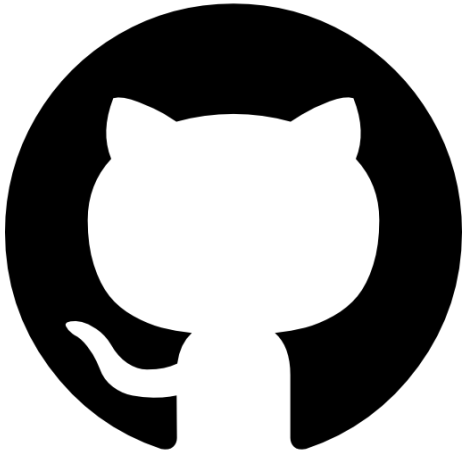
RAZAFINIRINA Mialisoa

Groupe 2A

S2.02 - Exploration algorithmique d'un problème

Comparaison et évaluation des solutions



Efficacité

Algorithmes	26efficacite.java	45efficacite.java	06efficacite.java	58efficacite.java	54efficacite.java	points
Passe le test initiale	18	18	18	18	—	18
Fonctionne mais ne passe pas le test	—	—	—	—	10	10
Non respect de l'anonymat	—	—	—	—	—	-1
Non respect de la consigne sur les méthodes de java.utit	—	—	—	—	—	-1 (hors concours)
Présence javadoc et commentaire	—	—	+0.25 (pas de javadoc et très peu de commentaire)	—	+0.5 (pas de javadoc mais le code est commenté)	+1 (s'il y a le javadoc et des commentaires)
Nom respect du : "public class Algo;"	—	-0.5	-0.5	-0.5	-0.5	-0.5
Nom respect du nom des méthodes	—	—	—	—	-0.5	-0.5
Ne traite pas les exceptions	—	-1	—	-1	-1	-1
Des méthodes non fait	—	—	—	—	-2	-1 par méthode
Compile / Ne compile pas	Compile	Compile	Compile	Compile	Ne compile pas	note diviser par 2 (hors concours)

Note finale	18	16.5	17.75	16.5	3.25	/20
Test codacy	A	A	B	C	B	A B C D (A : meilleur et D : pire)
Complexité obtenu avec codacy	5	7	7	5	5	
Complexité algorithmique	**Sans itération** RLE : O(n) unRLE : O(n) **Avec itération** RLE : O(n * iteration) unRLE : O(n * iteration)	**Sans itération** RLE : O(n) unRLE : O(n) **Avec itération** RLE : O(n * iteration) unRLE : O(n * iteration)	**Sans itération** RLE : O(n) unRLE : O(n) **Avec itération** RLE : O(n * iteration) unRLE : O(n * iteration)	**Sans itération** RLE : O(n) unRLE : O(n) **Avec itération** RLE : O(n * iteration) unRLE : O(n * iteration)	RLE : O(n) RLEIteration : O(n * iteration)	
Remarques	Les méthodes fonctionnent bien mais il manque les commentaires et javadoc.	Le programme devrait bien fonctionner, mais la gestion des exceptions a été oubliée. En ajoutant cette gestion, le programme réussit le test initial.	Le programme passe le test initial, mais la personne n'a pas respecté la déclaration "public class Algo". L'algorithme est efficace, mais il pourrait être amélioré en termes de robustesse et de lisibilité.	Le programme devrait fonctionner, mais la gestion des exceptions a été oubliée. En ajoutant cette gestion, le programme passe le test initial. Il serait également bénéfique d'ajouter des commentaires pour expliquer les parties critiques du code.	2 méthodes ne sont pas faites (unRLE et unRLE avec iteration). Le nom de la méthode RLE avec itération est incorrect. Les exceptions ne sont pas élevées et il a mis le main dans la classe Algo.	

Classements efficacité :

- 1 : 26efficacite.java
- 2 : 06efficacite.java
- 3 : 45efficacite.java
- 4 : 58efficacite.java
- 5 : 54efficacite.java

Simplicité

Algorithmes	45simplicite.java	61simplicite.py	19simplicite.java	30simplicite.py	63simplicite.java	points
Passe le test initiale (pour java) et passe nouveau test (pour python)	18	18	—-	18	18	18
Fonctionne mais ne passe pas le test	—	—	10	—	—-	10
Lisibilité du code	+0.5	+1	+0.5	+0.5	+1	+1 à -1
Présence de javadoc ou docstrings et de commentaires	+0.5 (présence des commentaires)	+0.5 (pas de docstrings mais il y a des commentaires)	—-	+0.5 (pas de docstrings mais il y a des commentaires)	+0.25 (pas de javadoc et très peu de commentaire)	+1
Nom respect du : “public class Algo;” (pour les classes .java)	-0.5	—	-0.5	—-	—-	-0.5

Non respect de l'anonymat	—-	—	—-	—-	—-	-1
Compile / Ne compile pas	Compile	Compile	Ne compile pas, car une seule fonction a été faite (RLE sans itération) et le nom de la fonction est incorrect.	Compile	Compile	note diviser par 2 si ne compile pas (hors concours)
Des méthodes non fait	—-	—	-3	—-	—-	-1 par méthode non fait
Note finale	18.5	19.5	3.5	19	19.25	/20
Test Codacy	C	B	B	B	A	(ABCD)
Remarques	Le code est lisible et bien commenté. Toutefois, le choix des noms de variables pourrait être amélioré.	Le code est bien commenté et fonctionne bien.	Algorithme inachevé. Seul le RLE est fait.	Le code est lisible et bien commenté.	Algorithme très clair, mais il manque le javadoc et très peu de commentaires.	

Classements simplicité :

- 1 : 61simplicite.py
- 2 : 63simplicite.java
- 3 : 30simplicite.py
- 4 : 45simplicite.java
- 5 : 19simplicite.java

Sobriété

Algorithmes (Java)	20sobriete.java	53sobriete.java	58sobriete.java	Points
Passe le test initiale pour les classes .java	18	—-	18	18
Fonctionne mais ne passe pas le test	—-	10	—-	10
Non respect de l'anonymat	—-	—-	—-	-1
Présence javadoc et commentaire	+0.5 (il y a les javadoc mais pas de commentaire)	+0.5 (pas de javadoc mais il y a des commentaire)	—-	+1
Nom respect du : "public class Algo;"	-1	-1	-1	-1
Nom respect du nom des méthodes	—-	-1 (nom de méthodes incorrect et trop de paramètre en entré pour la méthodes RLE_recursif())	—-	-1
Ne traite pas les exceptions	—-	-1	-1	-1
Des méthodes non fait	—-	—-	—-	-1 par méthodes
Compile / Ne compile pas	Compile	Ne compile pas	Compile	note diviser par 2 (hors concours)
Note finale	17.5	3.75 +1 = 4.75 (+1 car la personne a ajouté une partie optionnelle regroupant unRLE,	16	

		RLE recursif et unRLE recursif)		
Taille du fichier	31,6 Ko (32 364 octets)	3,97 Ko (4 073 octets)	1,65 Ko (1 690 octets)	
Test codacy	C	B	B	A B C D (A : meilleur et D : pire)
Remarques	Le code fonctionne correctement. C'est le fichier qui a la taille la plus élevée.	Le code fonctionne, mais il manque de lisibilité. De plus, pour le RLE récursif, deux paramètres supplémentaires ont été ajoutés par rapport à la version initiale.	Le code fonctionne mais il manque les exceptions.	

Classements sobriété :

- 1 : 58sobriete.java
- 2 : 20sobriete.java
- 3 : 53sobriete.java