1. **Qu'affiche le main et pourquoi ?**

Nom	Ce qu'affiche le Main
System.out.println(mere.meth());	**42**
mere.printMeth();	**42**
System.out.println(fille.meth());	**24**
fille.printMeth();	**24**
System.out.println(mereFille.meth());	**24**
mereFille.printMeth();	**24**

2. S'il est dans Fille, à combien de méthodes meth() un objet de type Fille à

accès (et comment il y accède) ? Et s'il est dans Main ?

S'il est dans Fille, un objet Fille a accès uniquement à la methode meth() de la classe Fille.

Il y accède de cette manière :

```
System.out.println(fille.meth());
```

S'il est dans Main, un objet Fille a accès à la méthode meth() de la classe Fille et de la classe mère.

Il y accède de cette manière :

```
Fille fille = new Fille ();

        System.out.println(fille.meth());

        fille.printMeth();
```

2. **Quel est le comportement si les méthodes meth() sont statiques ?**

   On ne peut plus accéder à la méthode meth() avec l'objet fille à partir du Main. De même pour l'objet mère.

Nom	En static
System.out.println(mere.meth());	**42**
mere.printMeth();	**42**
System.out.println(fille.meth());	**24**
fille.printMeth();	**42**
System.out.println(mereFille.meth());	**42**
mereFille.printMeth();	**42**

Commenté [MA1]: Une méthode statique ne peut être redéfinie dans une sous-classe
Cela fait qu on a maintenant uniquement accès à la méthode « meth() » de la classe Mere

Commenté [MA2]: A la compilation, on récupère les attributs et méthodes de la classe de l'objet « merefille »

Ce Mère merefille = new Fille() ;
Signifie qu on crée un obj mèrefille de type Mère mais qui récupère aussi TOUS LES ATTRIBUTS ET METHODES DE LA SOUS CLASSE FILLE

4. Et si meth sont maintenant des champs ? Pourquoi ?

Nom	Champ
System.out.println(mere.meth());	**42**
mere.printMeth();	**42**
System.out.println(fille.meth());	**24**
fille.printMeth();	**42**
System.out.println(mereFille.meth());	**42**
mereFille.printMeth();	**42**

Commenté [MA3]: printMeth est dans la classe Mère et fait appel au champ « meth » qui est dans la classe Mère. Ici on n appellera pas la méthode « meth » de la classe Fille car meth est considéré comme un champ

On récupère ce tableau car les champs crées sont directement liés à leurs classes.

1. **Quelles sont les erreurs de compilation et pourquoi ?**
   - **Il manque des } et ;**
   - The method miage() is undefined for the type MereExo2→ Cette methode n est pas définie dans la méthode Mere.

   `mereFille.miage(); →` **Même problème que précédemment**

2. Retirer les méthodes provoquant les erreurs.

3. Rappeler ce qu'est une redéfinition et une surcharge, et indiquer où sont les surcharges et où sont les redéfinitions ici.

Une redéfinition est quand on a par exemple deux méthodes qui ont le même nom et le même profil dans deux classes et dont l'une hérite de l'autre. (i.e. même nom + nombre ET type d'arguments égaux+ même valeur de retour)

Une surcharge est quand on a des méthodes avec le même nom mais des profils différents dans une même classe. (i.e. même nom + nombre et/ou type d'arguments différents)

Réponses : Cf Code Eclipse.

4. Expliquer chaque affichage.

Code	Affichage	Explication
`fille.miage();`	Miage	Appelle la méthode miage() issue de la classe fille « fille » dont le comportement est d'afficher « Miage » car l'objet fille crée est de type « Fille »
`((FilleExo2)mereFille).miage();`	Miage	On considère l'objet « mereFille » de type « fille » à la compilation. Donc cela fait qu'on appelle la méthode miage de la classe « Fille » C'est pourquoi on obtient le même résultat que précédemment
Mere_a() ;	**Mère_a**	On appelle la méthode a() issue de la classe mère « Mere » dont le comportement est d'afficher « Mere_a » car l'objet mere crée est de type « Mere »
`mereFille.a();`	**Fille_a**	**mereFille est déclarée comme une Mère mais constaté comme une Fille.** **En effet, à la compilation mèreFille est considéré de type « Mère » mais à l'exécution mereFille est considéré de type « Fille »**
`fille.a();`	**Fille_a**	Appelle la méthode a() issue de la classe fille « fille » dont le comportement est d'afficher « Fille_a » car l'objet fille crée est de type « Fille »
`((MereExo2)mereFille).a();`	**Fille_a**	mereFille est déclarée comme une Mère mais constaté comme une Fille. En effet, à la compilation mèreFille est considéré de type « Mère » mais à

		l'exécution mereFille est considérée de type « Fille » Peu importe le « cast », mereFille sera considérée de type « Fille » à l'exécution
`mereFille.b(null);`	**Fille_b(Fille)**	mereFille est déclarée comme une Mère mais constaté comme une Fille. En effet, à la compilation mèreFille est considéré de type « Mère » mais à l'exécution mereFille est considérée de type « Fille » Peu importe le « cast », mereFille sera considérée de type « Fille » à l'exécution