G)
    - spawned(killLeming(G,L,numero))=spawned(G)
    - tours(killLeming(G,L,numero)=tours(G)
    - nbSauves(killLeming(G,L,sc,ss))=nbSauves(G)
  - [step]
    - sizeColony(step(G))=sizeColony(G)
    - spawnSpeed(step(G))=spawnSpeed(G)
    - tours(step(G))=tours(G)+1
  - [loadLevel]
    - sizeColony(loadLevel(G,L,sc,ss))=sc
    - spawnSpeed(loadLevel(G,L,sc,ss))=ss
    - spawned(loadLevel(G,L,sc,ss))=0
    - tours(loadLevel(G,L,sc,ss))=0
    - nbSauves(loadLevel(G,L,sc,ss))=0

Lemming

- **Service**: Lemming
- **Types**: int, bool, enum Status{WALK, FALL, BUILD, FLOAT, BOMB, STOP, BASH},enum Direction{DROITIER,GAUCHER}
- **Observators**:
  - getX: Lemming → int
  - getY: Lemming → int
  - getNumber: Lemming →int
  - getDir(): Lemming → Direction
  - getStatus(): Lemming → Status
  - timeFalling(): Lemming → int
  - gameEngine(): Lemming → [gameEngine];
- **Constructors**:
  - init: [gameEngine] → Lemming
- **Operators**:
  - changeDir: Lemming → Lemming
  - setStatus: Lemming * Status → Lemming

  - step: Lemming → Lemming

- **Observations**:
  - [invariants]
  - [init]
    - getX(init(Le,G))=gameEngine::entree_X()
    - getY(init(Le,G))=gameEngine::entree_Y()
    - getDir(init(Le,G))=DROITIER;
    - getStatus(init(Le,G))=TOMBEUR;
    - timeFalling(init(Le,G))=0;
  - [changeDir]
    - getX(changeDir(Le))=getX(Le);
    - getY(changeDir(Le))=getY(Le);
    - if(getDir(Le)==DROITIER)then getDir(changeDir(Le))=GAUCHER else getDir(changeDir(Le))=DROITIER;
    - getStatus(changeDir(Le))=getStatus(Le);
    - timeFalling(changeDir(Le))=timeFalling(Le);
  - [setStatus]
    - getX(setStatus(Le,s))=getX(Le);
    - getY(setStatus(Le,s))=getY(Le);
    - getDir(setStatus(Le,s))= getDir(Le);
    - getStatus(setStatus(Le,s))=s;
    - timeFalling(setStatus(Le,s))=timeFalling(Le);
  - [step]