# Programmation Orientée Objet
## TD-TP N°11 : Exceptions

## Objectifs :

- Se familiariser avec le mécanisme des exceptions.
- Gérer des exceptions contrôlées.

### Exercice 1 : exceptions prédéfinies

Il existe en java des exceptions prédéfinies : ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Le point commun à ces exceptions est qu'elles sont toutes d'une catégorie d'exception que l'on n'a pas besoin de déclarer dans la clause throws des méthodes parce qu'elles sont fréquentes.

Complétez le programme suivant pour que les erreurs susceptibles de se produire soient gérées.

```
class Exercice1{
        static int[] tableau = {17, 12, 15, 38, 29, 157, 89, -22, 0, 5};
        static int division(int indice, int diviseur){
                return tableau[indice]/diviseur;
        }
        public static void main(String[] args){
                int x, y;
                ecrire("Entrez l'indice de l'entier à diviser: ");
                x = lireInt();
                ecrire("Entrez le diviseur: ");
                y = lireInt();
                ecrire("Le résultat de la division est: ");
                ecrire(division(x,y));
        }
}
```

## Exercice 2

Reprendre le TP 7 (gestion d'une bibliothèque)

**Rappel :** Au sein d'une bibliothèque chaque document est identifié par un numéro unique qui lui est attribué lorsque le document est ajouté à la bibliothèque (et pas avant).

L'objectif de cet exercice est de compléter la classe *Bibliothèque* donnée ci-après :

```
public class Bibliotheque
{
        private String nomBibliotheque  ;
        private Document [ ] listeDocs     ;          // Contient les documents.
                                                      // vous pouvez utiliser une collection (ArrayList)
        private int prochainNumero        ;          // Prochain numéro de document à attribuer.
        private static final int CAPACITE = 100 ;   // Nombre maximum de documents
…
}
```

Noter que :
*i)* les variables *listeDocs, prochainNumero* et la constante *CAPACITE* sont à usage strictement interne. Elles ne sont donc ni connues ni accessibles de l'extérieur de la classe.

*ii) prochainNumero* est le numéro de document que la Bibliothèque affectera au prochain document ajouté. Cette variable est incrémentée à chaque nouvel ajout de document et jamais décrémentée.

1. Écrire la méthode *getDocument(int num)* qui retourne le document de numéro *num*. Si aucun document de numéro *num* n'est présent dans la *Bibliothèque* courante, créer et déléguer une exception de type *Exception* avec un message explicite. Tester cette méthode sur la bibliothèque *bibli*. Que se passe-t-il sachant que *bibli* est actuellement vide ?

2. Écrire la méthode *retirerDocument(int num)* qui élimine de la *Bibliotheque* courante le document de numéro *num* et le retourne. Prévoir toute exception utile. Tester cette méthode sur la bibliothèque *bibli*. Que se passe-t-il sachant que *bibli* est actuellement vide ?

3. Écrire la méthode *ajouterDocument(Document d)* qui ajoute le document référencé par *d* à la *Bibliothèque* courante. Lever une exception de type *Exception* si la *Bibliothèque* courante est déjà remplie. Tester cette méthode sur *bibli* en utilisant la méthode *saisirDocument( )* de la classe *Bibliotheque*.

*4.*
   a. Créer une nouvelle classe *DocumentDejaPresent* qui étend la classe *Exception*. Cette classe comprendra une variable privée de type *Document*, une méthode *getDocument( )* retournant la valeur de cette variable et une méthode *toString( ).* Une exception de cette classe sera levée lorsqu'on tente d'ajouter un média qui existe déjà dans la médiathèque courante.
   b. Compléter la méthode *ajouterDocument* de façon à ce qu'une exception de ce type soit levée et déléguée lorsqu'on tente d'ajouter un document déjà présent. Tester sur la bibliothèque *bibli* en tenant compte du fait que la méthode *ajouterDocument* est désormais susceptible de lever et déléguer 2 types d'exceptions.
   c. Modifier le *main* de *TestBibliotheque* de façon à ajouter des documents jusqu'à ce que la bibliotheque bibli soit remplie (le programme doit pouvoir continuer si une exception de type *DocumentDejaPresent* survient).

.