Formateur : LAMOURI Najib

Compléments sur la P.O.O :

Exercice 1:

Définir une interface IVehicule caractérisée par les propriétés Carburant (Essence ou diesel), Puissance (nombre de chevaux), NombrePlace, Distance (Nombre de Km parcourus depuis la fabrication), Reserve (nombre de litre dans le réservoir de carburant) et les méthodes void Deplacer(int distance) ainsi que void Approvisionner(int nbLitres).

Définir la classe Voiture qui implémente l'interface IVehicule, ajouter les champs correspondants aux propriétés de l'interface en plus du champ reserveMax qui indique le nombre de litres maximal qui peut être stocké dans le réservoir, définir les constructeurs les propriétés et les méthodes Deplacer et Approvisionner sachant que :

La méthode Deplacer permet tout d'abord de calculer le nombre de litres nécessaire à parcourir la distance passée en paramètre sachant qu'une voiture à essence consomme 7% (c à d : 7 litres par 100Km) et une voiture diesel consomme 6% de carburant. Si la réserve du carburant n'est pas suffisante, un message d'erreur indique à l'utilisateur ceci, sinon le compteur de la voiture sera augmenté de la distance parcourue et la réserve de carburant sera diminuée.

La méthode Approvisionner permet d'augmenter la réserve du carburant sans dépasser la valeur maximale (reserveMax).

Redéfinir la méthode ToString qui retourne une chaine représentant l'objet voiture en cours.

Créer une classe Program permettant de tester la classe précédente.

Exercice 2: Définir une classe Complexe caractérisée par la partie réelle et la partie imaginaire, et un champ qui informe sur le nombre des complexes créés. Définir un constructeur par défaut, et un constructeur d'initialisation avec un nombre de paramètres variables (1 seul paramètre signifie que la partie imaginaire est nulle, deux paramètre signifie que le première correspond à la partie réelle et le deuxième à la partie imaginaire). Définir un constructeur de recopie.

Définir les propriétés pour accéder aux différents champs de l'objet.

Définir une méthode Module qui retourne le module du complexe en cours sachant que :

Si $Z=x+iy$, le module de Z est racine(x^2+y^2).

Définir un indexeur qui retourne la partie réelle pour l'index 0, la partie imaginaire pour l'index 1et le module du complexe pour l'index 2.

Définir l'opérateur + entre deux complexes qui retourne la somme des deux complexes, l'opérateur + entre un complexe et un réel qui retourne la somme du complexe et du réel sachant que le résultat est

$Z=x+iy$, R est un réel, $Z+R=(x+R)+iy$.

Définir l'opérateur * entre un réel et un complexe qui retourne le produit des deux.

Définir l'opérateur – entre deux complexes, et entre un complexe et un réel à partir des opérateurs précédents.

Définir l'opérateur == entre deux complexes qui retourne un booléen indiquant si les deux complexes sont égaux.

Définir l'opérateur != entre deux complexes qui retourne un booléen indiquant si les deux complexes ne sont pas égaux.

Définir l'opérateur * entre deux complexes qui retourne le produit des deux complexes sachant que si :

$Z1=x1+iy1$ et $Z2=x2+iy2$ alors $Z1*Z2=(x1x2-y1y2) + i(x1y2+x2y1)$

Redéfinir la méthode ToString qui retourne une chaine représentative d'un complexe (Exemple : 5+7i)

Créer une classe Program pour tester la classe Complexe.

Exercice 3: Définir une classe Vecteur caractérisée par un tableau de doubles qui sont les coordonnées du vecteur, et un champ qui informe sur le nombre des vecteurs créés. Définir un constructeur d'initialisation avec un nombre de paramètres variables et un constructeur de recopie.

Définir les propriétés pour accéder aux différents champs de l'objet.

Définir une méthode Norme qui retourne la norme du vecteur en cours sachant que la norme est la racine carrée de la somme des carrés des coordonnées : racine($x_1^2+ x_2^2+ x_3^2+\ldots$)

Définir un indexeur qui retourne la coordonnée correspondante à chaque index.

Définir l'opérateur + entre deux vecteurs qui retourne la somme des deux vecteurs.

Définir l'opérateur * entre un réel et un vecteur qui retourne le produit des deux.

Définir l'opérateur – entre deux vecteurs à partir des opérateurs précédents.

Définir l'opérateur == entre deux vecteurs qui retourne un booléen indiquant si les deux vecteurs sont égaux.

Définir l'opérateur != entre deux vecteurs qui retourne un booléen indiquant si les deux vecteurs ne sont pas égaux.

Définir l'opérateur * entre deux vecteurs qui retourne le produit scalaire entre les deux vecteurs sachant que les deux vecteurs doivent avoir la même dimension et que si :

V1=(x1, x2, x3,…) et V2=(y1, y2, y3,…) alors V1*V2=x1y1+x2y2+x3y3+…

Redéfinir la méthode ToString qui retourne une chaine représentative d'un vecteur (Exemple : (1, 2.5, 13, 19, 11.67))

Créer une classe Program pour tester la classe Vecteur.