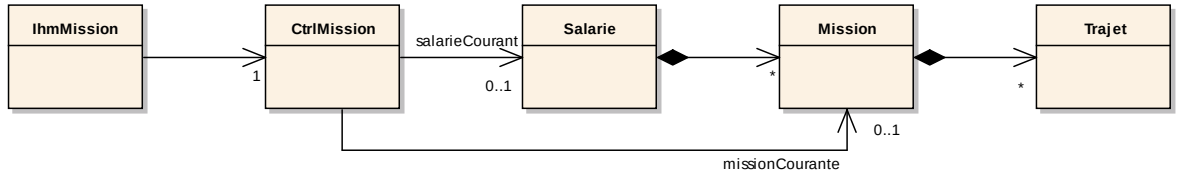
Méthodologie de conception objet
Emmanuel Pichon
Examen n°2 – 18/12/2013

Durée : 1h15.
Documents autorisés : 1 feuille A4 recto/verso de notes manuscrites ou dactylographiées.
Thème : conception du logiciel de saisie de missions et de remboursement des frais de mission.
Barème donné à titre indicatif.
Niveau de notation UML demandé : niveau esquisse avec distinction objet/classe, public/privé, association/héritage, message/message retour/création d'objet.

Exercice n°1 : diagramme de séquence de conception du cas d'utilisation « Saisir une mission » (8 points)

Construire un diagramme de séquence contenant les objets créés à partir des classes suivantes :



NB : le lien `salarieCourant` a déjà été positionné lors du login du salarié dans le logiciel.

Faire apparaitre dans le diagramme de séquence tous les messages cités ci-dessous.

- Le message `creer` déclenché par l'acteur `Salarié` permet, par délégation, de :
  - créer une mission,
  - positionner le lien `missionCourante` (message interne `setMissionCourante`),
  - afficher une page de saisie d'une mission (message interne `afficherPageSaisie`).
- Le message `ajouterTrajet` déclenché par l'acteur `Salarié` permet, par délégation, de :
  - créer un trajet,
  - afficher la liste des trajets mise à jour (message interne `afficherListeTrajets`).
- Le message `enregistrer` déclenché par l'acteur `Salarié` permet, par délégation, de :
  - passer l'état de la mission à la valeur « à valider » (message interne `setEtat`),
  - afficher une confirmation (message interne `afficherConfirmation`).

Faire apparaitre trois messages de retour vers l'acteur `Salarié` en indiquant ce qui est affiché.
Positionner un fragment indiquant une boucle permettant de traiter plusieurs trajets.
Ne pas indiquer les paramètres des messages.

Exercice n°2 : diagramme de classes et encapsulation (3 points)

Compléter le diagramme de classes de l'exercice précédent avec :

- les opérations correspondant aux messages apparaissant dans le diagramme de séquence,
- l'attribut `etat` dans la classe Mission.

Dans le diagramme, indiquer la visibilité (public ou privé) de ces éléments et des rôles `salarieCourant` et `missionCourante` en appliquant le principe d'encapsulation. Ne pas indiquer le type des attributs.

Exercice n°3 : spécialisation (6 points)

Chaque trajet est caractérisé par trois attributs (`montant`, `description`, `reference`), et par un lieu de départ et un lieu d'arrivée représentés par la classe `Lieu`.

Afin de réduire la charge de travail des responsables hiérarchiques concernant la validation des missions, une validation automatique est demandée pour les missions ne contenant que des trajets en train. Au niveau de chaque trajet, l'opération correspondante est `validerAutomatique`.
Cette opération retourne « vrai » pour un trajet en train et « faux » pour les autres types de trajet.

Au niveau de chaque trajet, l'opération permettant de calculer le montant à rembourser est `calculerMontant`. Ce montant correspond au montant des billets ou des factures pour tous les types de trajet (attribut `montant`). Pour les trajets en voiture, il faut ajouter à ce montant un forfait kilométrique multiplié par la distance parcourue (attribut `distance`).

Construire un diagramme de classes présentant les classes `Mission`, `Trajet`, `Lieu` et les sous classes de la classe `Trajet`. Positionner dans le diagramme :

- les opérations `validerAutomatique`, `calculerMontant` et leurs redéfinitions,
- les attributs `montant`, `description`, `reference` et `distance`,
- les relations d'héritage, les associations, les rôles et les cardinalités.

Dans le diagramme, indiquer la visibilité (public/privé) des opérations, des attributs et des rôles en appliquant le principe d'encapsulation. Ne pas indiquer le type des attributs.
NB : le forfait kilométrique n'est pas modélisé dans cet exercice.

Hors diagramme : si l'on souhaite ajouter des *getteurs* et des *setteurs* pour accéder aux attributs, indiquer quelle devrait être la visibilité des *getteurs* et des *setteurs* selon le principe d'encapsulation.

Exercice n°4 : généralisation (3 points)

Votre logiciel de gestion des missions étant un succès, on vous demande d'étendre ce logiciel à la gestion des achats réalisables par les salariés (fournitures, abonnements, …).

Dans cet exercice, la classe `Mission` est renommée `Achat` pour correspondre au nouveau périmètre du logiciel.

Une classe `Article` est ajoutée pour généraliser la classe `Trajet`. L'opération `calculerMontant` retourne la valeur de l'attribut `montant` pour tous les types d'article sauf pour les trajets en voiture. L'opération `validerAutomatique` retourne « faux » pour les types d'article sauf pour les trajets en train. Contrairement aux trajets, les articles n'ont ni lieu de départ ni lieu d'arrivée.

L'objectif de l'exercice est de répartir entre la classe `Article` et la classe `Trajet` les attributs, les opérations et les associations initialement localisés dans la classe `Trajet` (cf. exercice précédent).

Construire un diagramme de classes présentant les classes `Achat`, `Article`, `Lieu`, `Trajet` et ses sous-classes. Positionner dans le diagramme leurs relations, leurs attributs et leurs opérations.

Dans le diagramme, indiquer la visibilité (public/privé) des opérations, des attributs et des rôles en appliquant le principe d'encapsulation. Ne pas indiquer le type des attributs.