

Qualité de développement

Exceptions en TypeScript

Exercice 1

Soit la fonction :

```
function creeInitiales(nom: string, prenom: string): string {  
    return nom[0].toUpperCase() + prenom[0].toUpperCase();  
}
```

Remarque : la méthode `toUpperCase` permet de transformer une chaîne de caractères en majuscules.

1. Dans un premier temps, tester cette fonction avec votre nom et votre prénom. Puis, avec une chaîne vide pour le nom, puis avec une chaîne vide pour le prénom.
2. Quelle exception est levée ?
3. Sécuriser la fonction avec une ou plusieurs conditionnelles permettant de lever une exception de type `Error` lorsque les paramètres passés ne sont pas valides.
4. Créez trois classes d'exceptions : `NomVideError`, `PrenomVideError` et `NomEtPrenomVideError` qui héritent de la classe `Error`. Chaque classe doit avoir un constructeur qui prend en paramètre un message et qui appelle le constructeur de la classe parente avec le message en paramètre. Chaque classe doit avoir une propriété `name` qui contient le nom de la classe.
5. Modifier la fonction `creeInitiales` pour qu'elle lance une exception de type `NomVideError` si le nom est vide, une exception de type `PrenomVideError` si le prénom est vide et une exception de type `NomEtPrenomVideError` si les deux chaînes sont vides.
6. Ecrire un programme qui appelle la fonction `creeInitiales` avec des chaînes vides. Gérez les exceptions de type `NomVideError`, `PrenomVideError` et `NomEtPrenomVideError`. Affichez le message de l'exception. Affichez "Fin du programme" à la fin du programme.
7. Prenez du recul sur votre code. La création de trois classes d'exception vous semble-t-elle justifiée ? Si non, comment mieux faire ? Implémenter la solution retenue.

Exercice 2

1. Ecrivez une classe `Etudiant` qui contient un attribut `INE`, son nom et son prénom. Un code `INE` peut être de deux formes possibles :

- 10 chiffres puis 1 lettre
 - 9 chiffres puis 2 lettres
2. Ecrivez une fonction `setINE` qui prend en paramètre un INE et qui vérifie que le format est correct. Si le format est incorrect, la fonction doit lever une exception de type `INEError` avec le message `"Format incorrect"`. Si le format est correct, la fonction doit affecter la valeur au champ INE de l'objet.

Indication : on peut tester si un caractère est une lettre avec la fonction suivante :

```
function isLetter(c: string): boolean {  
    return (c >= "a" && c <= "z") || (c >= "A" && c <= "Z");  
}
```

3. Utiliser cette fonction avec des données saisies par l'utilisateur et bouclez tant qu'il y a une exception.

Exercice 3

1. Créer une fonction qui à partir d'une date au format chaîne « 22/05/1994 », renvoie un tableau de trois entiers correspondant aux jours, mois, et année.
2. Lever une exception que vous aurez définie si la chaîne n'est pas au bon format.
3. Lever une exception que vous aurez définie si les jours ou mois ne sont pas dans l'intervalle 1-31, 1-12.

Indication : utiliser la méthode `split` de la classe `String` pour découper la chaîne en trois parties. Par exemple :

```
let date = "22/05/1994";  
let tab = date.split("/");
```

retourne un tableau de trois chaînes de caractères : `["22", "05", "1994"]`.

Indication : on peut tester si un caractère est une chiffre avec la fonction suivante :

```
function isDigit(c: string): boolean {  
    return c >= "0" && c <= "9";  
}
```

Indication : vous pouvez utiliser la fonction `parseInt` pour convertir une chaîne de caractères en entier.

- Utiliser cette fonction avec des données saisies par l'utilisateur et bouclez tant qu'il y a une exception. Vous indiquerez à l'utilisateur que la saisie est incorrecte et quel est le problème.
- Si la chaîne est au format US « MM-JJ-AAAA », indiquez-le par une exception dédiée.

Exercice 4

On vous fournit le code suivant :

```
function calcule(  
    tableau: Array<number>,  
    indice: number,  
    operateur: string,  
    operande: number  
): number {  
    switch (operateur) {  
        case "+":  
            return tableau[indice] + operande;  
        case "-":  
            return tableau[indice] - operande;  
        case "*":  
            return tableau[indice] * operande;  
        case "/":  
            return tableau[indice] / operande;  
    }  
    return 0;  
}
```

- Exécutez cette fonction avec divers paramètres "qui marchent"
- Trouvez trois cas d'erreurs différents ; implémentez-les. Constatez les résultats donnés par TypeScript
- Gérez les erreurs précédentes en levant des exceptions dans la fonction `calcule`. Si une classe d'exception de TypeScript correspond au problème, utilisez-la. Sinon, créez une classe d'exception et utilisez-la dans la fonction. Un cas `default` peut être ajouté au `switch...case` pour améliorer la fonction.
- Entourez chaque appel à `calcule` d'un bloc `try...catch` afin que le programme continue de se dérouler en cas d'erreur.