HAUTE ÉCOLE DE
NAMUR-LIÈGE-LUXEMBOURG
Catégorie technique
Année académique : 2020-2021

TI215 : programmation orientée objet 1
Exercices : série 7

Objectif : Gérer les cas d'erreurs via le mécanisme des exceptions

Toutes les variables d'instance et les variables de classe doivent être déclarées *private* !
Pour chacune des classes, prévoyez les getters et setters nécessaires.

Exercice 1

Créez une classe intitulée **Personne** qui contient les variables d'instances suivantes :
de type String :
- nom
- prénom

de type char :
- sexe

de type GregorianCalendar :
- dateNaissance

Prévoyez un constructeur qui permet d'instancier une personne à partir d'un nom, un prénom, un sexe, une année, un mois et un jour de naissance. Le constructeur devra donc instancier un objet de type GregorianCalendar à partir de l'année, du mois et du jour dans le mois et l'associer à la variable dateNaissance.
Les seules valeurs permises pour le sexe sont 'm', 'M', 'f' et 'F'. Une **exception** doit être **générée** si on tente d'affecter une autre valeur (erronée) pour le sexe. Cette exception doit être détectée par la méthode setter correspondant (setSexe( )).
Ce constructeur devra impérativement faire appel à la méthode setSexe( ) pour garnir la variable d'instance sexe. Ce constructeur doit **propager l'exception** éventuellement levée.

Prévoyez également la méthode toString( ) qui retourne uniquement le prénom suivi du nom :
ex : *Pierre Legrand*

Créez une classe intitulée **Etudiant** qui est une **sous-classe de Personne** et qui contient en plus les variables d'instances suivantes :
de type String :
- section

de type int :
- année

Les seules valeurs permises pour l'année sont 1, 2 ou 3. Une **exception** doit être **générée** si on tente d'affecter une autre valeur (erronée) pour l'année. Cette exception doit être détectée par la méthode setter correspondant (setAnnée( )).
Les seules valeurs permises pour la section sont : *compta*, *droit*, *market*, *infoGestion*, *technoInfo* et *sécu*. Une **exception** doit être **générée** si on tente d'affecter une autre valeur (erronée) pour la section. Cette exception doit être détectée par la méthode setter correspondant (setSection( )).

RAPPEL : Pour comparer un objet de type String à une valeur, il vaut mieux avoir recours à la méthode equals(...) qui s'applique à un objet de type String. L'argument de la méthode equals est la valeur à comparer. La méthode retourne un booléen (true si vrai, false sinon).
En effet, l'instruction : if (section = = "compta") est illogique puisqu'on compare une valeur ("compta") à une référence vers un objet (section). Il est préférable (plus portable et moins aléatoire) d'écrire : if (section.equals("compta") = = true), ou encore en version raccourcie :
if (section.equals("compta")).

Prévoyez un constructeur qui doit impérativement faire appel à la méthode setAnnée() pour garnir la variable d'instance année et à la méthode setSection() pour garnir la variable d'instance section. Ce constructeur doit ***propager toutes les exceptions éventuellement générées.***

Prévoyez également la méthode toString() qui doit **ajouter** à la description de la personne un message précisant l'année et la section d'inscription.

Ex : *Pierre Legrand est inscri**t** en 2e technoInfo*
*Julie Petit est inscri**te** en 3e droit*

Créez une classe **Principal** contenant la méthode **main**. Cette méthode doit :
- essayer (try-catch) de créer un étudiant et d'afficher à l'écran la description de cet étudiant *(via appel implicite à la méthode toString( )*);
- en cas d'erreur (exception capturée), afficher via System.out.println le message correspondant à l'erreur (càd la description de l'exception via appel implicite à la méthode toString( ) de la classe correspondant à l'exception).

Suggestion : Ne prévoir qu***'un seul bloc try*** dans la méthode main de la classe Principal. Ce bloc contiendra à la fois l'instruction de création de l'étudiant ET l'instruction d'affichage de l'étudiant.

Exercice 2

Reprenez la classe Principal :
En cas de réussite (pas d'erreur) :
Afficher un message de confirmation de l'inscription de l'étudiant via une boîte de dialogue (JOptionPane). Le titre de la boîte de dialogue est : *Confirmation d'inscription*, et le contenu du message est la description de l'étudiant créé (via appel implicite à la méthode toString() sur l'étudiant créé).
En cas d'erreur :
Afficher les messages d'erreurs via des boîtes de dialogue (JOptionPane) dont le titre est *Section erronée, Année erronée ou Sexe erroné en fonction de l'erreur* et le contenu du message est la *description de l'exception capturée* (via appel implicite à la méthode toString() sur l'exception).

NB. Pour rappel, ajouter l'instruction **System.exit(0)** à la fin de la méthode main de la classe Principal.

Exercice 3 (facultatif)

Idem, mais les ***informations nécessaires pour créer l'étudiant doivent être obtenues auprès de l'utilisateur via de boîtes de dialogue*** (JOptionPane).
Les informations à demander sont : le nom, le prénom, le sexe, l'**année** de naissance, le **mois** de la date de naissance, le **jour** de la date de naissance, la section et l'année.

Pour rappel, la méthode statique showInputDialog(…) de la classe JOptionPane permet de récupérer un **String**. L'objet de type String obtenu

pour l'année de naissance, le mois de la date de naissance, le jour de la date de naissance ou l'année d'inscription de l'étudiant doit donc *être transformé en un entier* (cf. méthode **parseInt(...)** de la classe **Integer**).
Il en va de même pour la variable sexe. L'objet de type String obtenu auprès de l'utilisateur doit être transformé en un caractère (type **char**). Pour ce faire, on peut utiliser la méthode **charAt(...)** qui peut être appelée sur un objet de type String. Cette méthode permet *d'extraire un caractère d'une chaîne de caractères*. La méthode charAt(…) prend un entier comme argument représentant la position du caractère à extraire (0 pour le premier caractère, 1 pour le deuxième,…) et retourne un caractère (de type char).

Suggestion : placer les instructions de saisie des informations (via des boîtes de dialogues) dans le bloc try (un seul bloc try dans la méthode main de la classe Principal).

NB. D'autres exceptions que celles que vous avez prévues peuvent être générées (ex : lors de l'appel à la méthode parseInt(…), si on tente de transformer en un entier une chaîne de caractères qui ne contient pas que des chiffres). Prévoyez par conséquent *la gestion de ce genre d'exception*, en affichant par exemple le message correspondant à cette exception.
Il suffit de prévoir un bloc catch supplémentaire *après tous les autres blocs catch* : catch (**Exception** e) et d'afficher alors via une boîte de dialogue le message correspondant (via appel à la méthode getMessage( ) sur l'exception e).

Exercice 4 (facultatif)

Idem, mais proposez à l'utilisateur de ***créer autant d'étudiants qu'il le souhaite***. Après chaque création d'étudiant, demander à l'utilisateur (via une boîte de dialogue) s'il désire continuer. Bouclez tant que l'utilisateur le désire.
NB. *Déclarez les variables locales* de la méthode main (nécessaires pour saisir les informations auprès de l'utilisateur et pour créer l'étudiant) *en dehors de la boucle*.

Exercice 5

Idem, mais affichez à la fin du programme (càd dès que l'utilisateur désire arrêter), le ***pourcentage de filles inscrites***. Pour ce faire, utilisez impérativement des **variables de classes** déclarées **private** dans la classe

Etudiant, et non pas des variables locales dans la méthode main de la classe Principal.

Exercice 6

Ajouter la méthode **nomUtilisateur** dans la classe **Etudiant**. Cette méthode doit retourner le nom d'utilisateur de l'étudiant qui doit être composé (dans l'ordre) des deux premières lettres de la section, suivi de l'année, suivi du nom complet de l'étudiant et se terminer par l'initiale du prénom.

Par exemple, pour les étudiants Pierre Legrand inscrit en 2e technoInfo et Julie Petit inscrite en 3e droit, les noms d'utilisateur sont respectivement ***te2LegrandP*** *et* ***dr3PetitJ****.*

NB. La méthode **subString**(…) s'applique à un objet de type String et permet d'*en extraire une chaîne de caractères*. Les deux arguments de la méthode subString() sont de type entier et précisent respectivement l'indice du premier caractère de la chaîne à extraire (0 représente le début) et l'indice du dernier caractère de la chaîne à extraire. Cette méthode retourne un objet de type String.

Modifiez ensuite la méthode toString() de la classe Etudiant pour y incorporer le nom d'utilisateur.
Ex : *Pierre Legrand est inscrit en 2e technoInfo.*
Son nom d'utilisateur est te2LegrandP.

Exercice 7

Ajouter la méthode **âge** dans la classe **Personne**. Cette méthode doit retourner l'âge de la personne à partir de la date système et de sa date de naissance. Soyez précis : si la date du jour est le 10 février 2018 et la date de naissance de la personne est le 12 février 2000, cette personne a 17 ans et non 18.

Modifiez ensuite la méthode toString() de la classe Personne pour y incorporer l'âge.
Ex : *Pierre Legrand* ***(18 ans)***

Exercice 8

Créez une classe intitulée **Cours** qui contient les variables d'instances suivantes :
de type String :
- libellé (libellé du cours)

*de type **Personne** :*
- professeur (professeur qui donne le cours)

Prévoyez un constructeur ainsi que la méthode toString( ) qui retourne la description d'un cours sous le format suivant :
ex : le cours de *probabilité* donné par *Luc Degroot* (*35* ans)

N.B. Les valeurs en italique varient d'un cours à l'autre.

Créez une classe intitulée **Bisseur** qui est une **sous-classe de Etudiant** et qui contient les variables d'instances supplémentaires suivantes :
*de type **tableau de Cours** (5 maximum):*
- coursDispenses (liste des cours dont est dispensé l'étudiant bisseur)

de type int :
- nbDispenses (nombre de dispenses auxquelles à droit l'étudiant bisseur : correspond au nombre d'éléments réellement utilisés dans le tableau)

Prévoyez un constructeur ainsi que la méthode toString() qui retourne la description d'un bisseur sous le format suivant :
ex : Le bisseur *Pierre Legrand* (*18* ans) est inscrit en *2e technoInfo.*
Son nom d'utilisateur est *te2LegrandP.*

N.B. Les valeurs en italique varient d'un étudiant bisseur à l'autre.

Pour rappel, **toutes les variables d'instance (y compris de type tableau) doivent être déclarées *private* !**

Prévoyez également les méthodes suivantes :
- la méthode **ajouterDispense** qui reçoit un argument de type Cours et qui ajoute ce cours *après le dernier cours existant dans le tableau* (cette méthode ne retourne rien) *;*
- la méthode **getDispense** qui reçoit en argument le numéro (**num**) du cours à récupérer dans le tableau et qui retourne le cours se trouvant en posistion **num** (1er élément du tableau : num = 1) ;

Créez une classe nouvelle classe **Principal** contenant la méthode **main**. Cette méthode doit **essayer** (try-catch) de :

- créer des professeurs ;
- créer 3 cours donnés par ces professeurs ;
- créer un étudiant bisseur ;
- afficher à l'écran la description de cet étudiant *(via appel implicite à la méthode toString()*);
- dispenser cet étudiant bisseur des 3 cours que vous avez précédemment créés : càd ajouter ces 3 cours à la liste (tableau) des cours dont est dispensé ce bisseur ;
- afficher les cours dont est dispensé ce bisseur : càd boucler sur les dispenses réellement octroyées à cet étudiant bisseur (**en bouclant sur le nombre contenu dans la variable d'instance nbDispenses**) et afficher la description de chacun de ces cours ;
- afficher uniquement la description des professeurs qui donnent les cours dont est dispensé ce bisseur ;

Exercice 9

Gérer les cas d'erreur liés à l'utilisation des tableaux :

A. Lorsqu'on tente d'ajouter un 6e cours. Prévoyez la classe d'exception **TableauPlein** (ne contenant aucune variable d'instance) qui permet d'afficher le message suivant :

   Le nombre de dispenses accordées est de 5 maximum.

   Le nombre de dispenses est donc déjà atteint !

B. Lorsqu'on tente d'utiliser la méthode **getDispense** avec un argument en entrée (numéro de cours) non valide : càd avec un numéro de cours négatif ou nul ou un numéro de cours plus grand que le nombre réel de dispenses accordées. Prévoyez la classe d'exception **MauvaisNumero** qui comprend deux variables d'instance : la valeur du ***mauvais indice*** et le ***nombre réel de cours*** dans le tableau (cfr nbDisp). Cette classe doit permettre l'affichage, selon le cas, d'un message signalant

   - soit que le numéro proposé est <= 0

     (ex : *Le numéro –2 que vous avez proposé est <=0*)

   - soit que le numéro proposé ne correspond pas à un cours dont est dispensé l'étudiant en rappelant le nombre de cours dont ce dernier est dispensé

     (ex : *L'étudiant bisseur n'est dispensé que dans 3 cours. Le numéro 4 que vous avez proposé ne correspond donc pas à un cours dont l'étudiant est dispensé*).

Pour tester ces cas d'erreurs, sachant que l'étudiant bisseur n'a que trois dispenses accordées, tentez :

- d'afficher le cours (la dispense) numéro 4 de cet étudiant
- d'afficher le cours (la dispense) numéro –1 de cet étudiant ;
- d'afficher le cours (la dispense) numéro 0 de cet étudiant ;
- d'ajouter un 4e, 5e puis 6e cours dans lequel serait dispensé l'étudiant bisseur ;

Exercice 10

Même exercice que 8 mais en utilisant une ArrayList plutôt qu'un tableau.
Ne modifiez rien dans la classe Principal. Modifiez le code des méthodes ***ajouterDispense*** et ***getDispense*** et supprimez ***getNbDisp***.
Attention, les méthodes ajouterDispense ne doit plus lever des exceptions (plus aucun throws dans sa signature) !