TRAVAUX DIRIGÉS - OCL - PARTIE 2 (AVEC COLLECTIONS)

1 Contraintes sur un modèle : les Tortues

Nous continuons l'étude du modèle de classes pour un système d'information dédié à une association de sauvegarde des tortues terrestres. Les tortues sont décrites par leur nom, âge, taille, sexe, dates auxquelles elles pondent, si elles vivent en captivité, le nom latin de leur espèce, leur taille maximale à l'âge adulte, leur alimentation actuelle et leur alimentation possible. Elles peuvent changer de taille (en vieillissant) et manger des aliments. On peut déterminer par un prédicat si un type d'aliment leur est autorisé. Une espèce de tortue a une répartition géographique constituée de différents lieux. Une tortue habite en un lieu donné. Ce lieu se caractérise par un type de milieu (garrigue, montagne, maquis, désert, etc.). Une espèce de tortue admet un mode d'élevage qui précise la température de jour et la température de nuit, si ce peut être dans un terrarium, si ce peut être en plein air. Une espèce de tortue se caractérise par des particularités biologiques incluant un comportement, si l'espèce hiberne, et un régime alimentaire général composé de types d'aliments admis. Le mode d'élevage précise aussi les types d'aliments utilisés en captivité (régime alimentaire de captivité).

Question 1.1 *Nous complétons la modélisation présentée figure 1 en y ajoutant de nouvelles contraintes :*

1. *une tortue mâle ne peut avoir de dates de ponte ;*
2. *une tortue habite l'un des lieux de la répartition géographique de son espèce ;*
3. *tout aliment utilisable pour l'élevage en captivité fait partie du régime alimentaire général ;*
4. *écrire la post-condition de l'opération* `nourrituresPossible(t:TypeAliment):boolean` *de la classe* `espèceTortue` *;* `t` *est un un type d'aliment possible s'il fait partie du régime alimentaire général de l'espèce.*
5. *écrire la pré-condition de l'opération* `mange(a:Aliment)` *de la classe* `Tortue` *; une tortue ne peut manger que des aliments prévus par un des régimes alimentaires de son espèce (régime général ou de captivité), il faut donc tester si la tortue est captive ou non pour connaître le régime alimentaire et savoir si* `a` *est admissible.*

Réponse 1.1

```
context Tortue inv:
  sexe = #M implies datesPonte->isEmpty()
  espèce.repartitionGéographique.lieu->includes(habite)

context EspeceTortue inv:
  modeElevage.typeAliment->forAll(
      t :TypeAliment | biologie.typeAliment->includes(t))
```

context EspeceTortue::nourrituresPossible(t :TypeAliment)
post: result=biologie.typeAliment->includes(t)

context Tortue::mange(a :Aliment)
pre:
if captive
then espèce.modeElevage.typeAliment->includes(a.typeAliment)
else espèce.biologie.typeAliment->includes(a.typeAliment)
endif

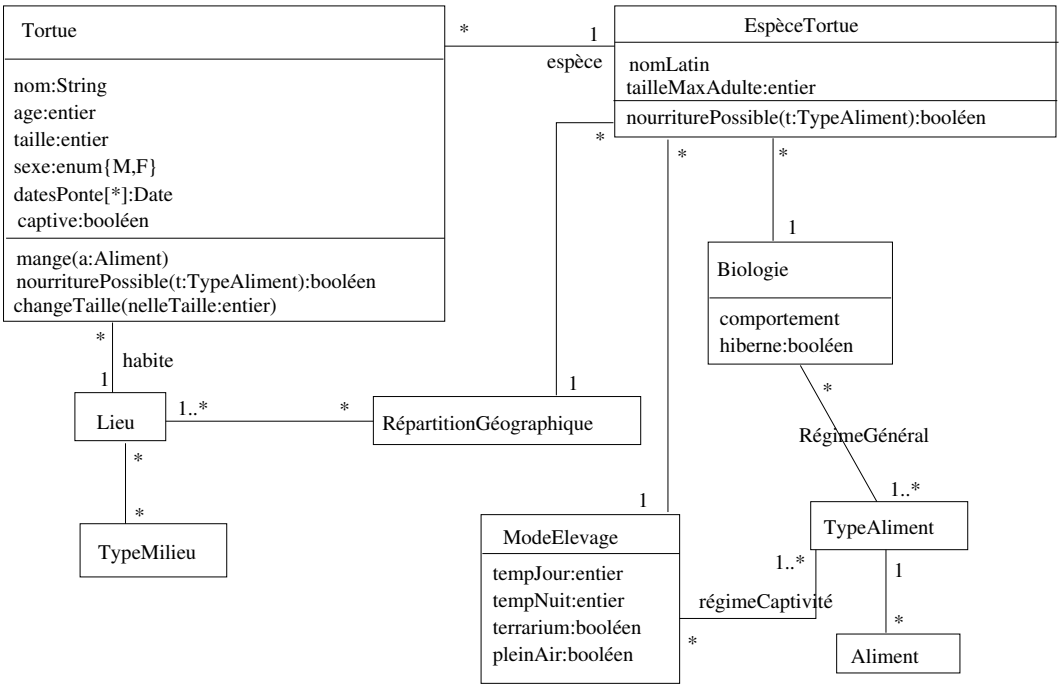

FIGURE 1 – Des tortues

2 Contraintes sur un métamodèle : UML2.5

Vous trouverez les solutions aux pages indiquées ci-dessous sous la forme *p. xx (yy)* pour la page *xx* du document qui est aussi la page *yy* du pdf. Attention, les contextes des contraintes ne sont pas rappelés dans le document, le contexte d'une contrainte est l'élément qui est décrit. S'il n'y a pas de page, c'est qu'il n'y a pas de telle contrainte dans le document actuel (qui est un draft). Le document de référence est :

— OMG Document Number : formal/2017-12-05
— Normative Reference : https/www.omg.org/spec/UML/

Root Diagram

— La requête `allOwnedElements:Element [0..*]` donne les éléments possédés directement ou indirectement par un autre élément. **p. 43 (85)**

— Un élément ne peut pas se contenir lui-même, que ce soit directement ou indirectement. **p. 44 (86)**

Classifier Diagram

— La requête `parents():Classifier [0..*]` donne tous les successeurs immédiats (généralisations) d'un classifier. **p. 135 (177)**
— La requête `allParents():Classifier [0..*]` donne tous les successeurs (généralisations) d'un classifier. **p. 134 (176)**
— La requête `conformsTo(other:Classifier):boolean` est vraie si et seulement si `other` est une généralisation du classifieur. **p. 134 (176)**
— `/general` est égal aux successeurs immédiats (je ne l'ai pas trouvée)
— Il n'y a pas de circuit dans les généralisations **p. 136 (178)**

PowerType Diagram

— Toutes les généralisations associées à un `GeneralizationSet` doivent avoir le même super-classifieur. **p. 139 (182)**

Packages Diagram

— Un PackageableElement qui est possédé par un Namespace doit avoir une visibilité. **p. 53 (95)**
— Un élément possédé par un package et qui a une visibilité est soit public, soit privé **p. 277 (320)**