=

Une image contenant texte, Police, logo, Graphique

Description générée automatiquement

Algorithmie(niv2)

Évaluation formative

**Nom : HUTT**

**Prénom : David**

**Classe : ABC Dev-119**

**Jalons n°3 : Algorithmie (niv 2)**

**Durée : Toute la journée**

**Règles du jalon :**

Internet, Cours, documents, IA autorisée, me renvoyer les questions et les réponses avec votre nom au format numériquement. Les codes devront être envoyés sur le GIT.

**Bon courage à vous**

# Algorithmie

## Répondez aux questions suivantes :

**Répondez à ces questions, j’attends des argumentations de votre part, vous pouvez utilisez des exemples ou même un pseudo code :**

**Question 1 : Pouvez-vous m’expliquer comment je peux faire un compteur avec une boucle qui compte de 1 à 10 : ?**

**Il y a 3 méthodes pour faire un compteur avec une boucle :**

1. Avec une boucle for
2. Avec une boucle while
3. Avec une boucle do-while

**Boucle for qui compte de 1 à 10 :**

**for (int i = 1; i <= 10; i++) {**

**System.out.println(i);**

**Explications :**

1. Initialisation : On commence à 1 (int i = 1)
2. Condition : On continue tant que i est inférieur ou égal à 10 (i <= 10)
3. Incrémentation : À chaque itération, on augmente i de 1 (i++)

**La boucle for est généralement la plus adaptée**

**Question 2 : Pouvez-vous m’expliquer comment faire afficher la date et l’heure d’aujourd’hui sur Java?**

**1. Avec java.time :**

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class….. {

public static void main(String[] args) {

// 1. Obtenir la date et l'heure actuelles

LocalDateTime maintenant = LocalDateTime.now();

// 2. Définir un format (ex: "dd/MM/yyyy HH:mm:ss")

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

// 3. Formater la date

String dateFormatee = maintenant.format(formatter);

// 4. Afficher le résultat

System.out.println("Date et heure actuelles : " + dateFormatee);

}

}

**Explications étape par étape :**

1. OBTENIR\_DATE\_HEURE\_ACTUELLE()
   * Récupère la date et l'heure du système (ex: 10/06/2024 15:30:45).
   * En Java, cela peut être fait avec LocalDateTime.now() ou new Date().
2. FORMAT ← "JJ/MM/AAAA HH:MM:SS"
   * Définit comment afficher la date.
3. FORMATER (dateHeureActuelle, format)
   * Convertit la date brute en texte selon le format choisi.
   * En Java, on utilise DateTimeFormatter (moderne) ou SimpleDateFormat (ancien).
4. AFFICHER(...)
   * Affiche le résultat final

**Options de formatage :**

Vous pouvez personnaliser le format en modifiant le motif dans DateTimeFormatter ou SimpleDateFormat :

* "dd/MM/yyyy" → 10/06/2023
* "MM/dd/yyyy" → 06/10/2023
* "yyyy-MM-dd" → 2023-06-10
* "HH:mm:ss" → 14:30:45 (format 24h)
* "hh:mm:ss a" → 02:30:45 PM (format 12h)
* "EEEE, dd MMMM yyyy" → Samedi, 10 juin 2023

**Question 3 : Comment fait-on pour attraper un message d’erreur après exécution du code ?**

**En Java, la gestion des erreurs (ou exceptions) est un mécanisme essentiel pour traiter les situations anormales qui peuvent survenir pendant l'exécution d'un programme.**

**1. Structure de base : try-catch**

**Syntaxe :**

try {

// Code susceptible de générer une erreur

} catch (TypeException e) {

// Gestion de l'erreur

}

**Exemple : Division par zéro**

try {

int result = 10 / 0; // Lève une ArithmeticException

} catch (ArithmeticException e) {

System.err.println("Erreur : Division par zéro !");

}

* try : Bloc où l'on place le code susceptible de lever une exception.
* catch : Bloc exécuté si une exception du type spécifié est levée.
* e.getMessage() : Affiche le message d'erreur associé à l'exception.

**2. Hiérarchie des exceptions**

**En Java, les exceptions sont classées en deux catégories :**

* RuntimeException (unchecked) : Erreurs logiques (ex: NullPointerException, ArrayIndexOutOfBoundsException).  
  → Pas obligatoire de les attraper (mais recommandé).
* Autres exceptions (checked) : Doivent être gérées obligatoirement (ex: IOException, SQLException).

Exemple avec IOException (checked)

try {

FileReader file = new FileReader("fichier.txt"); // Peut lever FileNotFoundException

} catch (IOException e) {

System.err.println("Erreur de fichier : " + e.getMessage());

}

**3. Attraper plusieurs exceptions**

**On peut avoir plusieurs blocs catch pour gérer différents types d'erreurs.**

**Exemple :**

try {

int[] arr = new int[3];

arr[5] = 10; // ArrayIndexOutOfBoundsException

String s = null;

s.length(); // NullPointerException

} catch (ArrayIndexOutOfBoundsException e) {

System.err.println("Index hors limites !");

} catch (NullPointerException e) {

System.err.println("Objet null !");

} catch (Exception e) { // Exception générale (si aucune autre n'est attrapée)

System.err.println("Erreur inattendue : " + e.getMessage());

}

Ordre important : Toujours aller du plus spécifique au plus général (Exception en dernier).

**4. Bloc finally**

* **Toujours exécuté, qu'une exception soit levée ou non.**
* **Utile pour fermer des ressources (fichiers, connexions BDD, etc.).**

**Exemple :**

FileInputStream file = null;

try {

file = new FileInputStream("fichier.txt");

// Traitement du fichier...

} catch (IOException e) {

System.err.println("Erreur IO : " + e.getMessage());

} finally {

if (file != null) {

try {

file.close(); // Fermeture garantie

} catch (IOException e) {

System.err.println("Erreur lors de la fermeture : " + e.getMessage());

}

}

}

**5. Récupérer la stack trace complète :**

Pour déboguer, on peut afficher la pile d'appels complète :

try {

int[] arr = {1, 2, 3};

System.out.println(arr[5]); // ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

e.printStackTrace(); // Affiche la stack trace dans System.err

}

Résultat :

text

Copy

Download

java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3

at Main.main(Main.java:10)

**6. Lever une exception personnalisée**

**On peut créer ses propres exceptions en étendant Exception.**

**Exemple :**

class MonException extends Exception {

public MonException(String message) {

super(message);

}

}

// Utilisation :

try {

throw new MonException("Erreur personnalisée !");

} catch (MonException e) {

System.err.println(e.getMessage());

}

|  |  |
| --- | --- |
| ALGORITHMIE |  |

## **Cas pratique :**

**Une fois terminée, veuillez pusher votre Jalon sur Github (avec votre lien) ou me l’envoyer par Teams. Vous devrez faire en sorte que votre code JAVA respect bien la demande du client, veuillez le tester sur votre terminal, je ne veux aucun message d’erreur ! VEUILLEZ LIRE TRES ATTENTIVEMENT !!!**

**Vous êtes l’agence « Les Avengers du Dev », vous avez un client qui aimerait avoir pour son site de réservation d’avion, un outil lui permettant de lister tous ses vols avec les informations demandées. Voici sa demande pour plus d’informations**

**Demande du client :**

**Bonjour les Avengers,**

**Nous sommes la compagnie aérienne AIRMESS, et nous aimerions disposer d’un outil qui nous permette de lister nous-mêmes nos propres vols avec les informations suivantes :**

* **Départ :** Ville / Pays
* **Arrivée :** Ville / Pays
* **Date et heure de départ :** heures et minutes. Si la date de départ du vol est très proche d’aujourd’hui (moins d’une semaine), le prix devra automatiquement augmenter de 40 %. En revanche, si la date du vol est très éloignée (plus de 6 mois avant la date de départ), le prix devra diminuer de 40 %.
* **Durée du vol :** heures et minutes
* **Date et heure d’arrivée :** calculées automatiquement à partir de la durée du vol.
* **Nombre de places** : minimum 80 places, maximum 200 places. Si le nombre de places est égal ou supérieur à 150, une réduction de 10 % sera appliquée sur le prix initial. Si le nombre de places est inférieur à 100, le prix augmentera de 10 %.
* **Prix initial du vol :** défini à la création du vol et ajusté automatiquement en fonction de la date de départ et du nombre de places, comme décrit précédemment.

**Une fois les informations saisies, j’aimerais que tous les vols soient affichés. Nous devrions pouvoir créer autant de vols que nécessaire.**

**Enfin, nous aimerions surtout avoir un système d’authentification pour accéder à cette application. Vous nous fournirez le login et le mot de passe.**

**Login : hutt**

**Mdp : david**

Une image contenant texte, Visage humain, dessin humoristique, affiche

Le contenu généré par l’IA peut être incorrect.