Mémo

1\. Déclaration de classe
public class NomDeLaClasse { ... }
Définit une classe.

2\. Déclaration de package
package fr.raphaelmakaryan.lombredesdragons.Game;
Indique le package du fichier.

3\. Import de classes
import fr.raphaelmakaryan.lombredesdragons.Game.Game;
Permet d'utiliser des classes d'autres packages.

4\. Déclaration de méthode
public void nomDeLaMethode() { ... }
Définit une méthode.

5\. Appel de méthode
objet.nomDeLaMethode();
Appelle une méthode sur un objet.

6\. Création d'objet
NomDeLaClasse obj = new NomDeLaClasse();
Instancie une classe.

7\. Boucle for
for (int i = 0; i < n; i++) { ... }
Boucle avec compteur.

8\. Boucle while
while (condition) { ... }
Boucle tant que la condition est vraie.

9\. Condition if/else
if (condition) { ... } else { ... }
Exécute un bloc selon la condition.

10\. Affichage console
System.out.println("Texte");
Affiche du texte dans la console.

11\. Tableau
Type[] tab = new Type[taille];
Déclare un tableau.

12\. Enumération
public enum NomEnum { ... }
Définit un type énuméré.

13\. Héritage
public class Warrior extends Character { ... }
Une classe hérite d'une autre.

14\. Interface
public interface NomInterface { ... }
Déclare une interface.

15\. Implémentation d'interface
public class NomClasse implements NomInterface { ... }
Implémente une interface.

16\. Gestion des exceptions
try { ... } catch (Exception e) { ... }
Gère les erreurs.

17\. Attributs de classe
private int vie;
Déclare une variable membre.

18\. Constructeur
public NomDeLaClasse() { ... }
Méthode spéciale pour créer un objet.

19\. Surcharge de méthode
public void attaque() { ... }
public void attaque(int force) { ... }
Plusieurs méthodes avec le même nom mais des paramètres différents.

20\. Collections
List<Type> liste = new ArrayList<>();
Utilise des listes dynamiques.

21\. Réflexion
Method method = obj.getClass().getMethod("nomMethode");
method.invoke(obj);
Appelle une méthode par son nom.

22\. Accesseurs (getters/setters)
public int getVie() { return vie; }
public void setVie(int vie) { this.vie = vie; }
Méthodes pour accéder/modifier les attributs.

23\. Static
public static void main(String[] args) { ... }
Méthode ou attribut partagé par toutes les instances.

24\. Final
public final class NomDeLaClasse { ... }
Empêche l'héritage ou la modification.

25\. Super
super.nomDeLaMethode();
Appelle une méthode de la classe parente.

26\. Override
@Override
public void nomDeLaMethode() { ... }
Redéfinit une méthode héritée.

27\. Scanner (entrée utilisateur)
Scanner sc = new Scanner(System.in);
Lit l'entrée clavier.

28\. Switch
switch (choix) {
case 1: ...; break;
case 2: ...; break;
default: ...;
}
Choix multiples selon une variable.

29\. Opérateurs logiques
&&, ||, !
Utilisés dans les conditions.

30\. Opérateurs arithmétiques
+, -, *, /, %
Calculs mathématiques.

31\. ToString
public String toString() { ... }

Affichage personnalisé d’un objet.

32\. Equals
public boolean equals(Object obj) { ... }
Comparaison d’objets.

33\. Exception
Classe de base pour la gestion des erreurs et événements exceptionnels. Permet d’interrompre le déroulement normal du programme et de traiter les erreurs de façon contrôlée.

34\. throw
Mot-clé utilisé pour lancer une exception explicitement dans le code.

35\. throws
Mot-clé utilisé dans la déclaration d’une méthode pour indiquer qu’elle peut lancer une ou plusieurs exceptions.

36\. Exception personnalisée
Classe qui hérite de Exception et permet de définir des erreurs spécifiques à l’application (ex: OutOfBoardException).
try {
// code susceptible de provoquer une exception
} catch (OutOfBoardException e) {
// traitement de l’exception
}

37\. Connetion à une base de données
Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nom_db",
"utilisateur", "mot_de_passe");
Permet de se connecter à une base de données.

38\. Statement
Statement stmt = conn.createStatement();
Permet d’exécuter des requêtes SQL.

39\. ResultSet
ResultSet rs = stmt.executeQuery("SELECT * FROM table");
Permet de récupérer les résultats d’une requête SQL.

40\. PreparedStatement
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO table

(colonne) VALUES (?)");
Permet d’exécuter des requêtes SQL avec des paramètres.

41\. ResultSetMetaData
ResultSetMetaData rsmd = rs.getMetaData();
Permet d’obtenir des informations sur les colonnes d’un ResultSet.

42\. ResultSet.next()
rs.next();
Permet de parcourir les lignes d’un ResultSet.

43\. Statement RETURN GENERATED KEYS
Statement stmt = conn.createStatement();
stmt.executeUpdate("INSERT INTO table (colonne) VALUES ('valeur')",
Statement.RETURN_GENERATED_KEYS);
Permet de récupérer les clés générées par une requête d’insertion.

44\. ResultSet.getInt("colonne")
int valeur = rs.getInt("colonne");
Permet de récupérer la valeur d’une colonne d’un ResultSet.

45\. ResultSet.getString("colonne")
String valeur = rs.getString("colonne");
Permet de récupérer la valeur d’une colonne d’un ResultSet sous forme de chaîne de caractères.

46\. Execution de requêtes SQL
stmt.executeUpdate("UPDATE table SET colonne = 'valeur' WHERE condition");
Permet d’exécuter des requêtes de mise à jour, d’insertion ou de suppression.