Java Cours 9

**Héritage et polymorphisme**

**Diapo 2-4**

Xxx est un yyy 🡪 héritage

Xxx a un yyy 🡪 délégation (agrégation ou composition)

Le surclassement est de déclarer une référence du type de la superclasse et y affecter une valeur qui est une référence d’une de ses sous-classes. Mais attention quand un objet est surclassé (càd associé au type de sa super-classe), il est considéré par le compilateur comme un objet de cette classe et il n’a donc accès qu’aux méthodes de la super-classe et plus à celle dont il est originaire.

**Diapo 5-10**

Principe de lien dynamique. A la compilation c’est la partie « statique » du code qui est vérifiée, en se basant sur les types déclarés. Le choix des méthodes à exécuter se fait au moment de l’exécution. Si la méthode à exécuter est présente dans la classe courante et la super-classe, ce sera celle de la classe courante qui sera exécutée. Par contre si la classe courante ne possède pas cette méthode, on remonte alors dans les super-classes pour exécuter la première méthode de même nom trouvée.

**Diapo 11**

L’upcasting et lien dynamique pour mettre en œuvre le polymorphisme.

Un objet EtudiantSportif peut être manipulé comme un Etudiant !

**Diapo 16**

Surcharge et polymorphisme. Sur la diapo on fait un upcasting avec l’instruction refA = refB, cette instruction est possible car B est une sous classe d’A donc les types sont compatibles. Cependant lors de l’appel de ClasseC.methodX(refA), on a bien le message faisant référence à la ClasseA car l’upcasting fait que refA étant maintenant de type B il reste un sous-type d’A et donc la méthode sélectionnée par le compilateur et celle faisant appel à ClasseA.

**Diapo 17**

Le downcasting permet de faire un « cast » sur des objets (changement de type). Cependant les objets doivent être compatibles, c’est-à-dire que l’objet à transtyper provient de la même classe ou d’une sous-classe de celle que l’on utilise pour le transtypage. Il est impossible transformer un String en un Point car les objets n’ont pas de sous type compatibles.

**Diapo 18**

Les deux méthodes equals sont en surcharge et pas redéfini, donc qd on lui donne un objet ou un point ce n’est pas la même méthode qui est appelée.

Avec o.equals(p1) 🡪 false. O est de type object donc compilateur regarde si methode equals() dans classe object : OK. p1 type point compatible avec type object et donc utilisation de la méthode equals de la classe object (ref des 2 différents donc faux).

**Diapo 19**

En faisant une redéfinition de la méthode equals dans point, on obtient un comportement juste ajoutant dans le code la vérification de la compatibilité entre l’objet et point.

**Diapo 20**

Exemple sur le upcasting et le downcasting 🡪 A bien maitriser !!!