Alexis Declippel 2B

SAE 2.02

Exploration algorithmique d’un problème

Sommaire

Alexis Declippel 2B

Introduction

Dans cette SAE, le but était dans la phase 1 de créer des algorithmes de tri avec un ordre. Il y avait 3 catégories d'algorithme : Simplicité, Efficacité et Sobriété.

Un algorithme simple doit avoir un code lisible, simple à comprendre qui possède une bonne Javadoc et avec des noms de variables cohérent pour qu'une personne débutant dans le code puisse le comprendre aisément. Un algorithme efficace doit avoir un code qui s'effectue de manière rapide et qui ne fait pas de calcul inutile. Un algorithme sobre doit avoir un code qui est léger à l'exécution il ne doit pas faire de calcul inutile qui ajouterai du poids.

Dans chaque catégorie on pouvait faire un algorithme comme étant le meilleur ou le pire. Ils peuvent avoir comme objectifs d'être le pire. Dans ce cas un code simple doit devenir un code compliqué, un code efficace doit devenir un code long à exécuter et un code sobre doit être un code lourd à exécuter.

Dans la phase 2, le but était d'évaluer les algorithmes. Chaque groupe possède des critères d'évaluation propre à eux et le but est de les évaluer par catégories par types (meilleur ou pire). Dans le barème de notation certain code sont sujet à des malus si il ne le respecte pas. À la fin on obtient un classement par catégories et type.

Alexis Declippel 2B

Simplicité

Critère de notation

Les critères de notation des algorithmes simple sont sa lisibilité, la cohérence du nom des variables, la Javadoc, l'utilisation de sous-programmes et la qualité du code.

La lisibilité du code permet à la personne qui le lis de pouvoir le comprendre de manière clair et concis. Il faut pour cela un code aéré par des sauts de ligne, une bonne indentation, etc.

L'utilisation de sous-programmes permet d'avoir un code digeste à lire et plus expliqué.

La qualité du code est le respect des normes de codage. Il est calculé par le Codacy, un site qui donne une note allant de A à F au code qui sont donnés. (A étant 5/5 et F étant 0/5).

Pour les algorithmes ayant pour but d'être les pire tout les critères sont inversés à la place on veut une mauvaise lisibilité, aucune cohérence dans le nom des variables, pas de Javadoc, un code dense sans sous-programme et une mauvaise qualité de code. Pour la qualité de code F donne 5/5 et A donne 0/5.

Alexis Declippel 2B

Simplicité Meilleur

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Simplicité Meilleur 12

Lisibilité du code /4	4
Noms des variables cohérents /4	3
Javadoc /4	0
Utilisation de sous-programme(s) /3	0
Qualité du code /5	5

Cet algorithme ne passait pas les tests qu'on a écrit. Il ne pouvait pas une note plus élevé que 18/20. Le code reste lisible avec une bonne cohérence dans ces variables et bien évalué par Codacy avec un A. En revanche il n'utilisait aucun sous-programme et ne possédait pas de Javadoc.

La note finale est **12/20**.

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Simplicité Meilleur 19

Lisibilité du code /4	4
Noms des variables cohérents /4	4
Javadoc /4	2
Utilisation de sous-programme(s) /3	3
Qualité du code /5	4

Cet algorithme ne passait pas les tests fourni initialement donc la note est ramené à 10/20 il avait aussi une mauvaise nomenclature ce qui lui fait perdre un point. Le code est lisible avec une bonne cohérence des variables. Il possède une Javadoc présente mais incomplète et plusieurs sous-programmes et une qualité de code à B sur Codacy.

La note finale est **10/20**. Il aurait pu avoir 16/20.

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Simplicité Meilleur 22

Lisibilité du code /4	4
Noms des variables cohérents /4	3
Javadoc /4	0
Utilisation de sous-programme(s) /3	0
Qualité du code /5	3

Cet algorithme ne passait pas les tests fourni initialement donc la note est ramené à 10/20 il avait aussi une mauvaise nomenclature ce qui lui fait perdre un point. Le code reste lisible avec une bonne cohérence dans ces variables et une évaluation moyenne par Codacy avec un C. En revanche il n'utilisait aucun sous-programme et ne possédait pas de Javadoc.

La note finale est **9/20**.

Classement :

Rang	Numéro de l'algorithme
1)	12
2)	17
3)	22

Alexis Declippel 2B

Simplicité Pire

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Simplicité Pire 7

Code illisible /4	0
Nom des variables non compréhensibles /4	0
Javadoc mal ou pas construite /4	4
Densité du code /3	2
Qualité du code /5	0

Cet algorithme ne passe pas les tests initiaux et possède une mauvaise nomenclature. Il est censé être un des pires est plutôt bon. Il possède un code lisible, de bon noms de variable et une bonne qualité de code. Ce qui lui rajoute des points sont une Javadoc absente et une grosse densité où il y a plusieurs boucle d'affilé.

La note finale est **5/20**.

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Simplicité Pire 17

Code illisible /4	0
Nom des variables non compréhensibles /4	4
Javadoc mal ou pas construite /4	4
Densité du code /3	0
Qualité du code /5	2

Ce programme n'a aucun malus sur le barème et c'est le seul. Il possède un code lisible un bonne densité avec des sauts de lignes et une bonne indentation, il possède aussi une assez bonne qualité de code avec Codacy et ce n'est pas ce qui est recherché. Il possède aucune Javadoc et des noms de variable incompréhensible.

La note finale est **10/20**.

Classement :

Rang	Numéro de l'algorithme
1)	17
2)	7

Alexis Declippel 2B

Sobriété

Les critères de notation des algorithmes sobre sont la sobriété numérique et la qualité du code.

La sobriété numérique est le fait de consommer le moins de ressources possible. La consommation en ressource d'un algorithme peut être calculée par ideone.com, un site donnant plusieurs informations sur l'exécution d'un programme.

La qualité du code est le respect des normes de codage. Il est calculé par le Codacy, un site qui donne une note allant de A à F au code qui sont donnés. (A étant 5/5 et F étant 0/5).

Pour les algorithmes visant la pire sobriété, ces critères sont inversés, on attend donc un algorithme consommant beaucoup de ressources. Pour la qualité du code, un F sur Codacy équivaut à un 5/5 et un A à un 0/5.

Alexis Declippel 2B

Sobriété Meilleur

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Sobriété Meilleur 3

Sobriété numérique /15	10
Qualité du code /5	4

Le programme ne passe pas les tests initiaux et possède une mauvaise nomenclature sa note est ramené à 10/20. Il possède une assez bonne sobriété numérique et une bonne qualité de code.

La note finale est **10/20**. Il aurait pu avoir 13/20.

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Sobriété Meilleur 8

Sobriété numérique /15	6
Qualité du code /5	5

Le programme ne passe pas les tests initiaux et possède une mauvaise nomenclature sa note est ramené à 10/20. Il possède une sobriété numérique moyenne et une très bonne qualité de code.

La note finale est **10/20**

Classement :

Rang	Numéro de l'algorithme
1)	3
1)	8

Alexis Declippel 2B

Sobriété Pire

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Sobriété Pire 20

Sobriété numérique /15	2
Qualité du code /5	3

Le programme ne passe pas les tests initiaux et possède une mauvaise nomenclature sa note est ramené à 10/20. Il possède une bonne sobriété numérique et une qualité de code moyenne.

La note finale est **4/20**.

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

2- Sobriété Pire 58

Sobriété numérique /15	15
Qualité du code /5	1

Le programme ne passe pas les tests initiaux et possède une mauvaise nomenclature sa note est ramené à 10/20. Il possède une très mauvaise sobriété numérique ce qui est positif dans sa catégorie mais il possède une bonne qualité de code.

La note finale est **10/20**. Il aurait pu avoir 15/20.

Classement :

Rang	Numéro de l'algorithme
1)	58
2)	20

Alexis Declippel 2B

Efficacité

Les critères de notation des algorithmes simple sont le temps d'exécution, l'efficacité et la non-utlisation de sous-programme.

Le temps d'exécution est calculé par le site ideone.com et la note est donné en fonction. C'est évidemment le critère le plus important pour cette catégorie.

L'efficacité est l'évaluation de la complexité algorithmique.

Il est important qu'il n'y ai pas de sous-programme car cela enlèverait beaucoup d'efficacité au programme.

Pour les algorithmes visant la pire efficacité, ces critères sont inversés, on attend donc un algorithme étant lent, avec une grosse complexité algorithmique ainsi que l'utilisation de sous-programme(s).

Alexis Declippel 2B

Efficacité Meilleur

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Efficacité Meilleur 33

Temps d'exécution /12	4
Efficacité /5	3
Non utilisation de sous-programme /3	2

Le programme ne passe pas les tests initiaux sa note est ramené à 10/20. Il possède un efficacité moyenne, un mauvais temps d'exécution et utilise un seul sous-programme.

La note finale est 9/20

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

2- Efficacité Meilleur 66

Temps d'exécution /12	11
Efficacité /5	2
Non utilisation de sous-programme /3	3

Le programme ne passe pas les tests la note est ramené à 10/20 et possède une mauvaise nomenclature. Il n'utilise aucun sous-programme et a un très bon temps d'exécution mais il n'a pas une bonne efficacité.

La note finale est **10/20**. il aurait pu avoir 15/20.

Classement :

Rang	Numéro de l'algorithme
1)	66
2)	33

Alexis Declippel 2B

Efficacité Pire

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

Efficacité Pire 48

Temps d'exécution /12	2
Efficacité /5	4
Utilisation de sous-programme /3	0

Dans ce programme, il manquait des imports sans ça le code ne compile pas. Après l'ajout des imports le code ne fonctionne pas car il retour seulement " [ ] ". J'ai quand même pu évaluer ce programme il n'a pas une bonne efficacité mais a un bon temps d'exécution et pas de sous programme.

La note finale est **2.5/20**. Sans tout les problèmes il aurait pu avoir 5/20

Alexis Declippel 2B

Problème	Sanction	
Ne compile pas	Note finale divisée par 2	Et hors concours
Ne correspond pas à une classe Exercice	Note finale divisée par 2	
Non respect de la nomenclature précise	-1	
Non respect de l'anonymat	-1	
Non respect de la consigne sur les méthodes de java.util (pour efficacité)	-1	Et hors concours
Passe tous les tests fournis initialement	18	
Passe vos tests supplémentaires plus complets	20	
Fonctionne mais ne passe pas les tests fournis initialement	10	
Ne fonctionne pas (retour erroné ou pas du bon type)	5	

: problème rencontré

2- Efficacité Pire 58

Temps d'exécution /12	8
Efficacité /5	5
Utilisation de sous-programme /3	0

Le programme ne passe pas les tests. Il a un long temps d'exécution et une mauvaise efficacité. En revanche il n'a pas de sous-programme ce qui aurait pu rendre le programme encore moins efficace.

La note finale est **10/20**. il aurait pu avoir 13/20

Classement :

Rang	Numéro de l'algorithme
1)	58
2)	48