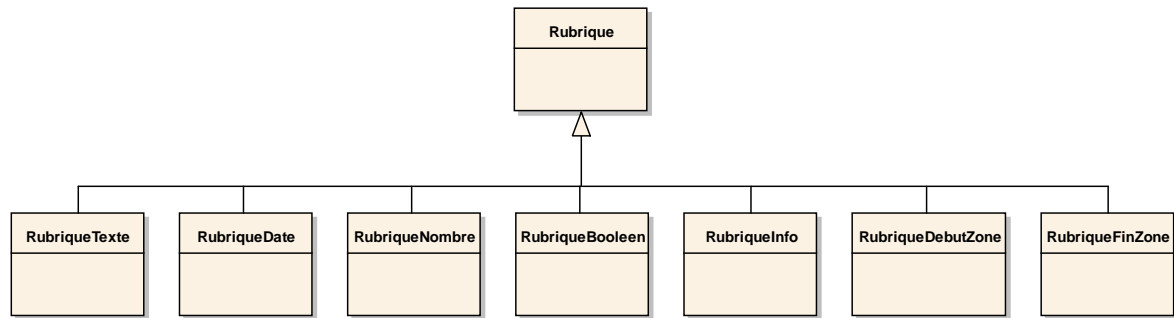


TD UML 2013 - Emmanuel Pichon

Conception : héritage

Eléments de corrigé

1 Première approche (souvent rencontrée lors de l'analyse)



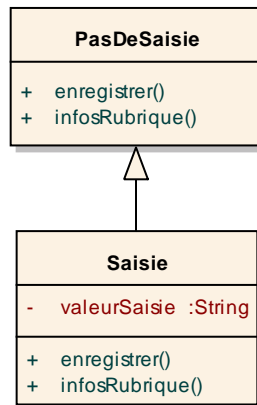
Cet arbre d'héritage n'est ni bon ni mauvais. Il est construit à partir d'un raisonnement par nature (ce que sont les choses). On ne peut pas savoir s'il permettra une bonne factorisation des opérations et des attributs lors de la conception et du codage.

2 Deuxième approche : héritage au niveau de la classe Saisie

2.1 Usage et comportement de chaque opération + usage de chaque attribut

| | | opérations | | attribut |
|----------------------|-----------------------------|-----------------------------------|----------------|--------------|
| | | enregistrer (la valeur saisie) | infosRubriques | valeurSaisie |
| Rubrique avec saisie | texte | X | X | X |
| | date | X | X | X |
| | nombre avec unité | X | X | X |
| | booléen | X | X | X |
| Rubrique sans saisie | information (pas de saisie) | - | X | - |
| | début de zone | - | X | - |
| | fin de zone | - | X | - |

2.2 Arbre d'héritage correspondant



Nota bene

- toutes les opérations publiques doivent être définies dans la super classe car elles peuvent être appelées pour toutes les classes de cet arbre d'héritage.

Classe PasDeSaisie

- enregistrer ne fait rien.
- infosRubrique ne retourne aucune valeur saisie.
- infosRubrique appelle super.infosRubrique.

Classe Saisie

- enregistrer stocke la valeur saisie par le professionnel dans valeurSaisie (en passant par un setteur pour respecter l'encapsulation).
- infosRubriques est redéfinie à ce niveau pour ajouter la valeur saisie contenue dans l'attribut valeurSaisie (en passant par un getteur pour respecter l'encapsulation).

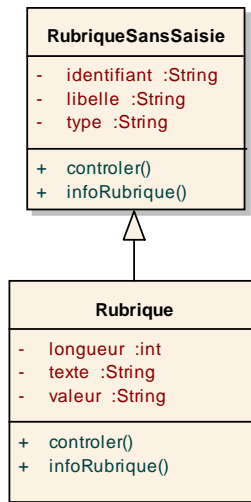
3 Deuxième approche : héritage au niveau de la classe Rubrique

3.1 Usage et comportement de chaque opération + usage de chaque attribut

| type de rubrique | opérations | | attributs | | | | | | |
|-------------------------|---------------------------------|----------------|-------------|---------|----------|---|------|----------------------|---|
| | contrôler (la valeur saisie) | infosRubriques | identifiant | libelle | longueur | texte indiquant que la valeur est obligatoire | type | valeur par défaut | |
| Rubrique avec saisie | texte | X | X | X | X | X | X | X | X |
| | date | X | X | X | X | X | X | X | X |
| | nombre avec unité | X | X | X | X | X | X | X | X |
| | booléen | X | X | X | X | - | X | X | X |
| Rubrique sans saisie | information (pas de saisie) | toujours OK | X | X | - | - | X | - | - |
| | début de zone | toujours OK | X | X | - | - | X | - | - |
| | fin de zone | toujours OK | X | X | - | - | X | - | - |

On fait le choix de ne pas traiter ce point par l'héritage car il s'agit d'une exception localisée.

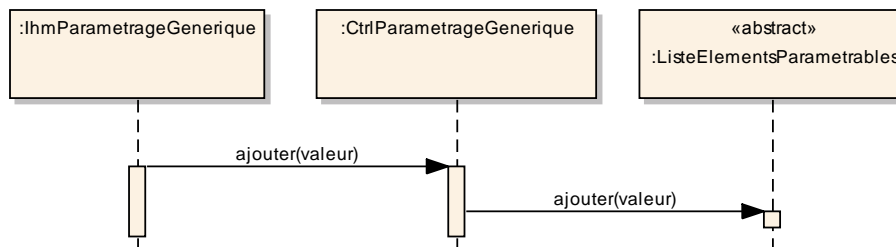
3.2 Arbre d'héritage correspondant



RubriqueSansSaisie n'a pas besoin des attributs longueur, texte et valeur.

4 Concevoir un écran générique de paramétrage (hors e-CERFA)

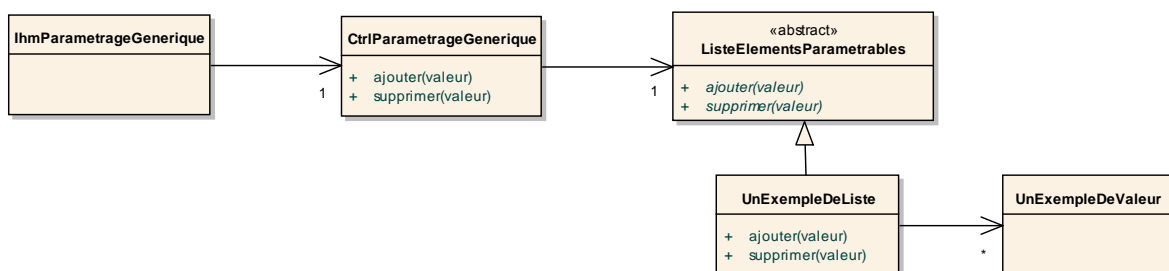
4.1 Diagramme de séquence « ajouter un élément »



L'écran fonctionne ainsi pour tout type de liste sous-classe de ListeElementsParametrables.

Le même diagramme peut être construit pour supprimer(valeur).

4.2 Diagramme de classes



La classe ListeElementsParametrables est abstraite. Elle doit être spécialisée.

La classe UnExempleDeListe redéfinit ajouter et supprimer pour les adapter à son contexte (règles d'ajout et de suppression spécifiques).

On pourra ajouter d'autres sous-classes sur le même principe sans modifier le système existant.