Java Cours 10

**Héritage et classes abstraites**

**Diapo 23**

Sur cette diapo on reste dans une approche classique d’héritage.

**Diapo 24**

Ici il y a une poste incrémentation dans le tableau donc on met en position 0 du tableau la Forme f puis on incrémente pour mettre la prochaine forme en 1.

Dans la classe Forme il n’y a pas de fonction périmètre donc erreur à la compilation.

Pour pallier le problème il faut faire une classe abstraite.

**Diapo 25**

On transforme la classe Forme en classe abstraite. On lui donne alors les méthodes perimetre() et surface() avec le mot clé « abstract » et clôturées par « ; ». C’est chaque objet (rond, carré, triangle) qui sera en charge de gérer ces méthodes. Elles sont redéfinies dans chaque sous-classe implémentant la classe abstraite.

**Diapo 26**

On ne peut pas instancier directement la classe abstraite, mais elle peut avoir un ou plusieurs constructeurs. On crée des objets à partir des sous-classes l’implémentant. On peut créer des objets de la classe abstraite en exploitant le polymorphisme. C’est-à-dire que pour la création de l’objet on fera appel à une sous-classe de la classe abtraite.

Une classe peut être abstraite sans méthode abstraite. Cela permet notamment d’empêcher des instanciations de cette classe.

**Diapo 27**

Une classe abstraite est destinée à être hérité. Les sous-classes concrètes doivent obligatoirement redéfinir toutes les méthodes définies en abstract dans la classe mère abstraite.

**Diapo 28**

En UML les classes ou méthodes abstraites sont écrites en italique.

FormeCirculaire est une classe abstraite car elle ne redéfinie pas la méthode abstraite périmètre() de sa super-classe.

**Diapo 29**

Maintenant, grâce à l’implémentation de la classe abstraite le compilateur est ok.