Examen Programmation Orientée Objet

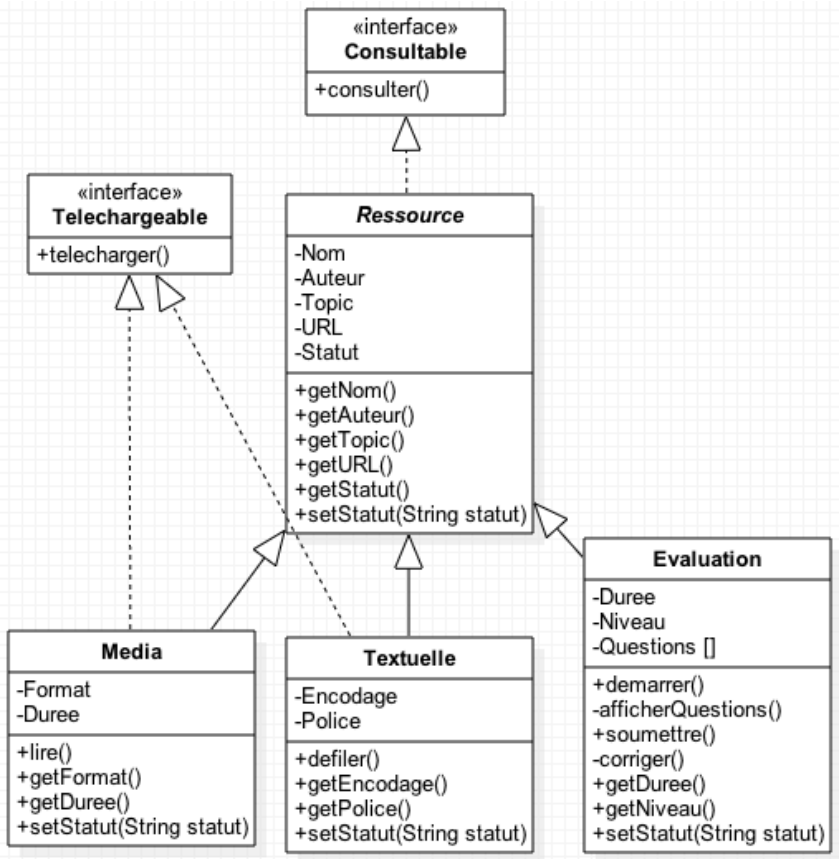
M2 CCI Tours, le 19 décembre 2017

Durée : 2h

Document de cours et accès Celene autorisés. Communication entre pairs non autorisée.

Instructions : Créez un nouveau projet Eclipse à votre nom et prénom (par exemple DupontJean). A la fin de l'épreuve, exportez votre projet avec Eclipse en une archive **zip** du même nom que le projet (Exemple DupontJean.zip) et déposez cette archive sur Celene.

Sujet : On souhaite implémenter un programme Java pour gérer l'activité d'un site d'apprentissage en ligne. On considère comme point de départ le diagramme de classe simplifié suivant, qui représente les ressources pédagogiques gérées par ce site :



Partie 1 (12 points) :

Créer un package *ressources*. Dans ce package, créer les classes décrites par le diagramme ci-dessus en respectant les principes d'héritage et d'encapsulation et en suivant les indications ci-après.

1- (0,5 point) Une interface ***Consultable*** qui définit une méthode *consulter()* sans type de retours.

2- (0,5 point) Une interface ***Telechargeable*** qui définit une méthode *telecharger()* sans type de retours**.**

3- (3,5 points) Une classe abstraite ***Ressource*** avec les attributs *Nom, Auteur, Topic, URL et Statut,* tous de type String.

La classe ***Ressource*** fournit deux constructeurs. Le premier constructeur prend en paramètre une chaine de caractères représentant l'URL de la ressource et qui sera également affectée au nom de la ressource. Les attributs restants sont initialisés à des valeurs par défauts (sous la forme d'une chaine vide). Le deuxième constructeur prend en paramètre 5 chaines de caractères afin d'initialiser tous les attributs de la classe, en respectant l'ordre donné sur le diagramme de classes.

La classe ***Ressource*** fournit également un *getter* pour chaque attribut ainsi qu'un *setter* pour l'attribut *Statut*.

La classe ***Ressource*** réalise l'interface ***Consultable***. La méthode *consulter()* est cependant abstraite et devra être implémentée de manière concrète selon la ressource (Media, Textuelle ou Evaluation).

4- (2 points) Une classe **Media** qui étend la classe ***Ressource*** et fournit une implémentation des méthodes *setStatut(String statut) et consulter().*

L'implémentation de *setStatut(String statut)* consiste à faire appel à celle fournie dans la classe mère en lui passant l'une des constantes suivante : *TELECHARGEE*, *CONSULTEE*, *LUE* ou *NON_CONSULTEE*. Ces constantes peuvent être définies au niveau de la classe ***Ressource***.

L'implémentation de la méthode *consulter()* consiste à faire appel à la méthode *setStatut(String statut),* en lui passant comme paramètre la constante *CONSULTEE,* puis à la méthode *lire().*

La classe **Media** réalise l'interface ***Telechargeable***. L'implémentation de la méthode *telecharger()* consiste à faire appel à la méthode *setStatut(String statut)* en lui passant comme paramètre la constante *TELECHARGEE.*

La classe **Media** possède les attributs *Format* de type *String* et *Duree* de type *int* et fournit un *getter* pour chacun des deux attributs. Elle fournit également un constructeur avec 3 paramètres correspondant à l'URL, le format et la durée et un constructeur avec 7 paramètres correspondants à tous les attributs de **Media** (propres et hérités).

La classe **Media** fournit une méthode *lire()* qui consiste à lancer la lecture de la ressource média dans l'application correspondante au format de la ressource. Lorsque la lecture va jusqu'à son terme (durée), le statut de la ressource est passé à *LUE.* Pour des raisons de simplification on se contentera d'afficher un message « Lecture de la ressource média + url » puis d'appeler la méthode *setStatut(String statut)* avec la constante *LUE* comme paramètre. Cette constante peut être définie au niveau de la classe ***Ressource***.

5- (2 points) Une classe **Textuelle** qui étend la classe ***Ressource*** et fournit une implémentation des méthodes *setStatut(String statut) et consulter().*

L'implémentation de *setStatut(String statut)* consiste à faire appel à celle fournie dans la classe mère.

L'implémentation de la méthode *consulter()* consiste à faire appel à la méthode *setStatut(String statut),* en lui passant comme paramètre la constante *CONSULTEE,* puis à la méthode *defiler().*

La classe **Textuelle** réalise l'interface ***Telechargeable***. L'implémentation de la méthode *telecharger()* consiste à faire appel à la méthode *setStatut(String statut)* en lui passant comme paramètre la constante *TELECHARGEE.*

La classe **Textuelle** possède les attributs *Encodage* et *Police* de type *String* et fournit un *getter* pour chacun des deux attributs. Elle fournit un constructeur avec 3 paramètres correspondant à l'URL,

l'encodage et la police et un constructeur avec 7 paramètres correspondants à tous les attributs de Textuelle (propres et hérités).

La classe **Textuelle** fournit une méthode *defiler()* qui consiste à défiler progressivement le texte dans le navigateur correspondant afin de simuler la lecture de son contenu. Lorsque la lecture va jusqu'à son terme (défilement impossible), le statut de la ressource est passé à *LUE.* Pour des raisons de simplification on se contentera d'afficher un message « Lecture de la ressource textuelle + url » puis d'appeler la méthode *setStaut* avec la constante *LUE* comme paramètre.

6- (3,5 points) Une classe **Evaluation** qui étend la classe ***Ressource*** et fournit une implémentation des méthodes *setStatut(String statut) et consulter()*. L'implémentation de *setStatut(String statut)* consiste à faire appel à celle fournie dans la classe mère. L'implémentation de la méthode *consulter()* consiste à faire appel à la méthode *setStatut(String statut),* en lui passant comme paramètre la constante *CONSULTEE,* puis à la méthode *demarrer().*

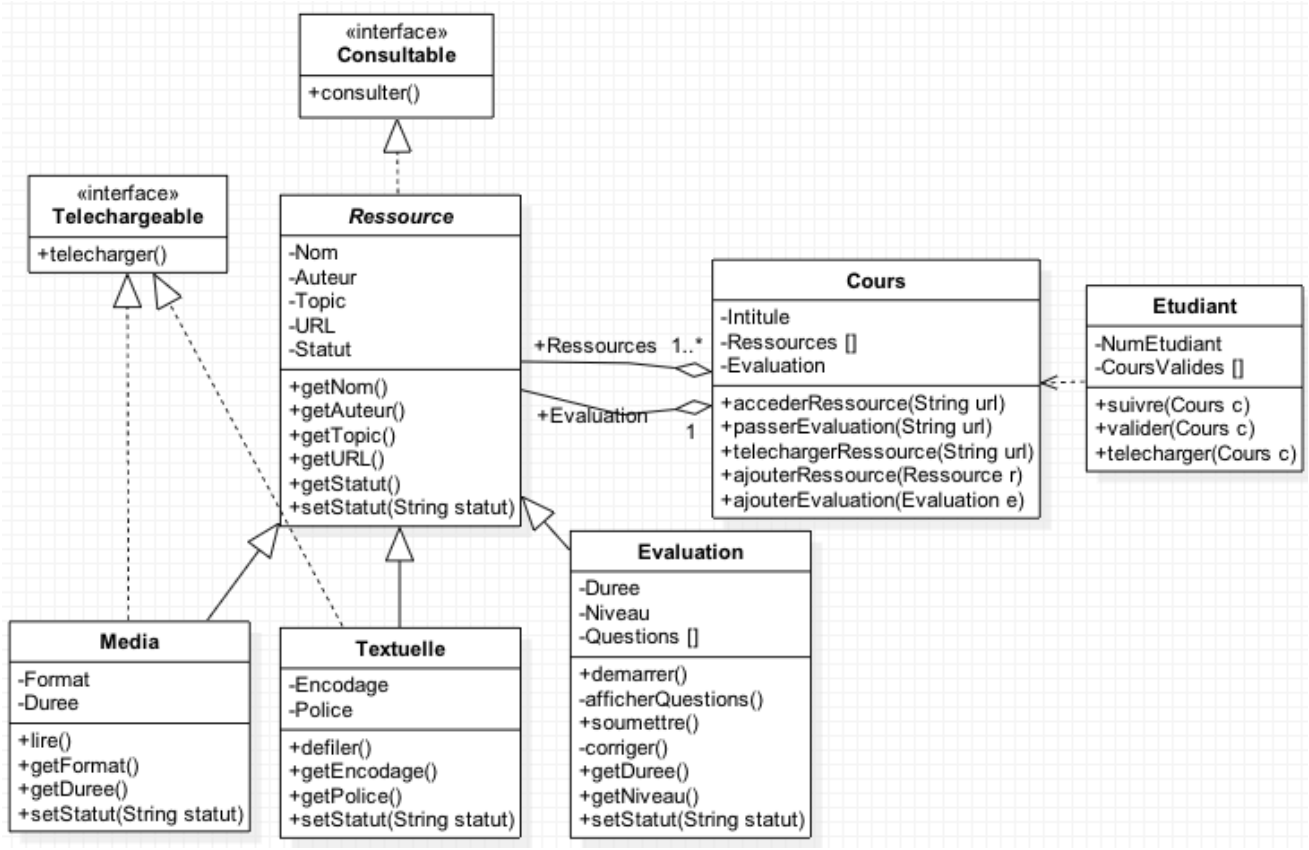
La classe **Evaluation** possède les attributs *Duree* de type *int*, Niveau de type *int* et Questions de type tableau de String et fournit un *setter* pour chacun des ces attributs. Elle fournit également un constructeur avec 4 paramètres correspondant à l'URL, la durée, le niveau et les questions de l'évaluation et un constructeur avec 8 paramètres correspondants à tous les attributs de **Evaluation** (propres et hérités).

La classe **Evaluation** fournit une méthode *demarrer()* qui consiste à lancer l'évaluation en affichant les questions dans le navigateur adéquat. Lorsque le test va à son terme (épuisement des questions) on pourra faire appel à la méthode *soumettre()*. La méthode soumettre vérifie le respect de la durée du test et appelle la méthode corriger pour vérifier l'ensemble des réponses aux questions du test. En cas de succès (pourcentage à définir) la méthode corrigée fait appel à la méthode setStatut() en lui passant comme paramètre la constante *VALIDEE*. Autrement, elle appellera la même méthode avec la constante *NON_VALIDEE*. Lorsque la durée n'est dépassée, la méthode soumettre appelle directement la méthode setStatut avec la constante NON_VALIDEE. Pour des raisons de simplification on se contentera d'afficher les messages « Activation du test + url », « Soumission du test + url » et « correction du test + url » dans ces trois méthodes, respectivement. Les constantes évoquées peuvent être déclarées au niveau de la classe ***Ressource***.

Partie 2 (6 points) :

On souhaite compléter le programme en respectant le diagramme de classe ci-après. Dans cette modélisation, un cours est caractérisé par son intitulé, une collection de ressources Media et Textuelle ainsi qu'une ressource Evaluation. La création d'un cours se fera à travers un unique constructeur qui prend en paramètre l'intitulé du cours. Les différentes ressources pourront être ajoutées au cours en utilisant les méthodes d'ajout nécessaires.

La validation d'un cours par un étudiant consiste à accéder à l'ensemble des ressources (méthode *suivre*) et à la validation de l'évaluation (méthode *valider*). Tout cours validé est ensuite ajouté à la collection des cours validés par l'étudiant. Le téléchargement d'un cours consiste à télécharger l'ensemble de ses ressources téléchargeables. Il ne donne pas lieu à la validation du cours. L'instanciation de la classe Etudiant se fera à travers un constructeur unique qui prend en paramètre le numéro de l'étudiant.



Créer un package **cours** et y implémenter les classes **Cours (3 points)** et **Etudiant (3 points)** en suivant le diagramme de classes et la description textuelle ci-dessus.

Partie 3 (2 points)

Créer un package ***main***. Dans ce package, créez la classe **FormationEnLigne** contenant une méthode *main()* pour illustrer le fonctionnement des implémentations précédentes.