Java

1 Rappel POO

1.1 Classes

- package
- import
- convention de nommage

1.2 Modifieurs

- Modifieurs d'accès : public, protected, private et accès par defaut
- Static, final, abstract

1.3 Chaines de caractères

- Manipulation :
  - Méthodes et opérateurs : concat, format, split, +, contains, indexOf, startsWith, endsWith, replaceFirst, replaceAll, toLowerCase, toUpperCase
  - StringBuilder, String.join, StringJoiner, Collectors.joining
- Expressions régulières[i] :
  - Pattern, Pattern.compile, Matcher,
  - Opérateurs (Or, Nor,Range, Union,Intersection, Soustraction)
  - classes de caractères (digits, white spaces, mots, …)
  - Quantifieurs

Application : capture d'adresse IP.

1.4 Méthodes et classes abstraites

- Méthode abstraite -> classe abstraite
- Classe abstraite
- Règles de redéfinition (override)
- Polymorphisme

1.5 Interfaces

- Programmation par contrat et principe de substitution
- Polymorphisme
- Implémentation d'interfaces
  - Implémentation par une classe
  - Implémentation par une classe anonyme
- Méthodes statiques et par défaut
- Exemples :
  - De la jdk : Comparable, Comparator
  - Dans la pratique : Services, Repositories, DAOs

1.6 Collections

- List, Set, Map et Queue
- Algorithmes : Collections, Arrays, Comparator, Comparable

1.7 Exceptions

- Exceptions personnalisées
- Gestion : try, catch, finally, try with resource

1.8 Dates

Classes introduites en java 8

- Thread safety, immutabilité, standardisation ISO
- Localisation : LocalDate, LocalTime, LocalDateTime, ZonedDateTime, OffsetDateTime
- Opérations sur les dates : arithmétique, parsing de string, comparaison, Ajustements, etc…
- Périodes et durées
- Classes antérieures à java 8 et compatibilité : Date, Time et Calendar
- Formatage
- Remarques sur l'utilisation dans les applications

2 Programmation fonctionnelle

2.1 Lambdas

- Introduction par classe interne anonyme (Exemple de filtrage)
- Anonymes, référençables pouvant être passées comme paramètres de méthodes ou retournées comme type de retour.
- Dérivable des interfaces fonctionnelles : @FunctionalInterface, ->, () ->
- Exemples d'interfaces fonctionnelles :
  - Predicate[ii]: passer un objet à une condition
  - Consumer[iii]: agir sur un objet
  - Function[iv]: Transformer un T to un U
  - Supplier[v]: fournir/produire une instance de T
  - UnaryOperator[vi]: T -> T
  - BinaryOperator[vii]: (T, T) -> T
- Lambdas et collections : forEach
- Chainage des fonctions

2.2 Streams

- Filtrage : filter
- Transformation : map et flatMap
- Consommation : forEach
- Function<T,R>, BiFunction<T,U,R>, UnaryOperator<T>, BinaryOperator<T>
- Lazy evaluation et immutabilité
- Ordonnancement et groupements
- Collecte de résultats : Collectors[viii]
- Reduction :
  - Optional<T> Stream.reduce(BinaryOperator<T> accumulator)
  - T reduce(T identity, BinaryOperator<T> accumulator)
  - U reduce(U id,BiFunction<U,? super T,U> accu,BinaryOperator<U> combiner)
- Parallélisation

Application :

1. Chercher les n premiers entiers dont les carrés sont entre x et y
2. Somme de nombres complexes (combiner)
3. Moyenne des âges des personnes dont l'âge est > 40
4. Chercher la longueur de la plus longue ligne dans un fichier.
5. Chercher la ligne la plus longue dans un fichier.

---

[i] https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html
[ii] https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html
[iii] https://docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html
[iv] https://docs.oracle.com/javase/8/docs/api/java/util/function/Function.html
[v] https://docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html
[vi] https://docs.oracle.com/javase/8/docs/api/java/util/function/UnaryOperator.html
[vii] https://docs.oracle.com/javase/8/docs/api/java/util/function/BinaryOperator.html
[viii] https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html