IUT des Pays de l'Adour - Dpt RT - BUT2
R308 - Consolidation de la programmation

Évaluation de Travaux Pratiques

novembre 2024
durée : 2h
responsable : M.Munier

Sujet 3 - Planning

Documents autorisés : Pour cette évaluation de travaux pratiques vous êtes autorisés à utiliser tous les documents pédagogiques qui vous ont été fournis : support de cours "consolidation de la programmation" (module R308) (éventuellement annotés) distribué par M.Munier, corrections des TD/TP, notes de TD et notes de TP. Vous pouvez également utiliser tout autre document (documentations récupérées sur le web, photocopies de livres, etc...) que vous jugerez utiles pour... vous faire perdre du temps. Par contre aucune assistance électronique (calculatrice, ordinateur de poche, smartphone, mail, Google+, Facebook, Twitter, Google Glass,...) n'est autorisée pour communiquer avec qui que ce soit. Toutes les notions vues en cours sont supposées acquises et, le cas échéant, doivent être mises en œuvre.

Restitution de vos travaux : Vous développerez toutes les classes de cette évaluation dans un seul et unique répertoire (éventuellement un seul fichier) dont le nom est **votre nom de famille en minuscules**. À la fin de la séance votre "copie d'examen" sera une archive (formats `zip` ou `tar` uniquement) contenant ce répertoire et dont le nom sera votre nom de famille en mininuscules, sans accents et sans espaces. Vous enverrez ensuite votre archive par mail à l'adresse manuel.munier@gmail.com

Conseils : Durant cette évaluation de TP (qui ne dure que 2h), votre but n'est pas de tout faire, mais plutôt de nous prouver que vous êtes capables de vous débrouiller en programmation objet en Python devant une machine. Cela signifie bien évidemment savoir écrire des classes correctes (constructeurs, méthodes, attributs,...), mais également savoir interpréter les erreurs de syntaxe et/ou d'exécution, les corriger, tester vos classes, etc... Dit de manière un peu plus directe, **il est plus "intéressant" pour vous d'avoir un programme fonctionnel validant une partie des classes que d'avoir écrit toutes les classes mais sans en avoir testé aucune !**

Tout manquement sur la procédure de restitution de votre travail sera immédiatement sanctionné sur votre note !

⋆ ⋆ ⋆

Exercice POO

L'objectif de ce sujet est de programmer en Python des classes nécessaires à la réalisation d'une application de gestion de planning : des créneaux horaires, des plannings,...

1. Écrire la classe `Creneau` permettant de gérer des créneaux horaire. Outre les constructeurs et les méthodes de base pour l'accès aux attributs, il faudra définir la méthode `conflictWith(Creneau)` qui teste si oui ou non le créneau horaire passé en paramètre entre en conflit (i.e. chevauchement) avec le créneau horaire sur lequel est invoquée cette méthode (le résultat est donc un booléen `True` ou `False`). Vous implémenterez également une méthode `duree()` qui retournera un entier représentant la durée en minutes de ce créneau horaire.

2. Écrire maintenant la classe `Planning` représentant un emploi du temps. Pour simplifier cet exercice nous nous limiterons à l'emploi du temps d'une personne sur une journée uniquement. La principale méthode de cette classe sera `ajouterCreneau(Creneau c)` permettant, comme son nom l'indique, d'ajouter un nouveau créneau horaire au planning **à condition qu'il ne soit pas en conflit avec les créneaux déjà présents dans ce planning**. Le résultat de cette méthode sera un booléen indiquant si oui ou non le créneau a été ajouté. La classe `Planning` devra également disposer d'une méthode `dureeTotale()` calculant la durée totale (en minutes) de tous les créneaux horaire insérés dans un planning.

3. Afin de pouvoir valider votre travail, il vous est naturellement demandé d'écrire un programme de test créant plusieurs créneaux, de les ajouter à un planning, et d'invoquer les différentes méthodes.

4. Dériver la classe `Creneau` en trois sous-classes : `CreneauCours`, `CreneauTD` et `CreneauTP`. La particularité de ces trois sous-classes est que le résultat de leur méthode `duree` ne sera plus en **heures réelles** mais en `heures équivalent TD` (ETD) selon le calcul suivant : 1 heure de TD vaut 1 heure ETD, 1 heure de cours vaut 1,5 heure ETD et 1 heure de TP vaut 2/3 heure ETD.

5. Valider votre code par un programme de test mélangeant dans un même objet `Planning` des créneaux horaires de types différents.

Exercice SDD

Sur la base du code fournit dans le cours et testé durant les TP à propos des listes (simplement) chaînées, ajouter à la classe `LinkedList` une méthode `uniq()` qui, dans une liste supposée ordonnée, va supprimer tous les nœuds redondants. Illustrons ceci par un exemple :

- si la liste initiale est `1` $\rightsquigarrow$ `3` $\rightsquigarrow$ `3` $\rightsquigarrow$ `4` $\rightsquigarrow$ `4` $\rightsquigarrow$ `4` $\rightsquigarrow$ `7` $\rightsquigarrow$ `9` $\rightsquigarrow$ `9` $\rightsquigarrow$ `null`
- le résultat sera la liste `1` $\rightsquigarrow$ `3` $\rightsquigarrow$ `4` $\rightsquigarrow$ `7` $\rightsquigarrow$ `9` $\rightsquigarrow$ `null`

* * *

Document rédigé en LaTeX sous Linux.