# Fiche d’exercices – Robustesse

Pour chaque exercice ci-dessous, fournir le code **Java**, ainsi que la **modélisation UML** (projet *NetBeans* & *Visual Paradigm for NetBeans* compacté sur *Nom-FicheRobust-ExerciceX.zip*).

## Exceptions

1. Créer une application qui reçoit en entrée (*par ligne de commande*) un nom de fichier. S’il s’agit d’un fichier simple, l’application affiche son contenu (en supposant qu’il s’agit d’un fichier texte). S’il s’agit d’un répertoire, l’application liste le contenu du répertoire. L’application doit gérer les exceptions en cas de fichier (ou répertoire) absent ou illisible.
2. Proposer une application capable de compter les mots, les voyelles et les consonnes. L’application doit être capable de lire le texte à compter à partir d’un fichier texte ou du clavier.
3. Modifier la classe de Tableau d’entier proposée précédemment de manière à ce qu’elle puisse soulever l’exception « ArrayIndexOutOfBoundsException » lorsqu’on essaie d’accéder, aussi bien pour récupérer que pour insérer, une valeur à position invalide (inférieure à zéro ou supérieur à la taille du tableau).

Modifier l’application qui utilise cette classe de manière à traiter cette exception, mais aussi toute autre exception liée à une mauvaise entrée de la part de l’utilisateur.

1. Proposer deux nouvelles exceptions, nommées « MyStackOverflowException » et « MyEmptyStackException ». Etendre les classes implémentant la notion de Pile (« *Stack*»), proposées dans la fiche d’exercice précédente, de manière à ce qu’elles lancent ces exceptions lorsqu’on essaie d’ajouter un nouvel élément à une pile déjà pleine, ou encore lors qu’on essaie de récupérer de données d’une pile vide.

## Tests Unitaires

1. Créer un jeu de tests unitaires pour vérifier le bon fonctionnement de la classe de calcul statistiques proposée précédemment (fiche d’exercices n° 2 – Modularité).
2. Créer de tests unitaires JUnit pour vérifier le bon fonctionnement de la classe étudiant proposée dans la fiche d’exercice précédente. Vérifier notamment :
   1. Le calcul de la moyenne et de l’admissibilité de l’étudiant
   2. L’usage de valeurs non-admis (notamment les notes inférieures à 0 ou supérieures à 20)
   3. La bonne définition des informations concernant un étudiant (nom, prénom…)
3. Créer de tests unitaires en JUnit pour vérifier le bon fonctionnement de la classe Personne proposée dans la fiche d’exercice précédente. Vérifier notamment la comparaison (méthode compareTo) et l’égalité (méthode equals) entre deux objets de cette classe.
4. Créer de tests unitaires en JUnit pour vérifier les classes Tableau d’entier et Tableau Objets proposés dans les fiches d’exercice précédentes. Vérifier notamment :
5. L’usage d’indices : i < 0, 0 < i < taille, taille < i…
6. L’usage de valeurs non-admis
7. Bon fonctionnement général de chaque méthode
8. Créer de tests unitaires JUnit pour vérifier le bon fonctionnement de la classe de liste double chaîne proposée dans la fiche d’exercices précédente.
9. Créer de tests unitaires JUnit pour vérifier le bon fonctionnement des implémentations de l’interface Pile (« Stack ») proposées dans les fiches précédentes.