CENTRE SCOLAIRE SAINTE-JULIENNE

TA 6 – POO Objet

Exercices Java – Série 2 – 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 une classe **Rectangle**. La tester avec une application **TestRectangle**. La classe Rectangle contient 4 variables d'instance de type short: coordonnée en x du coin gauche, coordonnée en y du coin supérieur gauche, **longueur** et **largeur**. Prévoir **1 constructeur** à 4 paramètres. Écrire les méthodes **accesseurs** (vérifier que la longueur est >= à la largeur et si ce n'est pas le cas, donner à la longueur la dimension de la largeur) ainsi que la méthode **toString**. Créer la méthode **calculerPerimetre** et la méthode **calculerAire** qui retournent un entier. Prévoir la méthode **isCarre** qui retourne un booléen.

2. Ex02

   Créer une classe **Rectangle**. La tester avec une application **TestRectangle**. La classe Rectangle contient 2 variables d'instance de type byte: **longueur** et **largeur**. Prévoir **2 constructeurs:** sans paramètre (alors le rectangle fait 50*50) et avec 2 paramètres. Écrire les méthodes **accesseurs** ainsi que la méthode **toString**. Créer la méthode **calculerPerimetre** et la méthode **calculerAire** qui retournent un entier.

3. Ex03

   Créer une classe **Fraction**. La tester avec une application **TestFraction**.
   La classe Fraction contient 2 variables d'instance de type int: **numerateur** et **denominateur**. Prévoir **3 constructeurs**: sans paramètre (alors la fraction vaut 1/2), 1 paramètre, le numérateur (alors la fraction vaut numerateur/10) et avec deux paramètres. Écrire les méthodes **accesseurs** ainsi que la méthode **toString**. Créer la méthode **additionner** et la méthode **multiplier** qui modifient la fraction courante.
   Écrire *éventuellement* une méthode private **reduire**.

4. Ex04

   Créer une classe **Complexe**. La tester avec une application **TestComplexe**.
   La classe Complexe contient 2 variables d'instance de type double: **partieReelle** et **partieImaginaire**. Prévoir **3 constructeurs**: sans paramètre (alors le complexe vaut (0,0)), avec 1 paramètre – la partie réelle – (alors le complexe vaut (partieReelle, 0)) et avec deux paramètres. Écrire les méthodes **accesseurs** ainsi que la méthode **toString**. Créer la méthode **additionner** et la méthode **soustraire** qui créent chacune un nouveau complexe.