Sujets pour l'examen final

Itérateur

1. Être capable de lire et écrire du code implémentant des itérateurs.
2. Connaître l'interface `Iterable<E>` et sa fonction `Iterator<E> iterator()`.
3. Connaître l'interface `Iterator<E>` et ses fonctions `boolean hasNext()` et `E next()`.

Fonction

Interfaces

1. Connaître les principales interfaces de fonctions : `Function<T,R>`, `Consumer<T>`, `Predicate<T>` et `Supplier<R>`.
2. Pouvoir construire des variables et des paramètres utilisant ces interfaces.

Lambda

1. Être capable d'écrire et lire des expressions Lambda.
2. Pouvoir passer une expression Lambda en paramètre à une méthode.
3. Connaître la notation pour désigner une méthode comme un lambda, exemple : `System.out::println`.

Récursion

1. Pouvoir écrire et lire du code récursif.

Stream

1. Pouvoir utiliser la classe `Stream<T>` et ces méthodes pour les manipuler.
    a. Construction :
        i. `stream()`
        ii. `Stream<T> iterate(T seed, UnaryOperator<T> f)`
        iii. `Stream<T> generate(Supplier<T> s)`
    b. Modification :
        i. `Stream<T> filter(Predicate<? super T> predicate)`
        ii. `Stream<R> map(Function<? super T,? extends R> mapper)`
        iii. `Stream<T> limit(long maxSize)`
    c. Terminaux :
        i. `T reduce(T identity, BinaryOperator<T> accumulator)`
        ii. `void forEach(Consumer<? super T> action)`
        iii. `boolean anyMatch(Predicate<? super T> predicate)`
        iv. `boolean allMatch(Predicate<? super T> predicate)`

GUI

1. Comprendre les concepts : composante, contenant, événement et réaction (listener).
2. Capable d'utiliser les éléments suivants :
   a. `JFrame`, `JPanel`.
   b. `JButton`, `JLabel`, `JTextField`, `JSlider`.
   c. `BorderLayout`, `GridLayout`, `FlowLayout`.
   d. `ActionListener`, `ChangeListener`.
3. Passer des Lambda en argument pour les `Listener`.

Algorithmes de recherche

1. Écrire, lire et utiliser un algorithme de recherche simple. $O(n)$. `boolean equals( Object obj )`.
2. Écrire, lire et utiliser un algorithme de recherche binaire. $O(\log n)$. `Comparable< E >`, `int compareTo( E e )`.

Arbre binaire de recherche

1. Comprendre les algorithmes et structures de données composant un ABR.
   a. Recherche : moyen $O(\log n)$, pire $O(n)$.
   b. Insertion : moyen $O(\log n)$, pire $O(n)$.
   c. Suppression : moyen $O(\log n)$, pire $O(n)$.
2. Écrire et lire du code récursif utilisant un ABR.

Algorithmes de trie

1. Écrire, lire et utiliser les algorithmes de tris.

	Meilleur cas	Cas moyen	Pire cas
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$
Sélection	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bulle	$O(n)$	$O(n^2)$	$O(n^2)$
Rapide	$O(n \log n)$	$O(n \log n)$	$O(n^2)$

2. Comprendre le choix de pivot et l'algorithme de partition pour le tri rapide.