Examen : durée 2h

CONSIGNES

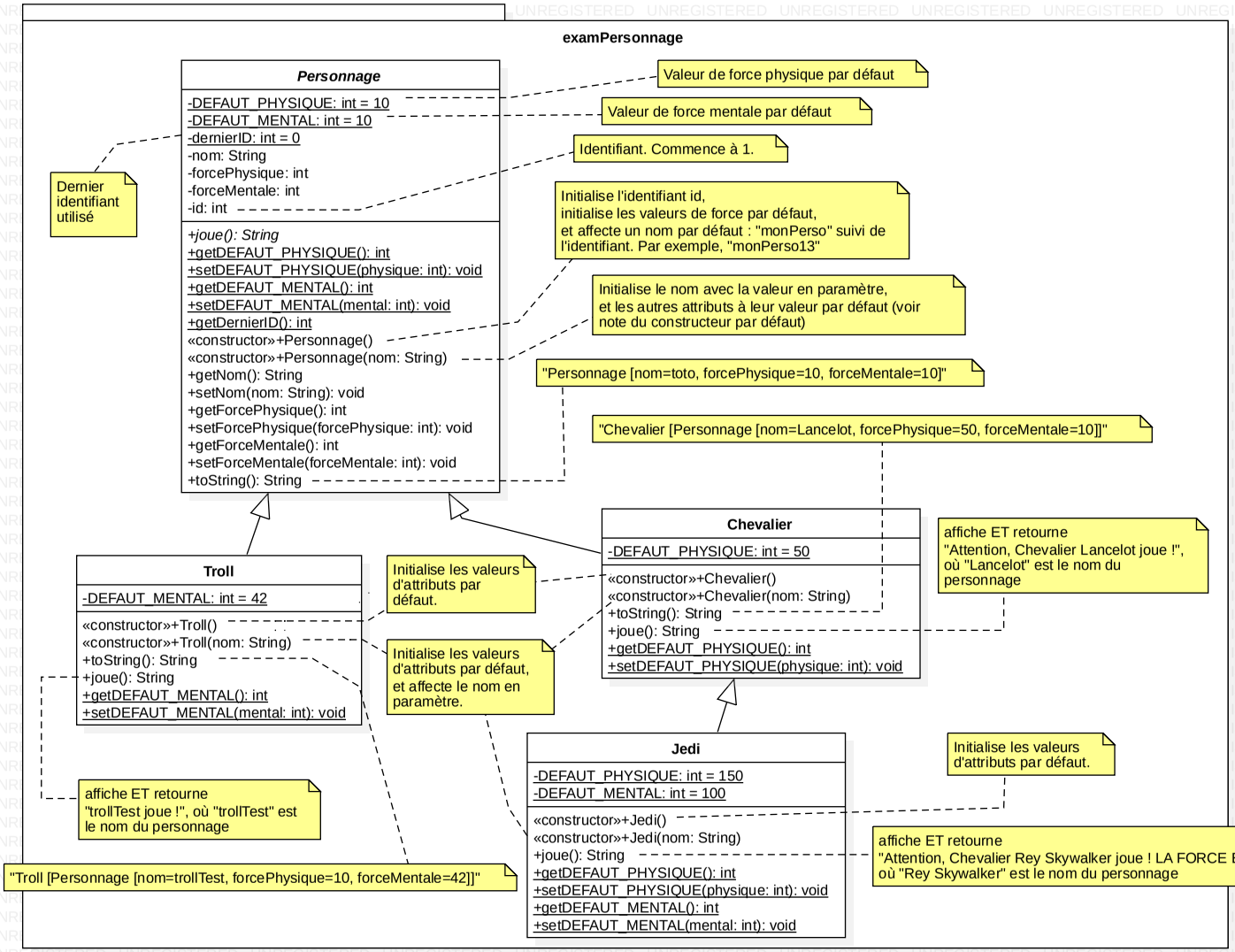
- L'examen a lieu sur machine. L'environnement de travail est le même que pour un TP classique.
- **Vous devez travailler seul.e.s** ! Toute communication avec un tiers est strictement INTERDITE, que ce soit avec quelqu'un dans la salle, quelqu'un dans une autre salle, ou quelqu'un de l'extérieur. Les applications de communication sont donc toutes strictement interdites, y compris le mail, discord, chatbox, etc.. Les sites web permettant une communication directe avec des personnes tierces sont également interdits. Toute utilisation de l'une de ces applications ou site web sera considérée comme un cas de fraude, et traitée comme tel.
- Chaque poste de travail est sous la surveillance continue du logiciel veyon, pendant toute la durée de l'examen.
- **Tout matériel électronique** autre que celui fourni par l'université est **strictement interdit**.
- Aucun document papier autorisé.
- **Le temps est limité à 2h** (ou 2h40 pour les tiers-temps). Aucun rendu supplémentaire ne sera accepté au-delà de cette limite.
- Le travail doit être déposé comme d'habitude sur le dépôt github que vous aurez cloné depuis le lien classroom suivant : `https://classroom.github.com/a/xrgJ8rQZ`
- Le code source doit être déposé sour la forme de fichiers `.java`. Le travail déposé uniquement sous forme illisible de byte code (`.class`) sera tout simplement ignoré.
- Le barème est donné à titre indicatif, et pourra varier légèrement.
- **TRÈS IMPORTANT !** Vos programmes seront évalués par des programmes : il est donc essentiel **que vous respectiez à la lettre les spécifications**. En particulier, prêtez attention aux **chaînes de caractères** que le sujet vous demande de produire : comme pour un programme, une erreur sur un seul caractère et c'est tout le programme qui est faux...

Ex. 1. Personnage (/7)

Au sein d'un projet de développement de jeu vidéo, vous êtes chargé de fournir le sous-projet qui concerne les personnages. Les spécifications sont données dans le diagramme de classe est fourni ci-après. On rappelle que le cadre qui englobe le reste du diagramme représente un package, nommé `examPersonnage`.

Conseil : même s'il n'est pas demandé de tester votre programme, cela reste **vivement recommandé !**

1. Implémenter le diagramme de classe.



Ex. 2. Etudiant (/7)

Cet exercice s'intéresse à des étudiants, que l'on note sur plusieurs matières, et dont on pourra calculer la moyenne.

IMPORTANT : dans les questions qui suivent, vous respecterez scrupuleusement les noms donnés aux classes et aux méthodes qui vous sont demandées. Vous penserez à définir vos attributs privés (pas de protected) et à ne pas définir de setters s'ils ne sont pas demandés.

Classe Note

- Ecrire une classe **`Note`** ayant deux attributs privés **`matiere`** (String) et **`valeur`** (double).
- Ecrire le constructeur à 2 arguments **`matiere`** puis **`valeur`**.
- Ecrire les getters (pas de setters) **`getMatiere`** et **`getValeur`**.
- Ajouter la méthode **`toString`** qui transforme une note, par exemple, ainsi : "Anglais : 12.5"

Classe **`Etudiant`**

- Ecrire une classe **`Etudiant`** ayant un attribut privé **`nom`** (et pas de setter).
- Ecrire un constructeur à un argument, le nom, puis le getter de cet attribut **`getNom`**.
- Ajouter un deuxième attribut **`lesNotes`** (la liste des notes), et le getter **`getNotes`**.
- Ajouter une méthode **`noter(String matiere, double valeur)`** qui met à jour **`lesNotes`**.
- Ajouter une méthode **`getMoyenne`**() qui retourne un double : cette méthode parcourt le tableau de notes, en faisant la somme, puis divise cette somme par le nombre total de notes.

Pensez à gérer le cas où l'étudiant n'a aucune note. Remarque : vous ne devez pas définir dans cette classe **`Etudiant`**, d'attribut correspondant à la moyenne.

- Ajouter une méthode **`toString`** à la classe **`Etudiant`** qui retourne un étudiant de la façon suivante :

"Etudiant Sophie - Moyenne : 12.50"

Classe Alternant

- Ecrire une classe **`Alternant`**, qui est donc un étudiant qui travaille aussi en entreprise. A la construction, on se contente de transmettre le nom de l'étudiant.
- Ces alternants possèdent une information supplémentaire : une évaluation de leur tuteur en entreprise, sous forme de chaine de caractères, et dont la valeur sera soit "Excellent", "Satisfaisant", "Passable" ou "Insuffisant". Ajouter une méthode **`setEvaluation`** à la classe **`Alternant`**.
- Définissez la méthode **`getMoyenne`**, de façon à ce que la moyenne d'un alternant soit affectée par l'évaluation du tuteur : si elle est excellente, l'étudiant gagne 2 points sur la moyenne de ses notes ; si elle est satisfaisante, il gagne 1 point ; si elle est passable, il ne gagne rien ; et enfin si elle est insuffisante, l'étudiant perd un point. Bien entendu, la limite supérieure reste 20, et inférieure 0. Vous penserez à traiter le cas où l'alternant n'a pas été évalué par son tuteur.
- Ajoutez enfin une méthode **`toString`** qui produit un résultat comparable pour un alternant :

"Etudiant Sophie - Moyenne : 14,50 - Evaluation : Satisfaisant
Anglais : 12.5
Maths : 14.5"

Remarque : le saut à la ligne s'écrit \n et la tabulation s'écrit \t.

Ex. 3. Véhicule (/7)

Examen R2.01 - 2022

Cet exercice s'intéresse à la gestion d'une flotte de véhicules pour la livraison de marchandises. La société possède trois types de véhicules : des voitures pour les commerciaux et des camions pour les transporteurs de marchandises.

Il vous est demandé de modéliser les classes suivantes :

Classe Véhicule :

1. Ecrire une classe Vehicule avec les attributs suivants : couleur (string) qui représente la couleur du véhicule, typeCarburant (string) qui indique le type de carburant d'un véhicule, capaciteReservoir (int) qui indique la capacité totale du réservoir en litre, carburantDisponible (int) qui indique le carburant disponible à l'instant T.
2. Par soucie d'optimisation budgétaire, la couleur d'usinage est "blanc" par défaut.
3. Définir deux constructeurs pour cette classe et les accesseurs nécessaires.
4. Une méthode toString() qui retourne la valeur suivante : [<nom d'attribut> : <valeur de l'attribut>] séparés par des virgules.
5. Des méthodes pour manipuler la quantité de carburant : void *consommerCarburant()*, void faireLePlein(), void ajouterCarburant(int quantité), boolean reservoirVide() et boolean reservoirPlein().
6. Il vous est demandé de traiter les cas suivants dans ces méthodes :
    - L'utilisateur doit pouvoir faire le plein sans faire déborder son réservoir, afficher un message dans ce cas.
    - Il ne peut pas rouler avec le Véhicule sans carburant, afficher un message dans ce cas.
    - Tous les véhicules ne consomment pas la même quantité de carburant, par conséquent la méthode doit être *spécifique* à chaque Vehicule.

Classe Voiture :

1. La classe Voiture hérite de la classe Vehicule et n'a pas d'attribut supplémentaire par rapport à cette dernière.

Classe Camion :

1. Écrire la classe Camion avec les attributs suivants : capaciteMarchandise (double) qui indique la quantité totale de marchandise que le camion peut transporter en m3.
2. Un attribut marchandiseEnCharge(double) qui indique la quantité totale en m3 de marchandise portée actuellement par le camion.
3. Deux constructeurs (le deuxième constructeur permettra de vider le camion de son contenu.
4. Les accesseurs pour les nouveaux attributs.
5. Deux méthodes pour manipuler la marchandise : void chargerMarchandise(double quantite) et void dechargerMarchandise(double quantite).
6. Un Camion consomme deux fois plus de carburant qu'une Voiture