T.P. n°7 en
Programmation Orientée Objet en JAVA

Exercice 1

1) Créer un package modificateur

Ecrire une classe Tapis, dans le package modificateur, de mode d'accès public avec trois attributs :

- longueur, float , de mode d'accès par défaut
- largeur, float, de mode d'accès public
- prixMetreCarre, float de mode d'accès protected

- Ecrire un constructeur public sans paramètres de la classe Tapis
- Ecrire un constructeur public permettant d'initialiser les attributs de Tapis
- Ecrire une méthode calculerSurfaceTapis() d'accès par défaut permettant de calculer la surface d'un tapis (surfaceTapis=longueur*largeur)
- Ecrire une méthode protected calculerPrixTapis(float surfaceTapis) permettant de calculer et de retourner le prix d'un tapis (prixTapis=prixMetreCarre*surfaceTapis)

2) Créer une classe Usage1Tapis dans le package modificateur qui as une méthode main permettant la saisie de trois valeurs float en utilisant la classe Scanner. Utiliser ces valeurs dans la création d'un objet t1 de type Tapis. Calculer et afficher le prix de t1 suite à l'appel de la méthode calculerSurfaceTapis puis la méthode calculerPrixTapis.

3) - Créer un package usageModif.

- Ecrire une classe Usage2Tapis dans le package usageModif qui as une méthode main permettant la saisie de trois valeurs float en utilisant les arguments de main. Utiliser ces valeurs dans la création d'un objet t2 de type Tapis. Calculer et afficher le prix de t2 suite à l'appel de la méthode calculerSurfaceTapis puis la méthode calculerPrixTapis.

- Est ce que la classe compile ?

Non, parce qu'on ne peut pas accédé a des method non visible

- Donner les causes des erreurs de compilation (les mettre en commentaire devant les instructions contenant l'erreur)

……
……
……
……

4) - Ecrire une classe HeritTapis dans le package usageModif. Ecrire une méthode main qui permet de créer un objet t3 de type Tapis, en donnant des valeurs de votre choix aux attributs longueur, largeur et prixMetreCarre. Calculer et afficher le prix de t3 suite à l'appel de la méthode calculerSurfaceTapis puis la méthode calculerPrixTapis.

- Est ce que la classe compile ?

Non, parce qu'on ne peut pas accédé a des method non visible .

- Donner les causes des erreurs de compilation

La méthode calculerSurfaceTapis est déclarée avec un accès package-private dans la classe Tapis, est uniquement accessible à d'autres classes du même package.
La méthode calculerPrixTapis est déclarée avec un accès protected dans la classe Tapis, Bien que cela autorise son accès aux sous-classes de Tapis, la classe Usage2Tapis n'est pas une sous-classe de Tapis.

5) Dans la classe HeritTapis, créer un objet ht de type HeritTapis comme suit :

HeritTapis ht= new HeritTapis() ; **//la classe HeritTapis n'as pas de constructeur, on utilise le //constructeur par défaut**

6) Quels sont les attributs de Tapis accessibles par HeritTapis ?

Les attributes de Tapis accessibles par HeritTapis sont largeur et prixMetreCarre .

Compléter le programme pour modifier les valeurs des attributs accessibles par **ht** en leurs donnant des valeurs de votre choix.

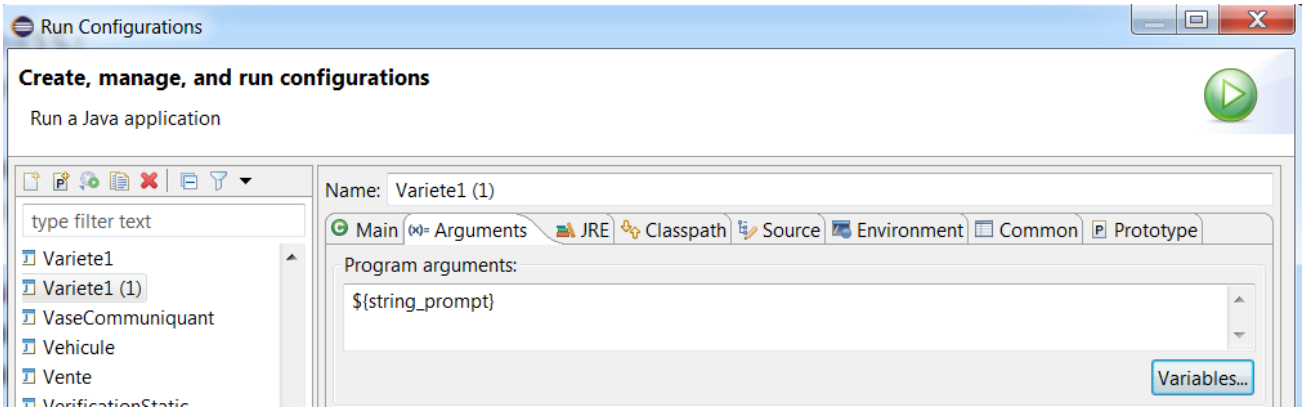
7) Peut-on avoir une solution pour modifier la valeur des attributs non accessibles sans modifier leurs modalités d'accès ? si oui, modifier le programme Tapis avec les instruction nécessaires puis modifier les attributs, non encore modifiés de ht par des valeurs de votre choix.

8) Compléter le programme pour calculer et afficher le prix de **ht** suite à l'appel de la méthode calculerSurfaceTapis puis la méthode calculerPrixTapis.

- **Est-ce qu'il ya des erreurs de compilation ? expliquez**

Comment utiliser les arguments de main avec Eclipse ?

Run => run configurations => cliquer sur l'onglet Arguments=> (variables/String_prompt) à répéter autant de fois que le nombre de valeurs voulues



Les classes Enveloppes (wrappers)

En java pour chaque type primitive, il existe une classe comme le montre le tableau suivant :

Primitive	**Wrapper Class**
Boolean	Boolean
Byte	Byte
Char	Character
Double	Double
Float	Float
Int	Integer
Long	Long
Short	Short

Ceci est pour permettre d'envelopper les primitives en objet en cas de besoin.
Pour convertir un String en int on utilise la méthode statique parseInt de la classe Integer. La forme générale de la méthode est : **primitive parseXxx(String)**

On peut convertir un String vers int, Short, Double, Float ou Long en utilisant l'une de ces méthodes :
parseInt, parseShort, parseByte, parseDouble, parseFloat, parseLong

Pour invoquer la méthode parseXxx : **ClasseEnveloppe.parseXxx(String)**

Les méthodes parseXxx sont déclarées static, c'est pourquoi on les appelle en utilisant les noms de leurs classes par exemple : int h= Short.parseShort("12") ; float p= Float.parseFloat("12.3") ;

La méthode parse lève une exception au moment de l'exécution si String n'est pas convertible vers le type souhaité.