

# Enoncé

Rendez-vous sur <https://classroom.github.com/a/BkM6ruW9>, téléchargez votre projet et réalisez le travail demandé dans la section **Exercices**. Un fonctionnement correct ne peut pas vous assurer une note parfaite. La qualité de votre code, performance et lisibilité sont également pris en compte ! Pensez également à commenter votre code si vous le jugez complexe.

Pour finir, n'oubliez pas de *pusher* votre code sur votre *repository* git – vous avez jusqu'au 03 novembre 2019 à 23h59.

## Exercice

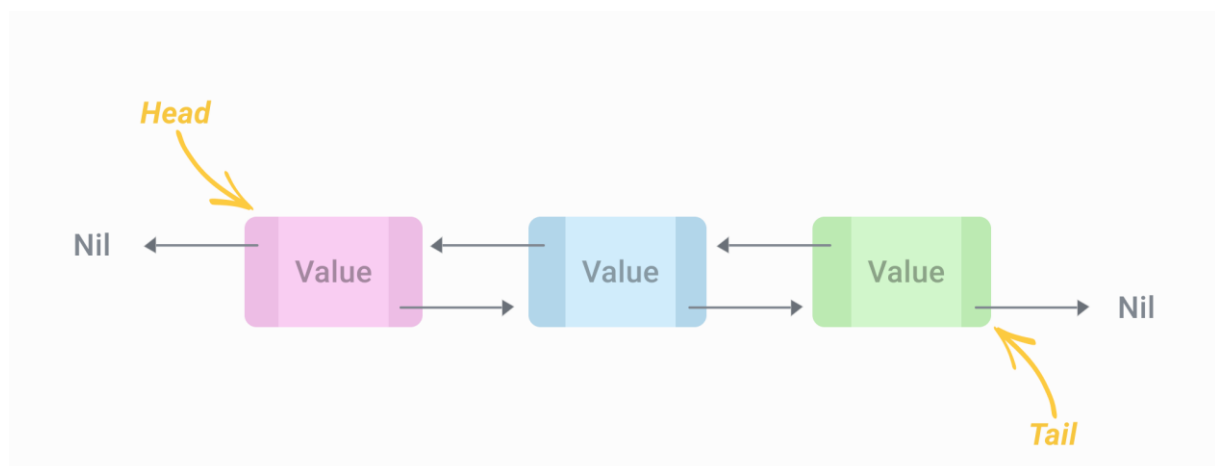
Dans cet exercice, vous avez 4 classes à coder, réparties dans 3 packages différents. Voici la structure finale de votre projet :

- collections
  - MyCollection (interface)
  - MyArrayList
  - MyLinkedList
- domaine
  - Headline
- outils
  - FileToStr

L'interface **MyCollection** contient toutes les méthodes que vos classes **MyArrayList** et **MyLinkedList** doivent implémenter, et elle vous est déjà fournie.

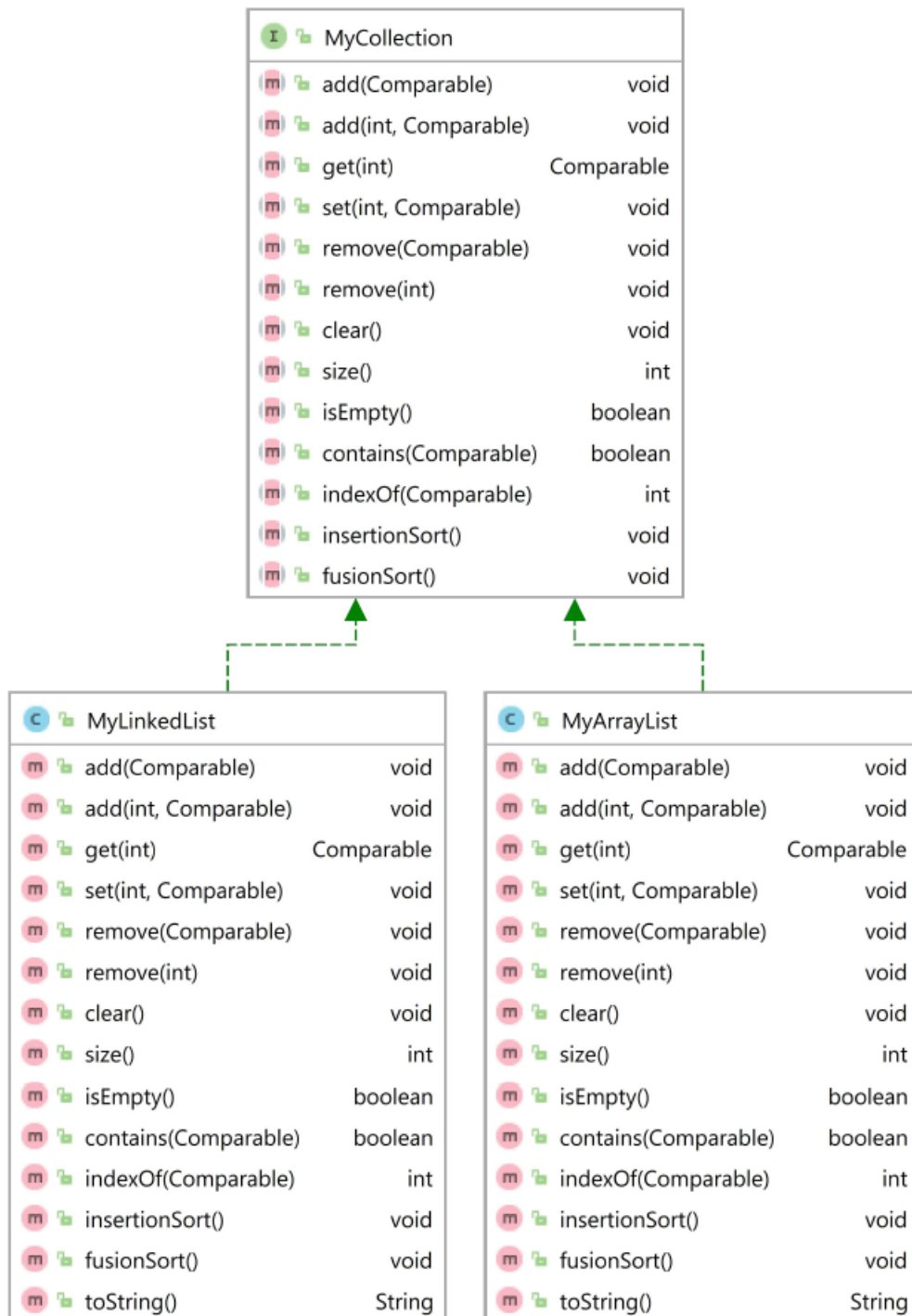
La classe **MyArrayList** doit contenir un tableau d'objets Comparables. Par défaut, ce tableau possède une taille 10 et celle-ci augmente de 50% si nécessaire.

La classe **MyLinkedList** est une liste doublement chaînée (Voir figure ci-dessous). Celle-ci doit posséder une référence vers la première valeur et une référence vers la dernière.



Pour finir, chaque structure de données doit posséder deux algorithmes de tri : tri par insertion et tri par fusion – pensez également à commenter votre code pour montrer que vous avez compris les algorithmes. L'utilisation des structures de données fournies dans les packages java, ou librairies externes, est strictement interdite; vous devez tout coder vous-mêmes ! Si vous avez un doute dans une fonction que vous souhaitez utiliser, consultez votre assistant.

Voici un modèle de classes représentant la structure du package « collections » :













La classe **FileToStr** contient les méthodes nécessaires au chargement du fichier *irishtimes-date-text.csv*. Pour utiliser ce fichier, vous devez d'abord le *dézipper*. Celui-ci contient environ 1.4 million de titres d'articles publiés dans les journaux irlandais depuis 1996. Il possède trois champs séparés par des virgules : `publish_date` – date de publication, `headline_category` – catégorie du titre et `headline_text` – titre.























Voici un exemple de quelques lignes du fichier :

```
publish_date,headline_category,headline_text
19960102,news,"UUP sees possibility of voting Major out"
19960102,news,"Pubs targeted as curbs on smoking are extended"
19960102,news,"Papers reveal secret links with O'Neill cabinet"
```

Et voici les méthodes que cette classe doit posséder :

		FileToStr	
		loadIntoDoublyLinkedList(String)	MyLinkedList
		loadIntoDoublyLinkedList(String, int)	MyLinkedList
		loadIntoArrayList(String)	MyArrayList
		loadIntoArrayList(String, int)	MyArrayList

Pour finir, la classe **Headline** doit contenir chaque ligne du fichier de texte mentionné ci-dessus. Elle doit contenir la date, le type et le titre de chaque publication. Ce dernier doit être stocké sans les guillemets. Lors d'un tri, la comparaison se fait d'abord par titre et ensuite par date.

		Headline	
		getDate()	Date
		setDate(Date)	void
		getType()	String
		setType(String)	void
		getHeadline()	String
		setHeadline(String)	void
		equals(Object)	boolean
		hashCode()	int
		compareTo(Object)	int
		toString()	String

Afin de savoir de ce que vous devez implémenter dans chaque méthode, veuillez consulter la javadoc qui est fournie à la racine de votre *repository* git :

```
./javadoc/index.html
```

Une fois toutes les méthodes correctement implémentées, comparez les différences de performance pour les ajouts, suppressions, modifications, accès et les différents tris. Attention, si vous appliquez un tri d'insertion à tout l'ensemble du jeu de données, cela risque de vous prendre trop de temps ! Sur un ordinateur de l'école, cela a pris plus de 10h pour trier un tableau. Par contre, si votre algorithme de tri par fusion est correct, cela ne devrait prendre que quelques secondes.