IUT Lyon 1.
Informatique — Langage Java
Année Spéciale — TP noté et SAE : Héritage, polymorphisme, classes abstraites, interfaces et modélisation d'un problème

---

Partie SAE (évaluation en groupe sur **PC de votre choix**) : à rendre à la fin de la séance du 13 décembre sur **GitLab** (projet privé et inviter M. Vidal en tant que développeur).

Héritage et modélisation d'un problème

On vous demande de développer une application pour aider l'IUT à gérer administrativement ses étudiants et ses salariés. Il faut gérer les absences des étudiants et des salariés afin de savoir si un étudiant ne validera pas son année car il a strictement plus de 5 absences non-justifiées sur un semestre et savoir si une retenue sur salaire doit être effectuée pour un salarié. Si un étudiant ne valide pas son année pour défaut d'assiduité, alors s'il est boursier, il devra rembourser sa bourse au CROUS. Pour simplifier la modélisation on supposera qu'un étudiant ne peut pas être aussi un salarié. Il faut gérer l'attribution des bureaux pour les salariés (on fera l'hypothèse qu'il y a toujours au moins une place disponible, un bureau pouvant contenir jusqu'à 3 places). Enfin, il faut gérer le montant de la paye de chaque salarié : un administratif à un salaire fixe, pas d'heure supplémentaire et une prime annuelle, un enseignant vacataire est uniquement payé à l'heure de cours et un enseignant titulaire est payé sur une partie fixe correspondant à un nombre d'heures statuaire à faire (mini 64h et maxi 384h ; cette information sera remplie par la secrétaire en début d'année scolaire), une partie d'heures supplémentaires et une prime annuelle (pour simplifier on ne fera pas de distinction entre toutes les primes existantes, on supposera qu'il a ou pas une seule prime annuelle d'un montant fixé par l'administration). Si un enseignant titulaire dépasse son volume horaire statuaire, il sera payé en plus en heures supplémentaires (sauf si le nombre d'heures statuaire est strictement inférieur à 192h auquel cas toutes les heures supplémentaires sont définitivement perdues) au même tarif horaire que le vacataire, ce tarif horaire étant fixé à 42€ bruts (pour simplifier).

- Modéliser, grâce à l'héritage, les types concrets (donc instanciables) *Etudiant*, *EtudiantBoursier*, *PersonnelAdministratif*, *PersonnelEnseignantTitulaire* et *PersonnelEnseignantVacataire*. Vous pouvez introduire des classes intermédiaires afin de maximiser la réutilisation du code (et en particulier des classes abstraites ou des interfaces!). Vous devez fournir tous les services (méthodes publiques) que vous jugerez nécessaires (getNom, getNumeroBureau, getSalaire, getMontantBourse etc.).
- Ecrire une classe applicative permettant au secrétariat de l'IUT de gérer ses étudiants et ses salariés. A minima on devrait pouvoir ajouter/supprimer des étudiants ou enseignants, avoir la liste des étudiants avec strictement plus de 5 absences, avoir la liste des

personnels ayant une retenue sur salaire pour absence injustifiée et afficher les informations de chaque étudiant ou enseignant. Les données de votre application sont ajoutées à chaque démarrage. Si vous souhaitez lire/écrire les données depuis un fichier binaire (qui jouerait le rôle d'une base de données peu efficace) ou encore mieux utiliser une véritable base de données cela vous rapportera un bonus (il ne faut envisager cela qu'en dernier).
- Déclarer `final` les classes et les méthodes qui doivent l'être.

Notation :

- Classes: conception/modélisation, constructeur, getters/setters, classes et méthodes `final` (sur 5 pts)
- Interfaces : conception/modélisation (sur 1 pt).
- Qualité de l'implémentation, maximisation de la réutilisation de code et qualité de l'API (stabilité, utilisabilité - *user friendly*, adaptabilité aux évolutions futures, sécurité) (sur 4 pts).
- Application en mode console (idéalement avec un menu interactif et une gestion des erreurs de saisie) et tests des fonctionnalités développées (sur 6 pts).
- Ecriture et génération de la Javadoc (sur 2 pts).
- Git et GitLab : une branche principale stable, une branche par développeur, des commits pour chaque membre avec un message clair (sur 2 pts).