Droit d'accès `protected`

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`».

Accès protégé et paquetages

Accès protégé (1)

- 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

Accès protégé (2)

Le niveau d'accès protégé correspond à une **extension du niveau privé** permettant l'accès aux sous-classes (et aux autres classes du même paquetage).

Exemple :

```
class Personnage {
// ...
   protected int energie;
}

class Guerrier extends Personnage {
  // ...
   public void frapper(Personnage lePauvre) {
      if (energie > 0) {
    // frapper le perso
   }
}
```

Utilisation des droits d'accès

- Membres *publics* : accessibles pour les **programmeurs utilisateurs** de la classe
- Membres *protégés* : accessibles aux **programmeurs d'extensions** par héritage de la classe ou travaillant dans le même paquetage
- Membres *privés* : pour le **programmeur de la classe** : structure interne, (modifiable si nécessaire sans répercussions ni sur les utilisateurs ni sur les autres programmeurs)