IUT2 Université Grenoble Alpes — Département INFO

R1.01 – TP7(B)

Vecteurs en ArrayList – Comparaisons Tris & Recherches

NE COMMENCEZ CE TP QU'APRÈS AVOIR TERMINÉ (ET TESTÉ) LE TP7A !!!

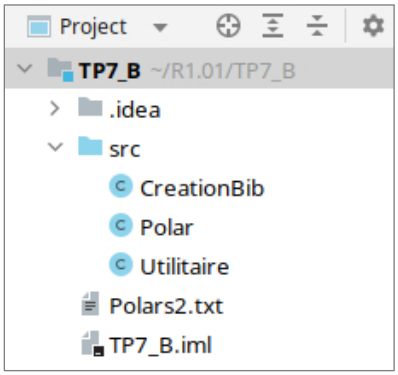
L'objectif de ce TP est de :

- *comparer les performances de certains algorithmes de tri et de recherche, ciblant des vecteurs en* `ArrayList`

Avant de commencer...

- Ouvrez un terminal et placez-vous dans votre répertoire `R1.01`
- Lancez `IJ` avec la commande `idea&`, puis à partir du menu `File` (fichiers), créez un nouveau projet `TP7_B`
- *S'il n'est pas déjà ouvert dans une autre fenêtre* `IJ` : ouvrez à partir du menu `File`, votre projet `TP7_A`
- Dans le dossier `src` du projet `TP7_A`, copiez les classes (`CreationBib`, `Polar` et `Utilitaire`), puis collez-les dans le dossier `src` du projet `TP7_B`.
- Dans le dossier du projet `TP7_A`, copiez le fichier `Polars2.txt`, puis collez-le dans le dossier du projet `TP7_B`.

Vérifiez avant de continuer que l'arborescence du dossier `TP7_B` est la suivante :



Partie A – Comparaison du tri par sélection et du tri par insertion

A1. Classe Utilitaire – outillage de procédures de tri

NOTE : VOUS POUVEZ VOUS INSPIRER DES ALGORITHMES DU COURS 7, PARTIE 3 (EN LES ADAPTANT À LA CLASSE POLAR)

Dans la classe `Utilitaire` :

A1.1. Transformez l'en-tête de la procédure `TriSelect` en :

```
public static int triSelectNbComp(ArrayList<Polar> vPolar) {
    // { } => { * vPolar est trié selon l'ORDRE (auteur, annee)
    //            selon la méthode DU TRI PAR SÉLECTION
    //          * résultat = nombre de comparaisons effectuées
    //            entre deux éléments de vPolar }
```

A1.2. Modifiez le code initial, de façon à ce que la fonction retourne le résultat attendu (nombre de comparaisons effectuées)

A1.3. Transformez l'en-tête de la procédure `TriInsertion` en :

```
public static int triInsertionNbComp(ArrayList<Polar> vPolar) {
    // { } => { * vPolar est trié selon l'ORDRE (auteur, annee)
    //            selon la méthode DU TRI PAR SÉLECTION
    //          * résultat = nombre de comparaisons effectuées
    //            entre deux éléments de vPolar }
```

A1.4. Modifiez le code initial, de façon à ce que la fonction retourne le résultat attendu (nombre de comparaisons effectuées)

A2. Classe Comparaisons – comparaison des tris étudiés

A2.1. Créez une classe `Comparaisons` et ajoutez-y une procédure `main`

A2.2. Ajoutez la déclaration : `ArrayList<Polar> mesPolars = CreationBib.lesPolars();`

A2.3. Ecrivez les instructions nécessaires à l'affichage du nombre de comparaisons effectuées par le TRI PAR SÉLECTION du vecteur `mesPolars`

A2.4. **Testez -** NOMBRE DE COMPARAISONS ATTENDU : 18 528

A2.5. Réinitialisez le vecteur mesPolars : `mesPolars = CreationBib.lesPolars();`

A2.6. Ecrivez les instructions nécessaires à l'affichage du nombre de comparaisons effectuées par le TRI PAR INSERTION du vecteur `mesPolars`

A2.7. **Testez -** NOMBRE DE COMPARAISONS ATTENDU : 9 579

Partie B – Comparaison recherche séquentielle et recherche dichotomique (vecteur trié)

B1. Classe Utilitaire – codage de fonctions de recherche

Dans la classe `Utilitaire` :

B1.1. **Ajoutez** et **codez** la procédure suivante (cf. cours 6, partie 2)

```
public static int rechSeqT(ArrayList<Polar> vPolar, String aut, int an) {
    // { vPolar trié selon l'ORDRE(annee, auteur) }
    // => { * résultat = indice dans vPolar du 1er élément de vPolar
    //                   écrit par aut, l'année an, si trouvé
    //      * résultat = -1, si aucun roman écrit par aut, l'année an
    //      LA RECHERCHE EST SÉQUENTIELLE }
```

INDICATION : Pour pouvoir utiliser la méthode `CompareTo` de la classe `Polar`, déclarez une variable de type `Polar`, qui a pour *auteur* `aut`, pour *annee* `an` et pour *titre*, une valeur quelconque (ex : "titre").

B1.2. **Ajoutez** et **codez** la procédure suivante (cf. cours 6, partie 3)

```
public static int rechDicho(ArrayList<Polar> vPolar, String aut, int an) {
    // { vPolar trié selon l'ORDRE(annee, auteur) }
    // => { * résultat = indice dans vPolar du 1er élément de vPolar
    //                   écrit par aut, l'année an, si trouvé
    //      * résultat = -1, si aucun roman écrit par aut, l'année an
    //      LA RECHERCHE EST DICHOTOMIQUE }
```

INDICATION : cf. B1.1

B2. Classe Comparaisons – test des fonctions codées

Dans la classe `Comparaisons` :

B2.1. Ajoutez les instructions de saisie d'un *auteur* et d'une *année*

B2.2. Ajoutez les instructions nécessaires pour tester les fonctions `rechSeqT` et `rechDicho`
Selon le résultat de ces fonction : l'indice du 1er élément du vecteur `MesPolars` *écrit par l'auteur saisi, l'année saisie sera affiché, ou l'échec de la recherche sera signifié à l'utilisateur.*

NOTE : les deux fonctions doivent renvoyer le même résultat pour l'auteur et l'année saisis

B2.3. **Testez** en vérifiant que vous obtenez les mêmes résultats que ceux proposés ci-dessous:

Auteur	Année	Indice renvoyé ou *non trouvé*
POUY	2000	113
POUY	2021	*non trouvé*
SYLVAIN	2007	169
ABRACADABRA	2000	*non trouvé*
ZORRO	2022	*non trouvé*
MANCHETTE	1976	76

B3. Classe PaireResultatCompteur

B3.1. Dans le projet TP7B, créez une classe `PaireResultatCompteur`

Dans la fenêtre d'édition de cette classe :

- sélectionnez les deux premières lignes (qui ont été automatiquement générées)
- remplacez-les par le code ci-dessous (cf. cours 07 - partie 3, page 18)

```java
/**
 * Classe générique pour outiller une fonction
 * @param <R> : R est le type du résultat de la fonction outillée
 */

public class PaireResultatCompteur<R> {

    private final R res;         // résultat constant de la fonction outillée
    private final int compteur; // compteur constant du code observé

    public PaireResultatCompteur(R res, int compteur) {
        this.res = res;
        this.compteur = compteur;
    }

    public R getRes () {
        return res;
    }
    public int getCompteur() {
        return compteur;
    }
}
```

B4. Classe Utilitaire – outillage des fonctions codées

Dans la classe `Utilitaire` :

B4.1. **Ajoutez** et **codez** la fonction suivante (cf. cours 7, partie 3, page 30)

```java
public static PaireResultatCompteur<Integer> rechSeqTComp(ArrayList<Polar> vPolar,
                                                          String aut, int an) {
    // { vPolar trié selon l'ORDRE(annee, auteur) } =>
    //   { * le premier élément de vPolar écrit par aut, l'année an a été cherché
    //       à l'aide d'un algorithme de RECHERCHE SÉQUENTIELLE
    //     * résultat = un objet de type PaireResCompteur dont :
    //       - l'attribut res a pour valeur l'indice dans vPolar du 1er élément d'auteur
    //         aut et d'année an, si trouvé / -1 si pas trouvé
    //       - l'attribut compteur a pour valeur le nombre de comparaisons effectuées entre
    //         un élément du vecteur et ce qui est cherché, pour produire la valeur de res }
```

B4.2. **Ajoutez** et **codez** la fonction suivante

```java
public static PaireResultatCompteur<Integer> rechDichoComp(ArrayList<Polar> vPolar,
                                                           String aut, int an) {
    // { vPolar trié selon l'ORDRE(annee, auteur) } =>
    //   { * le premier élément de vPolar écrit par aut, l'année an a été cherché
    //       à l'aide d'un algorithme de RECHERCHE DICHOTOMIQUE
    //     * résultat = un objet de type PaireResCompteur dont :
    //       - l'attribut res a pour valeur l'indice dans vPolar du 1er élément d'auteur
    //         aut et d'année an, si trouvé / -1 si pas trouvé
    //       - l'attribut compteur a pour valeur le nombre de comparaisons effectuées entre
    //         un élément du vecteur et ce qui est cherché, pour produire la valeur de res }
```

NOTE : Pour coder cette fonction, repérez les instructions qui effectuent les comparaisons à comptabiliser

B5. Classe Comparaisons – comparaison des fonctions de recherche

Dans la classe `Comparaisons` :

B5.1. Ajoutez la déclaration d'une variable de type `PaireResultatCompteur<Integer>`

B5.2. Mettez en commentaire les instructions écrites en réponse aux questions B**2.2** et B**2.3**

B5.3. Ajoutez les instructions nécessaires pour afficher le nombre de comparaisons, effectuées *respectivement* par les fonctions rechSeqTComp et `rechDichoComp`

B5.4. **Testez** en saisissant successivement les couples (Auteur, Année) du tableau présenté en B**2.4**

B5.5. Que remarquez-vous ?...