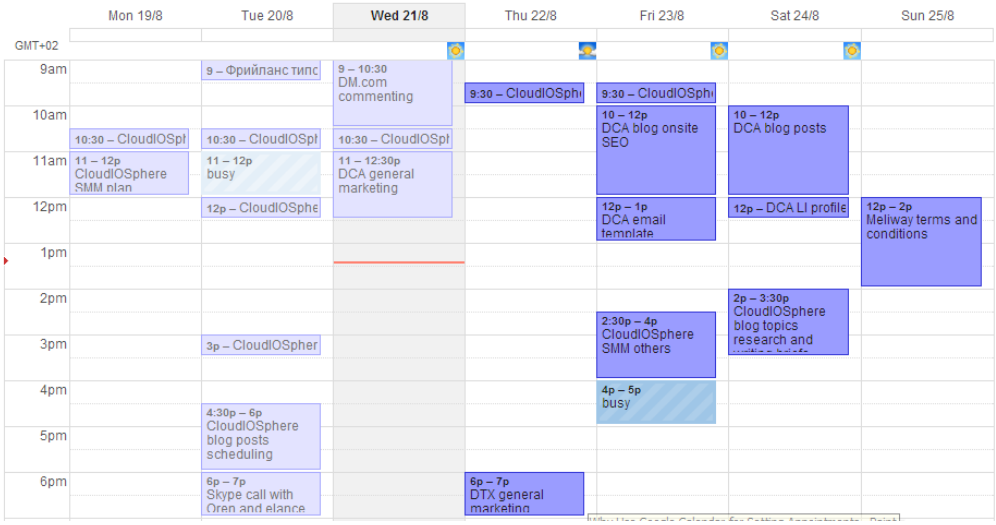
Programmation orientée objet : Examen individuel

Romain Marie & Hervé Gaudin

6 Novembre 2019



Pour éviter toute mauvaise surprise, prenez le temps de bien lire l'ensemble du sujet avant de commencer

Points importants avant de commencer :

- Le nom des classes, méthodes, et propriétés vous est imposé. Pour éviter toute ambiguïté, la syntaxe suivante sera respectée tout au long de l'énoncé : **`Classe`**, *méthode(???)*, **propriété**
- **Sauf indication contraire, les propriétés d'une classe doivent toutes être privées**
- Les méthodes qui vous sont demandées apparaissent avec des ??? à l'intérieur des parenthèses, et sans aucun type de retour. Cela ne signifie pas forcément qu'elles ont besoin de paramètres, ni qu'elles ne renvoient rien.
- Vous testerez **au fur et à mesure** les différentes méthodes que vous définissez, et utiliserez des commentaires pour garder une trace de ce que vous avez testé.
- Si une fonctionnalité vous manque, vous avez le droit de l'ajouter. Non seulement vous ne serez pas pénalisés, mais un point bonus pourrait même être attribué si l'ajout se justifie.

- Pour éviter toute erreur de manipulation, chaque fichier que vous créez devra commencer par un commentaire indiquant vos noms et prénoms
- N'oubliez pas de correctement mettre en forme chacun de vos fichiers (indentation = ctrl+maj+f)
- Vous ferez en sorte que votre programme compile. Si vous êtes coincé sur une question, et que vous ne trouvez pas votre erreur, mettez en commentaire le bloc qui pose problème (ne le supprimez pas !).

1 Mise en place du projet

Dans cet examen, nous allons définir les bases logicielles d'un agenda permettant de gérer les événements d'une journée.

1. Lancer Eclipse Java en choisissant comme workspace le dossier "WorkspaceExamen" qui se situe sur le bureau
2. Créez un projet qui portera vos nom et prénom sous la forme "ExamenPrenomNOM". Par exemple, Jean Dupont devra créer un projet qui s'appelera "ExamenJeanDUPONT".
3. Ajoutez un point d'entrée à votre programme (méthode *main(???)*) dans une classe ***ExamenMain***.
4. Effectuez un affichage à l'écran : "Bienvenue dans le projet agenda."

2 La classe Heure

Pour manipuler un agenda, il est nécessaire de correctement gérer le temps. Nous utiliserons pour cela une classe ***Heure***, composée de deux propriétés : **heure**, qui sera un nombre entier compris dans l'intervalle $[0, 24[$, et **minutes** qui sera également un nombre entier, mais compris cette fois dans l'intervalle $[0, 60[$. Vous ferez bien sûr en sorte que ces deux propriétés ne puissent jamais sortir de ces intervalles.

1. Ajoutez la classe ***Heure*** à votre projet.
2. Ajoutez un constructeur par défaut, qui créé une ***Heure*** arbitrairement réglée à $13h30$.
3. Ajoutez un constructeur avec paramètres, qui permet d'entièrement spécifier l'***Heure*** de son choix.
4. Ajoutez les accesseurs / mutateurs de la classe.
5. Ajoutez une méthode *toString(???)* qui renvoie une chaine de caractère décrivant l'***Heure***. Par exemple, pour $heure = 7$, et $minutes = 23$, la chaine de caractère générée devra être "7h23". Attention, il n'est pas demandé ici de faire un System.out.println(), mais bien de retourner un String.
6. Ajoutez une méthode *precede(???)* qui renvoie un booléen indiquant si l'***Heure*** précède une autre ***Heure*** passée en paramètre.
7. Dans la classe ***ExamenMain***, modifiez la méthode *main()* pour que deux ***Heure*** $h1$ et $h2$ soient créées aléatoirement (les ***Heure*** doivent bien sûr être valides). Affichez les dans l'ordre chronologique.

3 La classe Evenement

Une journée décrite par un agenda est composée d'un ensemble d'événements (rendez-vous, réunion, ...), chacun étant caractérisé par une heure de début, une heure de fin, et un intitulé. Nous allons donc considérer une classe ***Evenement***, dont les propriétés sont :

- **debut** : une ***Heure*** qui correspond à son début,
- **fin** : une autre ***Heure*** qui correspond à sa fin,

- **intitule** : une chaine de caractères (String) qui décrit son contenu.

Attention : l' *Heure* de début devra nécessairement précéder l' *Heure* de fin tout au long du programme. A vous de prendre toutes les précautions qui le garantissent.

1. Ajoutez cette classe à votre projet.
2. Ajoutez un constructeur qui prend en paramètres 4 entiers et un String pour spécifier entièrement un ***Evenement***.
3. Ajoutez un constructeur prenant en paramètres 2 ***Heure*** et un String permettant également de spécifier un ***Evenement***.
4. Ajoutez les accesseurs / mutateurs.
5. Ajoutez une méthode *decaller15Minutes(???)* qui décalle le début de l'***Evenement*** de 15 minutes (si c'est possible). L'***Heure*** de fin de l'***Evenement*** restera inchangée.
6. Ajoutez une méthode *collision(???)* qui renvoie un booléen indiquant si l'***Evenement*** est en collision avec un autre passé en paramètre (c'est à dire si les deux se passent au moins partiellement au même moment).
7. Ajoutez une méthode *toString(???)* qui renvoie un String décrivant l'***Evenement***. Par exemple, si l'intitulé est "Exemple", et que l'***Evenement*** se déroule de $9h00$ à $10h00$, la chaine de caractère renvoyée pourra être : "Exemple : 9h00 - 10h00".
8. Dans la classe ***ExamenMain***, modifiez la méthode *main()* pour que deux ***Evenement*** $e1$ et $e2$ soient créés en utilisant tour à tour les 2 constructeurs à votre disposition. Le premier Evenement aura lieu de $13h00$ à $14h15$ et s'intitulera "Evenement 1". Le second aura lieu de $11h00$ à $12h00$ et s'intitulera "Evenement 2". Vérifiez qu'il n'y a aucune collision entre les deux.

4 La classe Journee

La dernière classe de cette EI sera la classe ***Journee***, qui aura pour unique propriété **evenements**, un tableau de 10 ***Evenement*** (pas d'ArrayList, ni autre collection).

1. Ajoutez cette classe à votre projet.
2. Ajoutez un constructeur par défaut qui construit une ***Journee*** ne contenant aucun ***Evenement***.
3. Ajoutez une méthode *ajouterEvenement(???)* qui ajoute un ***Evenement*** passé en paramètre à la ***Journee***, à condition qu'il ne soit en collision avec aucun ***Evenement*** déjà présent.
4. Ajoutez une méthode *toString(???)* qui affiche dans l'ordre chronologique chaque ***Evenement*** composant la ***Journee***.
5. Dans la classe ***ExamenMain***, modifiez la méthode *main()* pour qu'une ***Journee*** $j1$ soit créée. Ajoutez-y les ***Evenement*** $e1$ et $e2$ créés précédemment. Ajoutez-y également un ***Evenement*** $e3$ de votre choix, et un ***Evenement*** $e4$ de votre choix, qui soit en collision avec $e1$ ou $e2$. Enfin, affichez le contenu de la ***Journee*** $j1$.