CENTRE SCOLAIRE SAINTE-JULIENNE

TA 9 – POO Héritage et Interface

Exercices Java – Série 5 – Enoncés

I- Mise en situation

Tu es analyste-programmeur dans une société et tu dois passer un test en langage Java. A travers une série d'exercices, tu dois comprendre et maitriser le langage Java pour obtenir la prime salariale.

II- Objets d'apprentissage

Appliquer	Transférer
• Modéliser une logique de programmation orientée objet • Déclarer une classe • Instancier une classe (objet) • Utiliser les méthodes de l'objet instancié • Traduire un algorithme dans un langage de programmation • Commenter des lignes de codes. • Tester le programme conçu	• Développer une classe sur la base d'un cahier des charges en respectant le paradigme de la programmation orientée objet (POO) • Programmer en recourant aux classes nécessaires au développement d'une application orientée objet • Corriger un programme défaillant • Améliorer un programme pour répondre à un besoin défini

Connaître

- Différencier la programmation impérative de la programmation orientée objet
- Caractériser une classe
- Décrire la création d'un objet (instanciation)
- Identifier l'instance d'une classe
- Caractériser les attributs dans une classe (encapsulation)
- Caractériser les méthodes dans une classe (encapsulation)
- Décrire la création d'un constructeur
- Différencier les types de visibilités

III- Travail à accomplir

1. Analyser l'énoncé du point IV correspondant au numéro de l'exercice demandé.
2. Modéliser en diagramme de classes l'exercice.
3. Réaliser l'exercice.
4. Commenter le travail.
5. Visualiser le travail.
6. Sauvegarder le document suivant les instructions données.
7. Imprimer le(s) document(s)

IV- Enoncés

1. Ex01

   Créer la classe **Point** comprenant 2 variables d'instance entières (x et y), 1 *constructeur* à 2 paramètres, les méthodes *accesseurs* et la méthode *toString*.
   Un **Carre** est un Point avec une variable d'instance supplémentaire de type double(*cote*). On y trouve 1 *constructeur* à 3 paramètres, les méthodes *accesseurs* et la méthode *toString*.
   Un **Cercle** est un Point avec une variable d'instance supplémentaire de type double(*rayon*). On y trouve 1 *constructeur* à 3 paramètres, les méthodes *accesseurs* et la méthode *toString*.
   Un **Rectangle** est un Point avec deux variables d'instance supplémentaires de type double (*longueur*, *largeur*). On y retrouve 1 *constructeur* à 4 paramètres, les méthodes *accesseurs* et la méthode *toString*.
   Un **TriangleRectangle** est un Point avec deux variables d'instance supplémentaires de type double (*base*, *hauteur*). On y retrouve 1 *constructeur* à 4 paramètres, les méthodes *accesseurs* et la méthode *toString*. Prévoir dans ces classes, une méthode *calculerPerimetre* retournant le périmètre de la figure et une méthode *calculerAire* retournant l'aire de la figure.
   La classe **Test** crée une instance et affiche le périmètre et l'aire de chacune des figures. A cet effet, 'utilisation du polymorphisme est impérative.

2. Ex02

   Classe: **CompteEnBanque** VI: String *titulaire*, int *solde*, String *numero*. VC: double *tauxInteret*.
   Classe: **CompteCourant** VI: String titulaire, int solde, String numero.
   Classe: **LivretEpargne** VI: String titulaire, int solde, String numero. VC: int *soldeMinimum*.
   Prévoir pour chacune de ces classes les *méthodes habituelles*, une méthode *retrait* (qui diminue la valeur du solde), une méthode *depot*(qui augmente la valeur du solde) et une méthode *calculerInteret*. Le corps des méthodes retrait et depot est écrit dans la classe CompteEnBanque. Le solde minimum renseigne la valeur en-dessous de laquelle on ne peut pas descendre. Elle adapte éventuellement le solde et affiche un message (retrait accepté ou refusé). La méthode calculerInteret adapte le solde. Le taux d'intérêt pour un livret est 3 fois le taux de base.
   Classe: **Test** est une application qui crée 5 comptes courants et 5 livrets d'épargne. Elle calcule ensuite pour chaque compte en banque et en utilisant le polymorphisme, le nouveau solde via la méthode calculerInteret. Elle teste ensuite les 2 cas de retrait pour un livret d'épargne.

3. EX03

Classe: **Adresse** VI: String *rue*, String *localite*. Prévoir les *méthodes habituelles*.

Classe: **Personnel** VI: String *nom*, double *salaire*, Adresse *adresse*. Prévoir les *méthodes habituelles* et une méthode *augmenterSalaire* qui augmente le contenu de la variable d'instance salaire du nombre de % passé en paramètre.

Classe: **Patron** (hérite de Personnel) VI: String *nom*, double *salaire*, Adresse *adresse*, String *nomSecretaire*, int *anneesAnciennete*. Prévoir les *méthodes habituelles* et une méthode *augmenterSalaire* qui augmente le contenu de la variable d'instance salaire du nombre de % passé en paramètre augmenté de 0.5% par année d'ancienneté (on fera appel à augmenterSalaire de Personnel après avoir adapté le pourcentage).

Classe: **Test** est une application qui crée des instances de Personnel et de Patron. Elle calcule ensuite pour chaque membre du personnel de l'entreprise et en utilisant le polymorphisme, le nouveau salaire via la méthode augmenterSalaire.