

Notes de révision - INF2120

Pas officiel : LMAO :D

Je pense que de faire de la révision sur les Interfaces de fonction est une bonne idée (Predicate, Function, Supplier, Consumer).

Peut être sur l'écriture de code utilisant des chainages est aussi bien : surtout écrire du code sur papier, sans ordinateur.

Plan

- Fonction
- Type de données abstraits
- Algorithme
- Arbre Binaire de Recherche

Fonction

- Lambdas :
 - Classe abstraite vs Interface vs Interface fonctionnelle? (voir autres diapos pour les 4 interfaces de java)
 - Syntaxe ?
- Fonction récursives :
 - Ne pas oublié le cas de base (cas d'arrêt)
 - Revoir le laboratoire sur les fonctions récursives au besoin
 - Schéma récursif souvent proche des maths (ex : fibonacci)
- Instance de méthodes :
 - Classe::methode()

Lambdas

- Des exemples :

```
public class Lambdas { no usages
    public interface Fonctionnelle<T> { 6 usages
        T methode(T argument); no usages
    }

    public interface NonFonctionnelle<T> { no usages
        T methode(); no usages
        T methode2(); no usages
    }

    public interface TjrsFonctionnelle { 1 usage
        boolean test(); 1 usage
        default boolean inverse() { 1 usage
            return !test();
        }
    }
}
```

```
public static void main() { no usages
    Fonctionnelle<String> salutation = (String x) -> {
        return "Allo !" + x;
    };

    Fonctionnelle<String> salutation2 = String x -> {
        return "Allo !" + x;
    };

    Fonctionnelle<String> salutation3 = String x -> "Allo !" + x;
    Fonctionnelle<String> salutation4 = (String x) -> "Allo !" + x;
    // Fonctionnelle<String> salutation5 = String x -> "Allo !" + x; NON !
    Fonctionnelle<String> classeAnonyme = new Fonctionnelle<String>() {
        @Override no usages
        public String methode(String argument) {
            return "WOW!";
        }
    };

    // Bonus !
    TjrsFonctionnelle vrai = () -> true;
    boolean faux = vrai.inverse();
}
```

Type de données abstraits

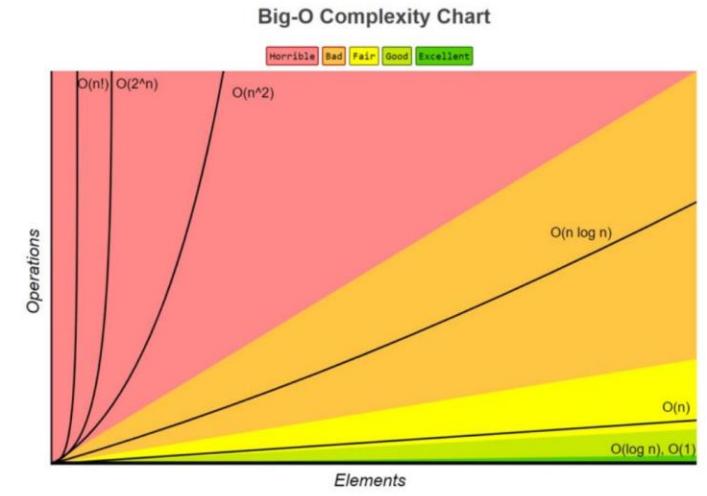
- C'est quoi ?
 - Description du quoi / garanti sur la complexité mais on sait pas comment c'est fait.
- Liste, file, pile
 - enfiler, défiler, tête
 - empiler, dépiler, sommet
 - insérer, insérer(position), get(position), set(position), supprimer(position), taille
- **ArrayList, Queue, Stack, Deque, ArrayDeque, (LinkedList)**
 - <https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html>

Algorithme

- Complexité :
 - Notation O
 - Temporelle vs spatiale ?
 - Ex :

```
public static boolean methode(int[] array, int valeur) {    no usages
    //assert estTrié(array);

    int i = 0;
    int j = array.length -1;
    int somme = array[i] + array[j];
    while(somme != valeur && i < j) {
        while(somme != valeur && i < j) {
            somme = array[i] + array[j];
            --j;
        }
        somme = array[i] + array[j];
        ++i;
    }
    return somme == valeur;
}
```



<https://www.freecodecamp.org/news/all-you-need-to-know-about-big-o-notation-to-crack-your-next-coding-interview-9d575e7eec4/>

```
public static int methode(int n) {    3 usages
    int k = 0;
    int i = 0;
    int j = i+1;

    for (; i < n; ++i){
        if (i == j){
            i = 0;
            j++;
        }
        k++;
    }

    return k;
}
```

Java :

Summary of Deque methods

	First Element (Head)		Last Element (Tail)	
	<i>Throws exception</i>	<i>Special value</i>	<i>Throws exception</i>	<i>Special value</i>
Insert	addFirst(e)	offerFirst(e)	addLast(e)	offerLast(e)
Remove	removeFirst()	pollFirst()	removeLast()	pollLast()
Examine	getFirst()	peekFirst()	getLast()	peekLast()

Comparison of Queue and Deque methods

Queue Method	Equivalent Deque Method
add(e)	addLast(e)
offer(e)	offerLast(e)
remove()	removeFirst()
poll()	pollFirst()
element()	getFirst()
peek()	peekFirst()

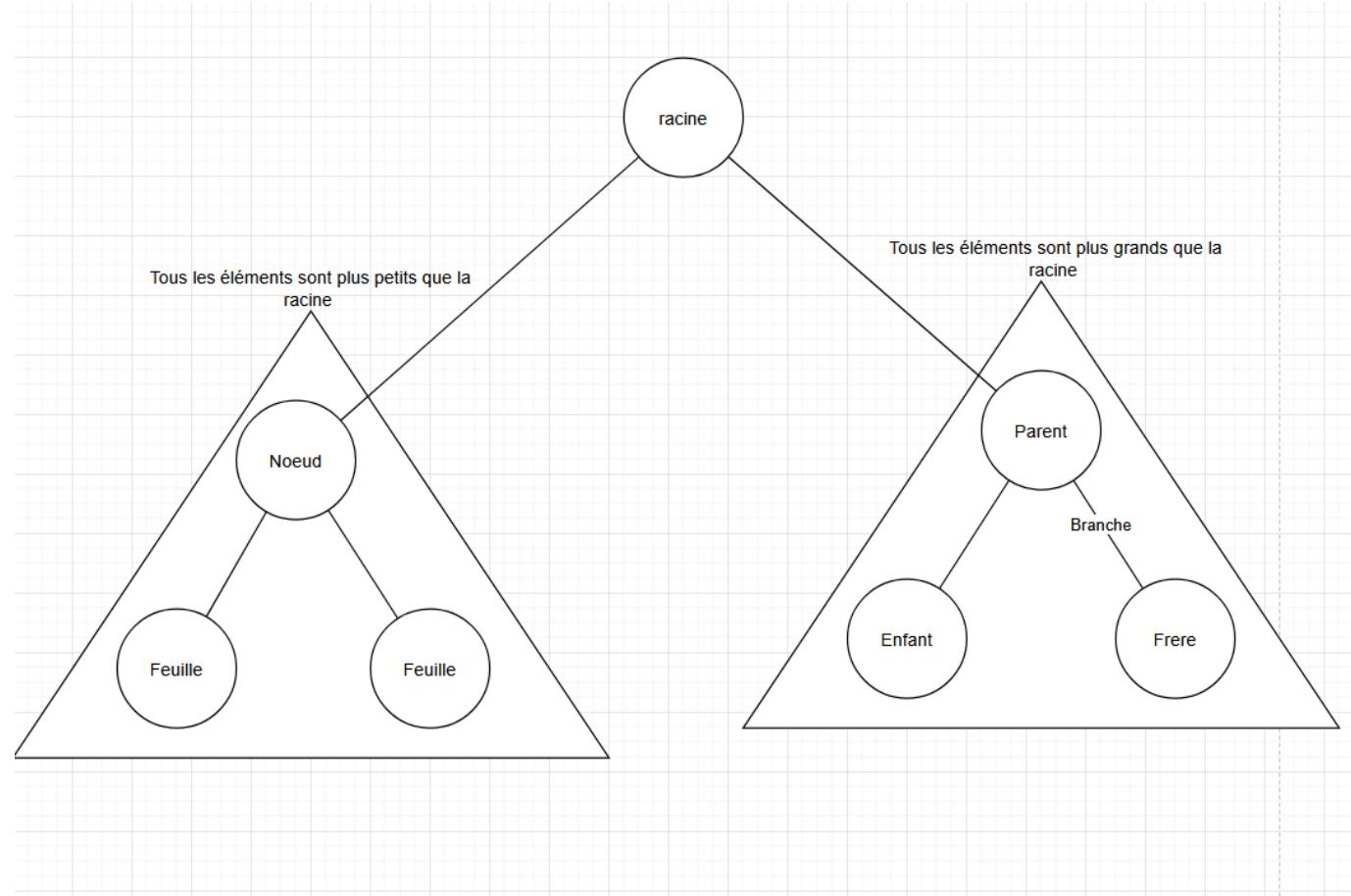
Comparison of Stack and Deque methods

Stack Method	Equivalent Deque Method
push(e)	addFirst(e)
pop()	removeFirst()
peek()	peekFirst()

Les algos vus en cours :

- Recherche linéaire, Recherche binaire
- Tri sélection, Tri insertion, Tri rapide
 - Voir les examens de préparation !

Arbre Binaire de Recherche



Merci !

Exercice de chainon a faire sur papier !

```
static public class Chainon { 7 usages
    public Chainon suivant; 2 usages
    public int valeur; 2 usages

    public boolean aSuivant() { no usages
        // Todo
    }

    public int somme(int depuis) { no usages
        // Todo : en utilisant somme(indice, depuis)
    }

    private int somme(int indice, int depuis) { no usages
        // Todo
    }

    private Chainon inverser() { 1 usage
        // Todo
    }

    public void parcourir(Consumer<Integer> i) { 2 usages
        // Todo
    }
}
```

