

TP : la programmation orientée objet avec TypeScript

Exercice 1

Considérons une classe appelée `Point` ayant les attributs suivants :

- `abs` : un attribut privé de type `number`
- `ord` : un attribut privé de type `number`

1. Définissez la classe `Point`, un constructeur avec deux paramètres `Point(private _abs: number, private _ord: number)`
2. Définissez les getters et setters pour les deux attributs.

3. Écrivez la méthode `calculerDistance(p: Point)` qui permet de calculer la distance entre le point de l'objet courant (`this`) et l'objet `Point` `p` passé en paramètre. Nous rappelons que la distance entre deux points $A(x_1, y_1)$ et $B(x_2, y_2)$, en mathématiques, est égale à :

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Utiliser la fonction `Math.sqrt(a)` pour calculer la racine carrée de `a` et `Math.pow(x, y)` pour calculer x^y

4. Écrivez la `calculerMilieu(p: Point)` qui permet de calculer et de retourner un objet correspondant au milieu du segment défini par le point de l'objet courant (`this`) et l'objet `Point` `p` passé en paramètre. Nous rappelons que les coordonnées d'un point $M(x_M, y_M)$ milieu de $A(x_1, y_1)$ et $B(x_2, y_2)$, en mathématiques, sont :

- $x_M = \frac{x_1 + x_2}{2}$
- $y_M = \frac{y_1 + y_2}{2}$

La méthode doit retourner un objet `Point` et pas les coordonnées.

Considérons maintenant une deuxième classe appelée `TroisPoints` ayant les attributs suivants :

- `premier` : un attribut privé de type `Point`
- `deuxième` : un attribut privé de type `Point`
- `troisième` : un attribut privé de type `Point`

5. Définissez les getters/setters et le constructeur avec trois paramètres de la classe `TroisPoints`.
6. Écrivez une méthode `TesterAlignement()` qui retourne `true` si les trois points `Premier`, `Deuxième` et `Troisième` sont alignés, `false` sinon. Nous rappelons que trois points `A`, `B` et `C` sont alignés si $AB = AC + BC$, $AC = AB + BC$ ou $BC = AC + AB$ (AB désigne la distance séparant le point `A` du point `B`, pareillement pour `AC` et `BC`).
7. Écrivez une méthode `estIsocèle()` qui retourne `true` si les trois points `premier`, `deuxième` et `troisième` forment un triangle isocèle, `false` sinon. Nous rappelons qu'un triangle `ABC` est équilatéral si $AB = AC$ ou $AB = BC$ ou $BC = AC$.
8. Dans un fichier `main.ts`, testez toutes les classes et méthodes que vous avez implémentées.

Exercice 2

Considérons une classe **Stagiaire** ayant les attributs suivants :

- **nom** : un attribut privé de type chaîne de caractères
 - **notes** : un attribut privé de type tableau de nombres
1. Créez la classe **Stagiaire** et définissez un constructeur avec deux paramètres
 2. définissez les getters et setters des deux attributs.
 3. Écrivez la méthode **calculerMoyenne()** qui permet de retourner la moyenne de notes d'un stagiaire
 4. Écrivez les méthodes **trouverMax()** et **trouverMin()** qui permettent de retourner respectivement les notes max et min d'un stagiaire.

Considérons maintenant une classe appelée **Formation** ayant les attributs suivants :

- **intitulé** : un attribut privé de type **string**
 - **nbrJours** : un attribut privé de type **number**
 - **stagiaires** : un tableau d'objets de type **Stagiaire**
5. Créez la classe **Formation**, Définissez les getters et setters de ses attributs, et définissez le constructeur **Formation(intitulé: string, nbrJours: number , stagiaires: Stagiaire [])**
 6. Écrivez une méthode **calculerMoyenneFormation()** qui retourne la moyenne d'un objet de type formation (la moyenne des moyennes des stagiaires)
 7. Écrivez une méthode **getIndexMax()** qui retourne l'indice du stagiaire dans le tableau **stagiaires** ayant la meilleure moyenne de la formation.
 8. Écrivez une méthode **afficherNomMax()** qui affiche le nom du premier stagiaire ayant la meilleure moyenne d'une formation.
 9. Écrivez une méthode **afficherMinMax()** qui affiche la note minimale du premier stagiaire ayant la meilleure moyenne d'une formation.
 10. Écrivez une méthode **trouverMoyenneParNom(nom: string)** qui affiche la moyenne du premier stagiaire dont le nom est passé en paramètre.
 11. Dans un fichier **main.ts**, testez toutes les méthodes réalisées dans les questions précédentes (créez par exemple trois objets **Stagiaire** et affectez les à une même formation et faites appel aux quatre dernières méthodes que vous avez implémentées).

Exercice 3

Considérons les deux classes **Personne** et **Adresse**. Les attributs de la classe **Adresse** sont :

- **rue** : un attribut privé de type chaîne de caractères.
- **ville** : un attribut privé de type chaîne de caractères.
- **codePostal** : un attribut privé de type chaîne de caractères.

Les attributs de la classe **Personne** sont :

- **nom** : un attribut privé de type chaîne de caractères.
- **sexe** : un attribut privé de type chaîne de caractères (cet attribut aura comme valeur soit 'M' soit 'F').
- **adresses** : un attribut privé de type tableau d'objet de la classe **Adresse**.

1. Créez les deux classes **Adresse** et **Personne** dans deux fichiers séparés. N'oubliez pas de définir les getters/setters et les constructeurs.
2. Créez une troisième classe **ListePersonnes** ayant un seul attribut **personnes** : un tableau d'objets **Personne**. Définissez les getters/setters et le constructeur de cette classe.
3. Écrivez la méthode **findByNom(s: string)** qui permet de chercher dans le tableau **personnes** si l'attribut **nom** d'un est égal à la valeur du paramètre **s**. Si c'est le cas, elle retourne le premier objet correspondant, sinon **null**.
4. Écrivez la méthode **findByCodePostal(cp: string)** qui permet de vérifier dans le tableau **personnes** si un objet possède au moins une adresse dont le code postal égal au paramètre **cp**. Si c'est le cas, elle retourne **true**, sinon **false**.
5. Écrivez la méthode **countPersonneVille(ville: string)** qui permet de calculer le nombre d'objets dans le tableau **personnes** ayant une adresse dans la ville passée en paramètre.
6. Écrivez la méthode **editPersonneNom(oldNom: string, newNom: string)** qui remplace les noms de personnes ayant un nom égal à la valeur **oldNom** par **newNom**.
7. Écrivez la méthode **editPersonneVille(nom: string, newVille: string)** qui remplace les villes de personnes ayant un nom égal à la valeur du paramètre **nom** par **newVille**.

Exercice 4

Considérons une classe Java appelée **MaDate** ayant les trois attributs suivants :

- **jour** : un attribut privé de type entier.
- **mois** : un attribut privé de type entier.
- **année** : un attribut privé de type entier.

1. Créez la classe **MaDate**
2. Définissez les getters et setters des trois attributs.
3. Définissez un constructeur avec trois paramètres **MaDate(int jour, int mois, int année)**
4. Écrivez la méthode **ajouterUnJour** qui permet d'ajouter un jour à notre date et faire des modifications, si nécessaire, pour les deux attributs **mois** et **année**. **Attention**, il faut traiter tous les cas. Par exemple si les trois attributs **jour**, **mois** et **année** contiennent respectivement les valeurs 31, 12 et 2016, alors la méthode **ajouterUnJour** doit affecter la valeur 1 à **jour**, 1 à **mois** et 2017 à **année**. Et s'ils contiennent 28, 02 et 2018 alors les nouvelles valeurs après modification seront respectivement 29, 02 et 2018.
5. Utilisez la méthode de la question précédente pour écrire la méthode **ajouterPlusieursJours(int n)** : **n** étant le nombre de jours à ajouter à la date enregistrée dans les trois attributs.
6. D'une façon similaire, définissez les méthodes **ajouterUnMois** et **ajouterUnAn()**.
7. Dans la méthode **main**, testez toutes les méthodes réalisées dans les questions précédentes.