

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W13-02-1-protectedmasquage-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

Droits d'accès. Classes du même paquetage. Super-classes. Droit d'accès. Classes de sa descendance. Classe. Programmeurs d'extensions de la classe. Cas de notre exemple. Lien d'héritage. Classes de la descendance. Membre. Priori naturel. Méthode de la classe. Bonne encapsulation. Forte dépendance.



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en Java).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

Héritage : droit d'accès `protected`

(Partie 2)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s



Jusqu'à maintenant, l'accès aux membres (attributs et méthodes) d'une classe pouvait être :

- ▶ soit **public** : visibilité totale à l'intérieur et à l'extérieur de la classe (mot-clé `public`)
- ▶ soit **privé** : visibilité uniquement à l'intérieur de la classe (mot-clé `private`)
- ▶ soit **par défaut** (aucun modificateur) : visibilité depuis toutes les classes du même paquetage (est aussi valable pour le paquetage par défaut que vous utilisez en exercice)

Un troisième type d'accès régit l'accès aux attributs/méthodes au sein d'une hiérarchie de classes :

- ▶ l'accès **protégé** : assure la visibilité des membres d'une classe dans les classes de sa descendance (et dans les autres classes du même paquetage). Le mot clé est «`protected`».

Comme il peut sembler à priori naturel qu'une sous-classe dispose directement de tous ses attributs, ses attributs directement définis dans la classe mais aussi ceux hérités de super-classes, il existe une autre voie pour les droits d'accès qui est le droit d'accès protégé. Lorsqu'un membre est déclaré comme 'protégé' dans une classe donnée alors, il devient directement accessible dans toutes les classes de sa descendance. Le droit d'accès protégé en java se désigne par le mot réservé `protected`. En clair, remplacer `private` par `protected` dans le cas de notre exemple, rend ceci possible. Notez aussi qu'en java, l'accès protégé assure la visibilité, aussi dans toutes les autres classes du même paquetage

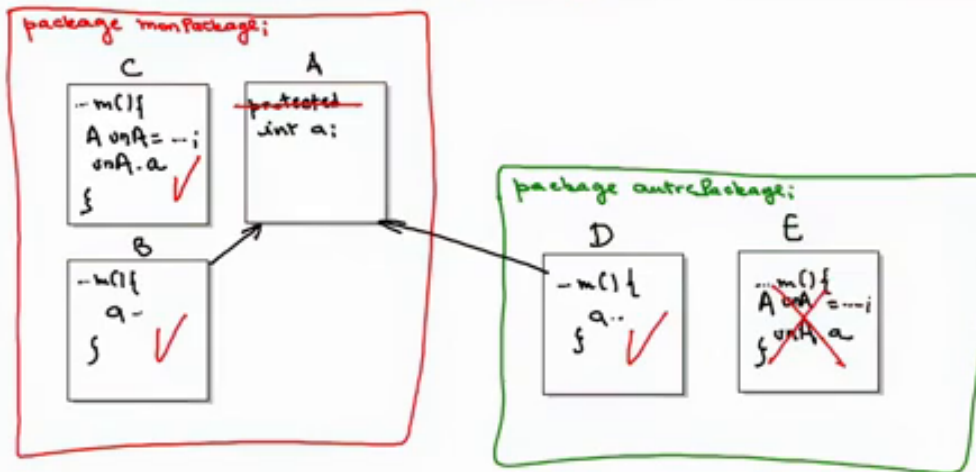
notes

résumé

0m 1s



Accès protégé et paquetages



que celui de la classe contenant le membre protégé. Schématiquement voici comment tout cela peut se résumer, donc supposons que nous ayons un membre protégé dans une classe qui est définie dans un certain paquetage, alors ce membre protégé est directement accessible dans toutes les classes de la descendance de 'A' qu'elle soit si il n'y a pas de lien d'héritage ? Et bien, en java un membre protégé est également accessible dans toutes les classes qui figurent dans le même paquetage que celle de la classe dans lequel le membre est défini et ceux, même si il n'existe aucun lien d'héritage. Par exemple, la classe 'C' n'a pas de lien d'héritage avec la classe 'A' mais se trouve dans le même paquetage, dans ce cas, dans une méthode de la classe 'C' on peut accéder directement à l'attribut protégé de 'A' via tout objet de type 'A' directement sans garde-fou. Cet accès est donc possible, par contre, une autre classe, qui n'aurait pas de lien d'héritage avec celle dans laquelle est défini le membre protégé et qui de surcroît se trouvait dans un autre paquetage, ne pourrait pas accéder aux membres protégés. Ici par exemple, la classe 'E' ne pourrait pas accéder au membre protégé petit 'a' via une instance de grand 'A'. Le droit d'accès protégé est dangereux pour l'encapsulation dans le sens où il permet, notamment, à tous programmeurs d'extensions de la classe 'A', donc tout programmeur qui souhaiterait hériter de la classe 'A', d'accéder aux membres protégés comme s'ils étaient publics ce qui induit une forte dépendance entre ces classes puisque du coup, le programmeur de la classe 'A' n'a plus le loisir de modifier les détails d'implémentation sur ses membres protégés, sans que le programmeur de la classe

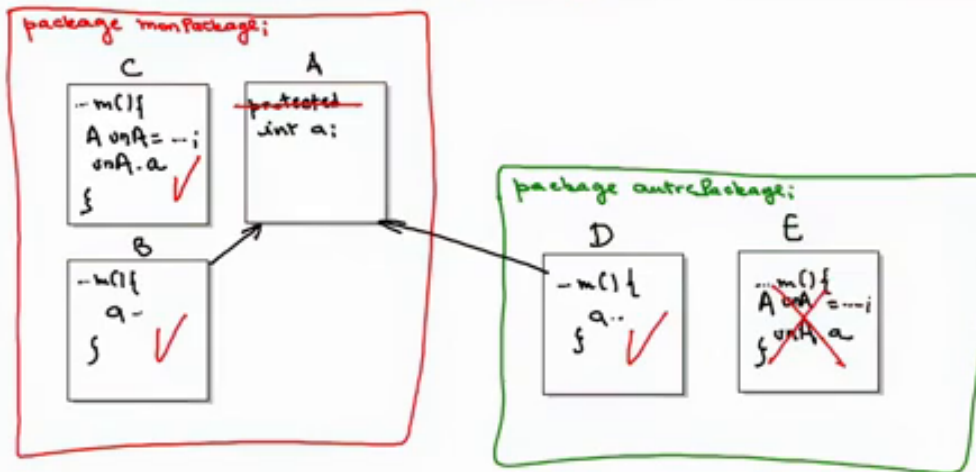
notes

résumé

0m 49s



Accès protégé et paquetages



'D' ne soit impacté. Le droit d'accès par défaut, c'est à dire celui où l'on ne précède la déclaration du membre par aucun mot clé, et pour peu que l'on soit dans un autre paquetage que le paquetage par défaut, est un peu plus restrictif que le droit protégé

notes

résumé

- ▶ Une sous-classe n'a **pas de droit d'accès** aux membres (attributs ou méthodes) **privés** hérités de ses super-classes
 - ☞ elle doit alors utiliser les getter/setters prévus dans la super-classe
- ▶ Si une super-classe veut permettre à ses sous-classes d'accéder à un membre donné, elle doit le déclarer non pas comme privé (**private**), mais comme protégé (**protected**).

Attention : La définition d'attributs protégés nuit à une bonne encapsulation d'autant plus qu'en Java un membre protégé est aussi accessible par toutes les classes d'un même paquetage

- ☞ Les attributs protégés sont d'un usage peu recommandé en Java

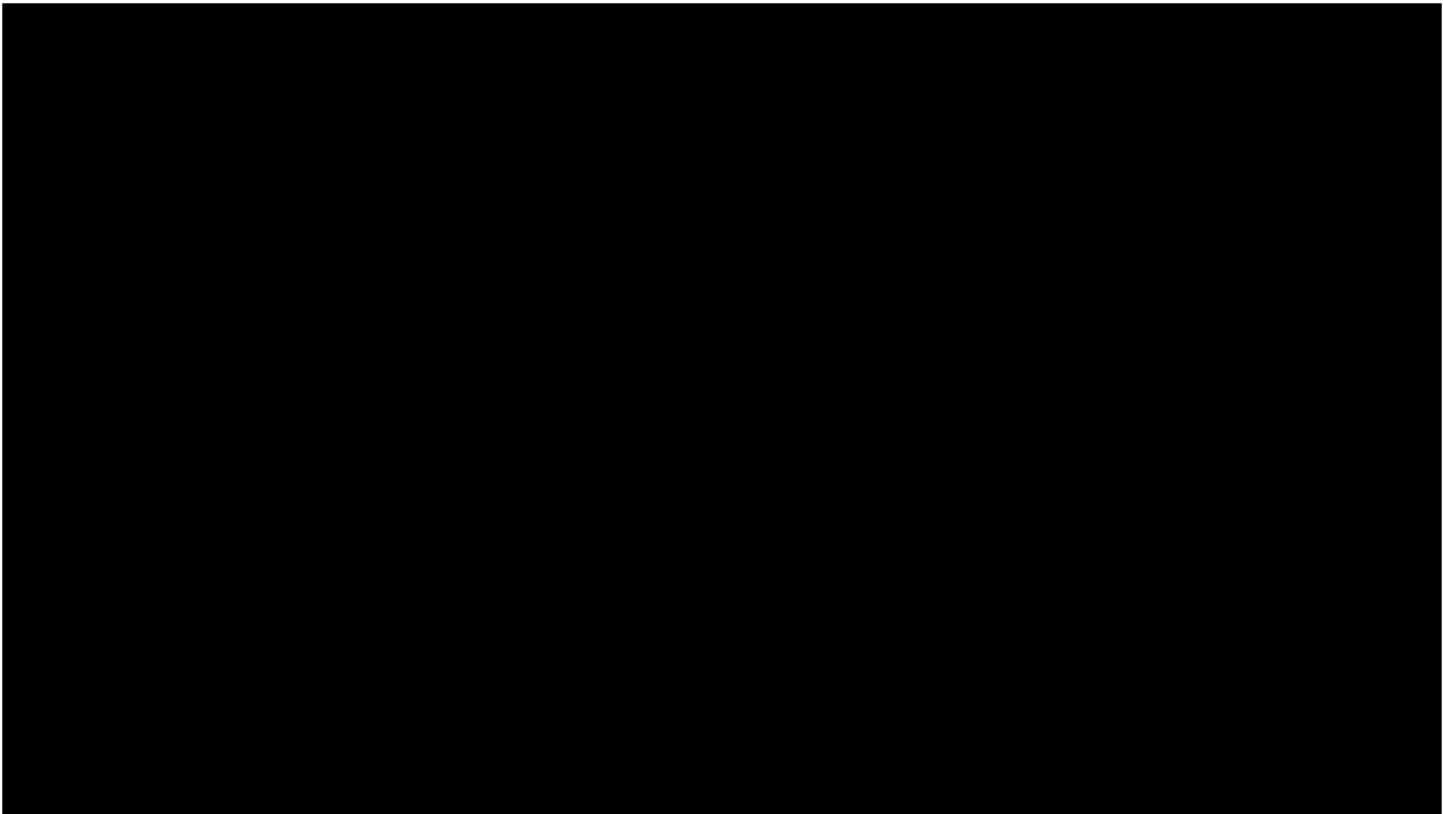
dans le sens où il ne va plus permettre ce genre d'accès. Avec le droit par défaut on ne peut accéder aux membres que dans les classes du même paquetage. Le droit d'accès par défaut, même si un peu plus restrictif que le droit protégé demeure nuisible à une bonne encapsulation puisqu'il permet un accès non contrôlé depuis toutes les classes du même paquetage. Pour résumer toute cette discussion, donc,

notes

résumé

2m 49s





une sous-classe n'a pas accès aux membres privés hérités des super-classes, si elle veut pouvoir utiliser ces membres privés, même si elle bénéficie de ces membres par héritage. Elle doit avoir recours à des getter/setters prévus dans la super-classe. Si une super-classe veut permettre à ses sous-classes d'accéder à un membre donné, alors elle doit le déclarer comme protégé mais attention, ceci nuit à une bonne encapsulation. Donc, les attributs protégés sont d'un usage peu recommandé. sont d'un usage peu recommandé.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

3m 13s



.....

.....

.....

.....

.....