En-têtes & commentaires

- Ecrits au format Javadoc
- En-têtes courts pour les méthodes, généralement 1-2 phrases suffisent
- Préciser les exceptions susceptibles d'être levées (@throws)
- Lorsqu'une méthode présente des cas particuliers (i.e. une valeur de retour particulière), le préciser dans l'en-tête
- Commentaires clairs et concis

Méthodes, attributs et variables

- Bonnes pratiques et conventions :
    - Constantes quand nécessaire
    - Noms parlants
- Respect des conventions Java pour le nommage
    - Classes : identificateur débutant par une majuscule, style « camel case »
      Exemple : **MaClasse**
    - Méthodes, variables et attributs : identificateur débutant par une minuscule, style « camel case »
      Exemple : **maVariable**, **maMethode()**
    - Constantes (*static final*) : identificateur entièrement écrit en majuscules, mots séparés par un « underscore »
      Exemple : **MA_CONSTANTE**
- Vérifier la validité des paramètres
    - Par exemple, si le paramètre doit être un entier positif, le vérifier au début de la méthode afin d'éviter un comportement erratique
- Utiliser la visibilité la plus restreinte
- Encapsulation des données (dans certains cas, les valeurs sont à copier pour garantir l'intégrité des données d'une classe)
- Privilégier l'utilisation des méthodes et des classes de l'API Java plutôt que d'écrire sa propre version
- Factoriser le code, éviter les doublons
- Limiter l'utilisation des exceptions aux erreurs et non au traitement de la logique « habituelle » du programme (en d'autres termes, ne pas utiliser les exceptions pour remplacer un « if »)
- Eviter les appels coûteux / inutiles (i.e : parcours multiples d'une liste alors que l'opération voulue pourraît être faite en un seul parcours)

Rapport et rendus

- Formater le code avant de le rendre afin qu'il tienne en largeur sur une page A4
- Préciser les tests effectués et leurs résultats dans le rapport
- Le code source des tests est également à rendre
- Encodage UTF-8

Diagramme de classes / UML

- Cohérence entre le diagramme UML et le code
- Clarté du diagramme
- Respect de la syntaxe UML
    - En particulier : les associations doivent avoir un nom, des cardinalités et un sens de lecture
- Qualité de la modélisation
    - Les classes nécessaires à l'exercice sont présentes
    - Pas de classes inutiles, pas d'attributs inutiles ou déduits
    - Les classes sont complètes (attributs et méthodes nécessaires à l'exercice)
    - Les associations (en particulier les cardinalités) sont cohérentes
    - Factoriser les éléments qui peuvent l'être
    - Utiliser judicieusement l'héritage et l'abstraction
    - La visibilité des éléments est cohérente
    - Utiliser correctement les propriétés statiques
- Indication des éventuelles contraintes d'intégrité (mais ne pas les indiquer lorsque c'est inutile)
- Justifier les choix effectués pour autant qu'ils soient pertinents dans le cadre du travail demandé