Classes Abstraites & Interfaces: Deviens un Maître du Code Flexible!

Aujourd'hui, on va...

L'Abstraction: C'est quoi ce truc? (abstract)

Interfaces: Les contrats du code!

Plusieurs Interfaces: Collectionne-les tous!

Personnage, Affichable, Comparable: Un trio de choc!

L'Abstraction: Le Mystère Dévoilé!

- Classe abstraite: Un plan incomplet, une base de construction
  - Tu ne peux pas créer un objet directement à partir d'une classe abstraite
- Mot clé: `abstract`

Abstraction: Exemple de Code

- public abstract class Personnage implements Serializable { ... } //Serializable? On verra ça plus tard!
- public abstract void attaquer(Personnage cible); //Une méthode que chaque personnage devra implémenter à sa façon
- Personnage est le modèle de base, mais on ne créera jamais de Personnage 'générique': On veut des Joueurs ou des Monstres!

Interfaces: Les Contrats du Code

- Interface: Une liste de promesses, un contrat à respecter
- Une classe qui implémente une interface s'engage à fournir le code pour toutes les méthodes de l'interface
  - Comme dire: 'Si tu veux être Affichable, tu dois absolument avoir une méthode afficher()!'
- Depuis Java 8, les interfaces peuvent avoir des méthodes `default` (avec code par défaut) et `static` (utilitaires)

Interfaces: Exemple en Code

- public interface Affichable {
- void afficher(); //La promesse d'afficher quelque chose
- }
- public class Joueur extends Personnage implements Affichable { ... } //Joueur signe le contrat Affichable

Plusieurs Interfaces: Collectionne-les Tous!

- Une classe peut implémenter PLUSIEURS interfaces: Deviens un expert multi-tâches!
  - Comme dire: 'Je suis à la fois Affichable (je sais m'afficher) et Comparable (je sais me comparer aux autres)'

Plusieurs Interfaces: Exemple de Code

- public class Joueur extends Personnage implements Affichable, Comparable<Personnage> {
- @Override public void afficher() { ... } //J'implémente Affichable
- @Override public int compareTo(Personnage other) { return this.vieCourante - other.vieCourante; } //J'implémente Comparable
- }
- Comparable<T>: Pour pouvoir ranger les objets (du plus petit au plus grand, par exemple)

À toi de jouer! - Description Abstraite

- Dans Personnage, ajoute une méthode abstraite:
- public abstract String description();
- Fais en sorte que Joueur et Monstre donnent une description sympa

À toi de jouer! - Deviens un Guérisseur!

- Crée une interface `Healable`:
- public interface Healable { void soigner(int montant); }
- Fais implémenter Healable par Joueur et Monstre
- La méthode soigner doit augmenter la vie (sans dépasser le max!)

À toi de jouer! - Trie tes Personnages!

- Fais en sorte que Personnage implémente Comparable<Personnage>
- Utilise la vieCourante pour comparer les personnages
- Crée une liste de Personnages et trie-la! (Collections.sort(maListe))

À toi de jouer! - Default Power!

- Ajoute une méthode `default void afficherNom()` dans l'interface `Affichable`
- Cette méthode doit afficher seulement le nom du personnage
- Teste ça dans la classe Joueur: Facile non?

À toi de jouer! - Le Choix Cornélien

- Pourquoi une classe abstraite pour Personnage et pas juste une interface?
- Dans quels cas on préfère une interface à une classe abstraite?
- Justifie tes réponses!

Besoin d'aide? Ressources Utiles

- Abstraction et classes abstraites (Oracle) : https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html
- Interfaces Java (Oracle) : https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html
- Default methods (Java 8) : https://www.baeldung.com/java-8-default-methods