TD2 de Tabbakh Mohamed Amine

Exercice 1 :

1. Le main affiche : 42 42 24 24 24 24 . Voyons en détail le code :
Tout d'abord on fait appel à la méthode meth sur une instance de la class mere donc de manière logique cela renvoie 42 et l'on print cela. Par la suite, l'on fait appel à la méthode printMeth qui elle fait un print du résultat de la méthode meth de manière logique l'on a également 42. Jusqu'à ici, on a pas de problème car la class est de type mere. De même pour fille, l'on créé une instance de type Fille et l'on appelle fille.meth() donc la méthode meth de la class fille et donc comme résultat pour les deux suivant, l'on a 24. Maintenant nous créer merefille, le pointeur est de type Mere mais pointe vers une class de type Fille. Sachant que toute fille est une mère car fille est hérite de la class mère. Ici lorsqu'on fait appel à la méthode meth, le compilateur ne s'intéresse pas au type du pointeur qui pointe mais à l'objet se trouvant à l'espace mémoire correspondante. Ici, l'on sait que l'objet est une fille donc c'est la méthode meth() de la classe fille qui sera appelé et donc l'on a de manière logique 24 qui est returné. L'on peut également cast le pointeur s'il l'on veut en Fille pour que ça soit plus beau avec mereFille = (Fille) mereFille.

2. Dans Fille, il a accès à son propre meth et au meth de la mère qui est en protected donc les classes filles y ont accès. S'il est dans le main, à aucun par principe d'encapsulation de la POO.

3. Si les meths sont static, on a le comportement suivant : 42 42 24 42 42 42. Nous allons étudier le code en détail pour comprendre cela. Pour les deux premiers, c'est juste normale car l'on appelle la méthode de la class mere. Pour la deuxième, lorsqu'on appel fille.meth() l'on appel la méthode de la class fille donc l'on renvoie 24 de manière. Maintenant pour la deuxième, l'on appel appelMeth() qui est une méthode de la class mère qui elle appelle Meth static de la classe Mère et d'ou le 42 printé. Pour la troisième, l'on voit le type du pointeur qui lui est type Mere donc on appel la méthode static de la class Mere et ce qui permet d'avoir comme retour 42. De même pour printMeth() car l'on appel une méthode de la classe Mere qui elle va utiliser la méthode static s'y trouvant et donc renvoyé 42.

4. Cette fois ci , l'on aura 42 42 24 42 42 42

Exercice 2 :

1. On a un problème avec la méthode char h() de fille, en effet l'on a fait une redéfinition mais avec une signature similaire d'un point de vue arguments, cela crée donc une erreur : solutions : soit changer la signature et donc rajouter par exemple un argument ex : h(int a), soit modifier le char en int et faire donc une surcharge. Même problème pour la méthode int i() de fille : soit on change en void et on aura une surcharge soit rajouter un argument par exemple int i(int a) et donc une redéfinition. Même problème pour void k() et là il faut remplacer throw IOException pour avoir une surcharge ou alors throw Exception dans la mère sachant que IOException est une class héritante de Exception.

3. Surcharge : lorsqu'on réécrit la même méthode que la mère sans changer la signature. Redéfinition : lorsque la signature est changé. À la compilation si l'on est dans le cas d'une redéfinition , l'on choisit selon l'argument et dans le cas d'une surcharge : l'on regarde le vrai type de l'objet. Ici , surcharge : a,d,e,f et redéfinition : b, c, g,j,l,m .

4. On a le résultat suivant :