UNIVERSITÉ DE NAMUR

ANALYSE ET MODÉLISATION DES SYSTÈMES DE DONNÉES

TRAVAIL PRATIQUE
Groupe 14

Analyse et modélisation du jeu Golden Quest

Auteurs
Julien CASTIAUX
Kenny WARSZAWSKI

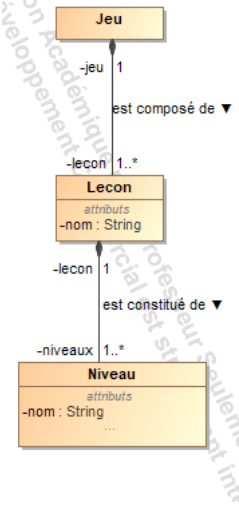
Professeurs
M. Moussa AMRANI
M. Tony LECLERCQ

2 novembre 2018

1 Plateau de Jeu

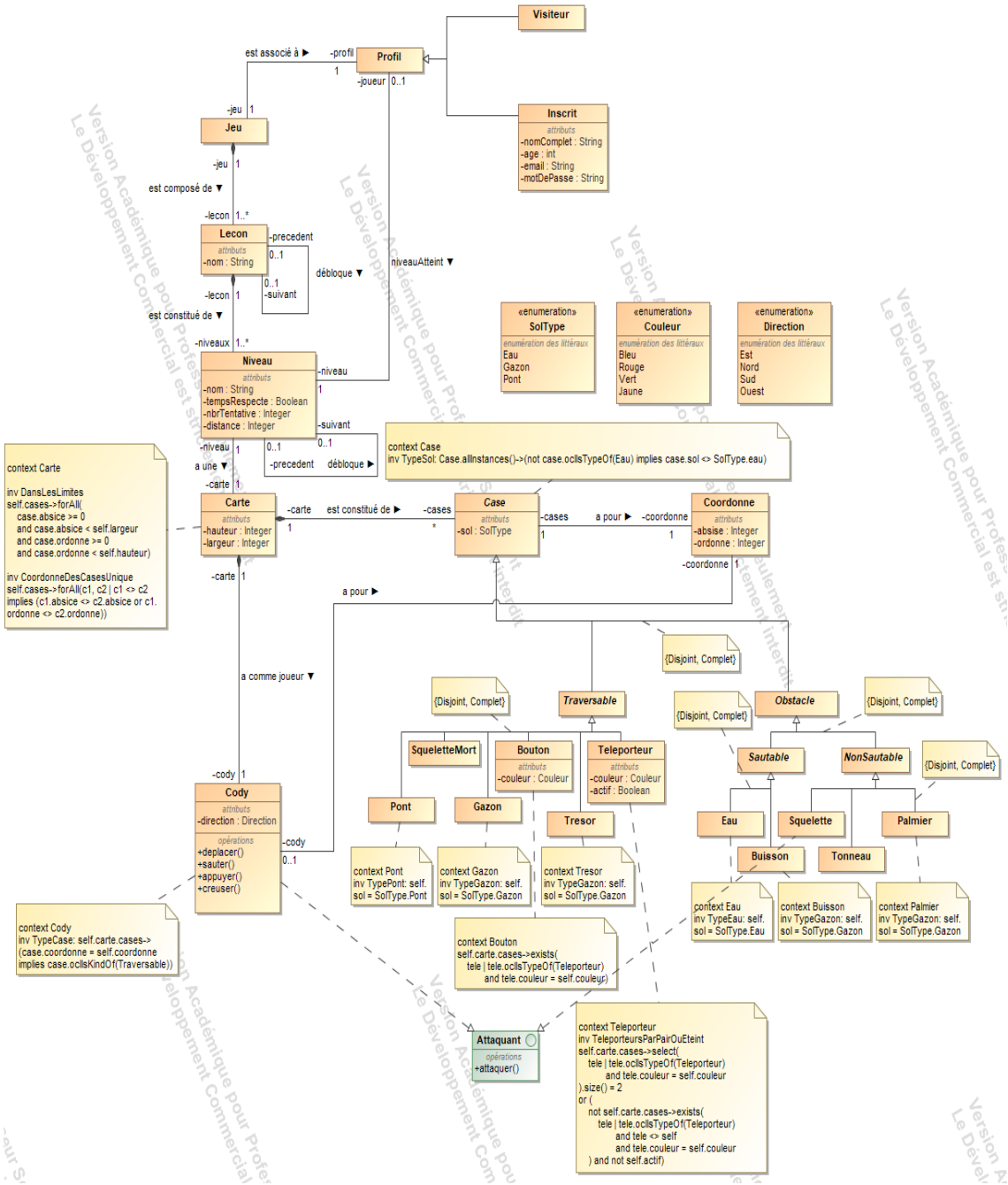
1.1 Question 1

Etablir une première version d'un diagramme de classes Uml qui fixe les éléments principaux : le jeu est constitué d'une série de leçons découpées en niveaux.



1.2 Question 2

Enrichir cette première version avec les détails nécessaires concernant les joueurs et leurs profils, ainsi que les niveaux.


Visiteur
Profil
est associé à ▶
-profil
1
-joueur
0..1
Inscrit
attributs
-nomComplet : String
-age : int
-email : String
-motDePasse : String
-jeu
1
Jeu
est composé de ▼
-lecon
1..*
Lecon
attributs
-nom : String
-precedent
0..1
-suivant
débloque ▼
niveauAtteint ▼
est constitué de ▼
-niveaux
1..*
Niveau
attributs
-nom : String
-tempsRespecte : Boolean
-nbrTentative : Integer
-distance : Integer
-niveau
-suivant
a une ▼
-precedent
débloque ▶
«enumeration»
SolType
enumération des littéraux
Eau
Gazon
Pont
«enumeration»
Couleur
enumération des littéraux
Bleu
Rouge
Vert
Jaune
«enumeration»
Direction
enumération des littéraux
Est
Nord
Sud
Ouest
context Case
inv TypeSol: Case.allInstances()->(not case.oclIsTypeOf(Eau) implies case.sol <> SolType.eau)
context Carte
inv DansLesLimites
self.cases->forAll(
case.absice >= 0
and case.absice < self.largeur
and case.ordonne >= 0
and case.ordonne < self.hauteur)
inv CoordonneDesCasesUnique
self.cases->forAll(c1, c2 | c1 <> c2
implies (c1.absice <> c2.absice or c1.
ordonne <> c2.ordonne))
Carte
attributs
-hauteur : Integer
-largeur : Integer
-carte
1
est constitué de ▶
-cases
*
Case
attributs
-sol : SolType
-cases
1
a pour ▶
-coordonne
1
Coordonne
attributs
-absise : Integer
-ordonne : Integer
-coordonne
1
a pour ▶
{Disjoint, Complet}
a comme joueur ▼
-cody
1
Cody
attributs
-direction : Direction
opérations
+deplacer()
+sauter()
+appuyer()
+creuser()
-cody
0..1
{Disjoint, Complet}
Traversable
{Disjoint, Complet}
Obstacle
{Disjoint, Complet}
SqueletteMort
Bouton
attributs
-couleur : Couleur
Teleporteur
attributs
-couleur : Couleur
-actif : Boolean
Sautable
NonSautable
{Disjoint, Complet}
Pont
Gazon
Tresor
Eau
Squelette
Palmier
Buisson
Tonneau
context Pont
inv TypePont: self.
sol = SolType.Pont
context Gazon
inv TypeGazon: self.
sol = SolType.Gazon
context Tresor
inv TypeGazon: self.
sol = SolType.Gazon
context Eau
inv TypeEau: self.
sol = SolType.Eau
context Buisson
inv TypeGazon: self.
sol = SolType.Gazon
context Palmier
inv TypeGazon: self.
sol = SolType.Gazon
context Cody
inv TypeCase: self.carte.cases->
(case.coordonne = self.coordonne
implies case.oclIsKindOf(Traversable))
context Bouton
self.carte.cases->exists(
tele | tele.oclIsTypeOf(Teleporteur)
and tele.couleur = self.couleur)
Attaquant
opérations
+attaquer()
context Teleporteur
inv TeleporteursParPairOuEteint
self.carte.cases->select(
tele | tele.oclIsTypeOf(Teleporteur)
and tele.couleur = self.couleur
).size() = 2
or (
not self.carte.cases->exists(
tele | tele.oclIsTypeOf(Teleporteur)
and tele <> self
and tele.couleur = self.couleur
) and not self.actif)

1.3 Question 3

Spécifier les contraintes suivantes, soit à l'aide d'éléments structurels dans le diagramme, éventuellement complétées par des contraintes en Ocl (ceci dépend de la manière dont le diagramme de classes est construit) :

1. Les coordonnees d'une case ne peuvent excéder les dimensions du niveau

```
context Carte
inv DansLesLimites
  self.cases->forAll(
    case.coordonne.absice >= 0
    and case.coordonne.absice < self.largeur
    and case.coordonne.ordonne >= 0
    and case.coordonne.ordonne < self.hauteur)
```

2. Chaque niveau ne contient qu'un seul Cody, et qu'un seul coffre

```
context Carte
self.cases->select(
    case | case.oclIsTypeOf(Tresor)
).size() = 1
```

   En ce qui concerne Cody, la multiplicité répond à cette contrainte.

3. Un personnage ne peut pas se trouver sur un obstacle
4. Deux personnages ne peuvent pas partager la même case

```
context Cody
inv TypeCase: self.carte.cases.forAll->(
case.coordonne = self.coordonne
   implies case.oclIsKindOf(Traversable)
)
```

```
context Carte
inv CoordonneDesCasesUnique
```

```
self.cases->forAll(
    c1, c2 | c1 <> c2
    implies (c1.absice <> c2.absice
             or c1.ordonne <> c2.ordonne))
```

Nous considérons que les squelettes sont des cases de type Obstacle. Il est donc impossible qu'un squelette se trouve sur un obstacle du fait de l'unicité des cases reprise par la contrainte OCL ci-dessus. Cody doit être sur une case Traversable et donc ne peut pas être sur un squelette.

5. Le coffre doit se trouver sur du gazon

```
context Tresor
inv TypeGazon: self.sol = SolType.Gazon
```

6. Un niveau doit toujours comporter deux tunnels de teleportation de même couleur, ou le tunnel unique doit être initialement fermé

```
context Teleporteur
inv TeleporteursParPairOuEteint
self.carte.cases->select(
       tele | tele.oclIsTypeOf(Teleporteur)
         and tele.couleur = self.couleur).size() = 2
     or (
       not self.carte.cases->exists(
           tele | tele.oclIsTypeOf(Teleporteur)
             and tele <> self
             and tele.couleur = self.couleur
) and not self.actif)
```

7. Un levier de téléportation ne peut être présent que s'il existe des tunnels de la même couleur

```
context Bouton
self.carte.cases->exists(
   tele | tele.oclIsTypeOf(Teleporteur)
       and tele.couleur = self.couleur)
```

8. Un palmier ou un buisson doivent obligatoirement être posé sur du gazon (sinon, il ne peut pas pousser).

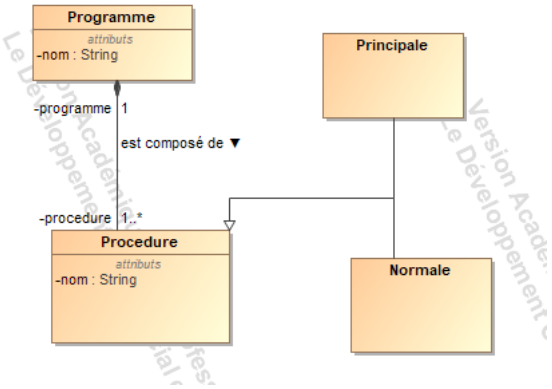
```
context Buisson
inv TypeGazon: self.sol = SolType.Gazon

context Palmier
inv TypeGazon: self.sol = SolType.Gazon
```

2 Langage Play

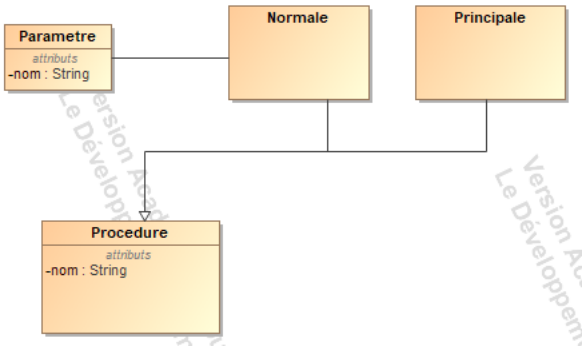
2.1 Question 5

Etablir une première version d'un diagramme de classe Uml qui fixe les éléments principaux :un Program(me) Play est un ensemble de procédures (dont l'une est la procédure principale).



2.2 Question 6

Modéliser le concept de procédure à partir d'une classe Procedure.



2.3 Question 7

Modéliser le concept d'expression comme indiqué en Section 3.3, à partir d'une classe Expression.