Objets Java: Connecte-les et Crée des Applications Géniales!

Ta Mission (Si Tu l'Acceptes):

Associations: Connecte tes objets entre eux!

Paramètres Objets: Envoie des messages à tes objets!

Retour d'Objets: Récupère les résultats de tes actions!

Diagrammes UML: Visualise tes créations!

L'Inventaire: Un exemple concret pour tout comprendre

Associations: Connecte Tes Mondes!

- Association: Une relation entre deux classes (objets)
  - Comme un Joueur qui possède un Inventaire, ou un Inventaire qui contient des Items
- Deux types: Composition (forte) et Agrégation (faible)

Composition: Fort Comme un Roc!

- Composition: Un objet contient un autre objet et est responsable de sa vie
- Si le premier objet disparaît, le deuxième aussi
- Dans l'Inventaire:
  - private final List<Item> objets; //La liste des objets n'existe que dans l'inventaire, et disparait avec lui

Agrégation: Une Liaison Plus Douce

- Agrégation: Un objet utilise un autre objet, mais ne le possède pas complètement
- Les objets peuvent exister indépendamment
- public boolean equiperObjet(Joueur joueur, Item item) //L'inventaire utilise un Joueur et un Item, mais ne les crée pas, et ne les détruit pas.

Paramètres: Envoie des Instructions!

- Paramètres: Des infos que tu envoies à une méthode pour qu'elle fasse son travail
- public boolean equiperObjet(Joueur joueur, Item item) //joueur et item sont les paramètres de la méthode
- Le Joueur et l'Item sont passés par référence: Si la méthode les modifie, les objets originaux seront modifiés!

Retour: Récupère le Butin!

- Une méthode peut te renvoyer un objet: C'est le résultat de son travail!
- public Item chercherEquipementParId(int id) { ... } //Cette méthode va te renvoyer l'item trouvé
- N'oublie pas de gérer le cas où l'objet n'existe pas (renvoyer null)

Diagrammes UML: Visualise Ton Univers!

- Diagramme UML: Un plan pour voir comment les classes sont connectées
- Classe: Un rectangle avec le nom, les attributs et les méthodes
- Association: Une ligne entre les classes (avec des flèches pour indiquer le sens)
- Multiplicité: Indique combien d'objets sont liés (1, 0..*, 1..*)

Diagrammes UML: Exemple

- Inventaire 1 <---> * Item
  - //Un inventaire contient une liste d'items
- Joueur 1 ---> * Inventaire
  - //Un joueur possède un inventaire
  - Les flèches indiquent qui connaît l'existence de qui

À toi de jouer! - Détective des Associations

- Dans la classe Inventaire, repère:
- Les relations de composition (quand un objet contient et gère un autre)
- Les relations d'agrégation (quand un objet utilise un autre, sans le posséder)
- Justifie tes réponses! Pourquoi c'est l'un ou l'autre?

À toi de jouer! - Le Voyage de la Potion

- On a ce code:
- Item potion = new Item(...); boolean ok = inv.utiliserObjet(joueur, potion.getId());
- Quel objet est passé en paramètre? Par valeur ou par référence?
- Que se passe-t-il avec la potion si le joueur l'utilise? (elle disparaît?)

À toi de jouer! - Recherche et Affichage

- La méthode chercherObjetParId(int id):
- Que se passe-t-il si elle trouve l'objet avec l'id?
- Que se passe-t-il si elle ne trouve rien du tout? (elle renvoie null, un objet vide!)
- Comment faut-il faire attention quand on utilise cette méthode?

À toi de jouer! - UML en Action!

- Sur une feuille, crée un diagramme UML simple avec les classes:
  - Inventaire
  - Item
  - Joueur
- Indique les attributs principaux et les méthodes les plus importantes
- N'oublie pas les flèches et les chiffres pour montrer les relations et les quantités

Besoin d'Aide? Contacte le Support!

- UML basics : https://www.uml-diagrams.org/
- Association/agrégation/composition UML : https://creately.com/blog/diagrams/composition-association-aggreagation/
- JSON et Gson : https://github.com/google/gson/blob/master/UserGuide.md