TP N°5

Chapitre : Gestion des exceptions en Java

Licence : INFO-AP | G-INFO

Université Moulay Ismaïl – Faculté des Sciences Meknès

Pr. MAROUANE NAZIH & Pr. ISMAILI ALAOUI ELMEHDI

Exercice — Gestion d'un support de cours numérique

Objectif pédagogique

Concevoir une application Java orientée objet permettant de modéliser un support de cours numérique, en appliquant une gestion rigoureuse des exceptions et des règles d'affichage normalisées.

Contexte

Une université souhaite gérer des supports de cours numériques accessibles en ligne. Chaque support est caractérisé par un identifiant numérique, un intitulé et une durée. Certains supports proposent un lien d'accès généré automatiquement.

Le système doit garantir la cohérence des données dès la création des objets et un comportement fiable lors de leur utilisation.

1. Classe d'exception métier : `SupportInvalideException`

Créer une classe `SupportInvalideException` héritant de `Exception`.

Rôle : Signaler toute incohérence lors de la création d'un support de cours.

Cette exception doit être levée exclusivement dans les constructeurs si :

- l'identifiant numérique est inférieur ou égal à zéro ;
- l'intitulé est nul ou vide ;
- la durée est inférieure ou égale à zéro.

Aucune validation métier ne doit être effectuée en dehors des constructeurs.

2. Classe `SupportCours`

La classe `SupportCours` représente un support de cours numérique.

Attributs :

- `id : int` — identifiant numérique du support ;
- `intitule : String` — intitulé du support ;
- `duree : int` — durée du cours en heures.

Règles de validité :

- `id > 0` ;
- `intitule` non nul et non vide ;
- `duree > 0`.

Contraintes de normalisation (obligatoires) :

- suppression des espaces inutiles en début et fin de l'intitulé ;
- affichage de l'intitulé avec la première lettre en majuscule et le reste en minuscules.

Les méthodes suivantes de la classe `String` doivent être utilisées explicitement : `trim()`, `toLowerCase()`, `substring()`.

Méthodes attendues :

- un constructeur validant les attributs et appliquant la normalisation ;
- `void afficher()` : affiche les informations du support ;
- `void modifierDuree(int nouvelleDuree)` :
  - met à jour la durée du support ;
  - déclenche une `IllegalArgumentException` si `nouvelleDuree <= 0`.

3. Interface `Accessible`

Créer une interface `Accessible` définissant la méthode suivante :

- `String getLienAcces()`

Contrainte d'utilisation :

- l'appel de la méthode `getLienAcces()` sur une référence nulle entraîne une `NullPointerException`.

Le lien retourné doit respecter le format :

`https://cours.univ.ma/{ID}`

4. Classe `CoursEnLigne`

La classe `CoursEnLigne` représente un support de cours accessible en ligne.

Relations :

- hérite de la classe `SupportCours` ;
- implémente l'interface `Accessible`.

Méthodes attendues :

- implémentation de la méthode `getLienAcces()` ;
- redéfinition de la méthode `afficher()` afin d'inclure le lien d'accès.

5. Exceptions prédéfinies à utiliser

En complément de l'exception métier, l'application doit utiliser explicitement les exceptions prédéfinies suivantes :

- `NullPointerException` :
  - levée lors de l'appel de `getLienAcces()` sur une référence non initialisée.
- `IllegalArgumentException` :
  - déclenchée dans la méthode `modifierDuree(int)` lorsque la valeur transmise est invalide.

6. Classe principale

La classe principale de l'application doit implémenter les trois scénarios suivants :

- **Scénario nominal** :
  - créer un objet de type `CoursEnLigne` dans un bloc `try` ;
  - afficher les informations du support ;
  - générer et afficher le lien d'accès.
- **Scénario d'erreur métier** :
  - intercepter l'exception `SupportInvalideException` levée lors de la création d'un support non valide ;
  - afficher un message d'erreur explicite sans interrompre brutalement le programme.

— **Scénario d'erreur prédéfinie** :

  — provoquer une exception prédéfinie lors de l'utilisation d'une méthode (par exemple, appel de `modifierDuree(int)` avec une valeur incorrecte ou appel de `getLienAcces()` sur une référence nulle) ;

  — intercepter cette exception et afficher un message d'erreur approprié.

Contraintes :

— la classe principale ne doit contenir aucune validation métier ;

— aucune normalisation de chaînes de caractères ne doit être réalisée dans cette classe ;

— toute gestion des incohérences doit être effectuée exclusivement par les classes métier ;

— le programme doit se terminer normalement dans tous les scénarios.

Exemples d'exécution

Scénario 1 — Création valide du support

```
Création du support...
Support créé avec succès
ID : 101
Intitulé : Programmation java
Durée : 10 h
Lien : https://cours.univ.ma/101

Fin du programme
```

Scénario 2 — Erreur de création (exception personnalisée)

Tentative de création d'un support avec des données invalides (identifiant inférieur ou égal à zéro ou durée invalide).

```
Création du support...
Erreur métier : données du support invalides

Fin du programme
```

Scénario 3 — Erreur d'utilisation (exception prédéfinie)

Tentative de modification de la durée avec une valeur incorrecte ou appel d'une méthode sur une référence non initialisée.

```
Création du support...
Support créé avec succès

Erreur standard : argument invalide

Fin du programme
```