Projet Java: Deviens un Architecte du Code!

Ton Plan d'Action:

Le Projet Modèle: On décortique un exemple pro!

MVC: Le Trio Gagnant (Modèle, Vue, Contrôleur)

POO: Tous les Super Pouvoirs du Code Réunis!

Eclipse: Deviens un Ninja de l'IDE!

Le Projet Modèle: Analyse du Terrain

- Imagine un jeu vidéo: On va regarder comment organiser tous les fichiers
- src/: La base, le coeur du projet
  - main/: Le point de départ, comme le menu principal
  - modele/: Les objets du jeu (Personnage, Item, etc.)
  - controller/: L'orchestre, celui qui fait le lien entre les objets et l'écran
  - ...

Arborescence du Projet: La Carte

- src/
  - ├── main/ (points d’entrée Main.java, MainConsole.java)
  - ├── modele/ (Cartes, Personnage, Joueur, Monstre, Combat, Case)
  - ├── controller/ (ControleurJeu, ControleurJeuConsole)
  - ├── engine/ (MoteurDeJeu logique métier)
  - ├── view/ (MenuJeu, panneaux Swing et console)
  - ├── utils/ (GenererMonde, GestionSauvegarde)
  - └── test/ (futurs tests unitaires)

MVC: Diviser pour Mieux Régner

- MVC = Modèle, Vue, Contrôleur: Un trio qui assure!
  - Modèle: Les données, les objets (Personnage, Item, etc.)
  - Vue: Ce que l'utilisateur voit (l'écran, la console)
  - Contrôleur: Le chef d'orchestre, il gère les actions de l'utilisateur et met à jour le modèle et la vue

Packages: Chaque Chose à sa Place

- modele: Les classes métiers, comme les acteurs du jeu
- engine: Le coeur du jeu, les règles du jeu
- controller: Le marionnettiste, il contrôle tout!
- view: L'interface, ce que le joueur voit
- utils: Des outils pratiques pour le code
- main: La porte d'entrée du jeu

Diagramme UML: La Carte du Code

- Imagine un plan pour visualiser les relations entre les classes
  - Personnage <= Joueur, Monstre: Héritage (le Monstre EST un Personnage)
  - Inventaire --- Joueur: Association (le Joueur A un Inventaire)
  - VueConsole ---> Controller: Agrégation (le Contrôleur utilise la Vue)

À toi de jouer! - Analyse du Modèle

- Prends une feuille et un crayon (ou ton logiciel préféré)
- Liste toutes les classes qui font partie du modèle de notre jeu
- Dessine les relations entre ces classes (héritage, association...)

À toi de jouer! - Organise le Code!

- Toujours sur ta feuille, dessine un diagramme des packages
- Indique quelles classes vont dans chaque package
- Justifie tes choix: Pourquoi cette classe va dans ce package?

Module 2: Code en Action!

- Bienvenue dans le monde de l'implémentation et de l'évolution du code!

Encapsulation: Protège Tes Classes!

- Vérifie si les attributs sont bien private: On ne veut pas que n'importe qui puisse les modifier!
- Regarde bien les méthodes: Les public doivent être utilisées avec précaution!
- Si tu vois du code qui se répète, crée une interface!

Le Pouvoir du Polymorphisme

- On utilise une List<Personnage> pour gérer Joueurs et Monstres ensemble
- L'Inventaire utilise une List<Item> pour ranger tout et n'importe quoi
- Admire la beauté du code qui s'adapte à toutes les situations!

Refactoring: Améliore Ton Code!

- Le refactoring, c'est comme ranger ta chambre: Tu ne changes rien au fonctionnement, mais c'est plus propre!
- Eclipse a des outils géniaux pour ça: Utilise-les!
- Renommer une méthode (Rename refactoring): Facile!
- Extraire une partie du code dans une nouvelle méthode (Extract Method): Top pour la clarté!
- Déplacer une classe vers un autre package (Move): Pour bien ranger les fichiers!

Tests Unitaires: Teste Ton Code!

- Les tests unitaires: Des petits programmes qui vérifient que ton code fait ce qu'il doit faire
- Crée un dossier test et une classe MoteurDeJeuTest
- Écris un test pour vérifier qu'un combat se déroule bien: Qui va gagner?

Module 3: Eclipse à Fond!

- On va explorer les outils cachés de l'IDE!

Eclipse: Deviens un Expert

- Recherche de références (Ctrl+Shift+G): Où est utilisée cette méthode?
- Open Declaration (F3) et Go to Definition: Voyage au coeur du code!
- TODO et tâches (Favoris): Pense-bête intégré!

Documentation: Laisse une Trace!

- JavaDoc: La doc automatique, générée à partir de tes commentaires
- Visualise JavaDoc en passant la souris (Ctrl+Hover): Super pratique!
- Pense à bien commenter ton code, c'est important pour les autres (et pour toi dans 6 mois!)

Git: Sauvegarde et Partage!

- Git: Ton allié pour ne jamais perdre ton code et travailler en équipe
- Initialise un repository Git (comme créer un coffre-fort)
- Commit: Enregistre les changements (comme faire une sauvegarde)
- Push: Envoie ton code sur Internet (comme partager ton coffre avec tes amis)

Packaging: Emballe Ton Jeu!

- Transforme ton code en un fichier JAR (exécutable): Comme un cadeau prêt à offrir!
- Configure Run As > Java Application pour lancer ton jeu
- Tu peux même créer plusieurs configurations pour tester différentes options

Scrum: Travaille en Equipe!

- Scrum: Une méthode pour organiser le travail en équipe (comme un plan d'attaque!)
- Tâches, User Stories, Backlog: Le vocabulaire des pros
- Utilise Eclipse Mylyn (ou Jira) pour gérer les problèmes et les idées (comme un tableau de bord)

Ressources Utiles

- Organisation projet Java (Oracle) : https://docs.oracle.com/javase/tutorial/extra/module/
- MVC Java simple : https://www.javatpoint.com/mvc-architecture
- Refactoring Eclipse : https://www.vogella.com/tutorials/EclipseRefactoring/article.html
- JUnit 5 démarrage : https://junit.org/junit5/docs/current/user-guide/