

# Programmation Orientée Objet (OBJET)

TP 7 : Finalisation de « World of ECN »

—

Finalisation du projet

Jean-Marie Normand – Bureau E211

[jean-marie.normand@ec-nantes.fr](mailto:jean-marie.normand@ec-nantes.fr)



# Instructions

- Suivez les slides les uns après les autres
- A la fin de chaque séance de TP, vous devrez nous rendre un rapport par binôme
- Ce rapport devra contenir :
  - Une introduction et une présentation rapide du sujet de la séance
  - Les réponses aux questions posées dans les slides repérés par une icône de panneau STOP
  - Une conclusion
- La notation tiendra compte du respect de ces consignes



# **1<sup>RE</sup> PARTIE : MISE À NIVEAU (RATTRAPAGE DU RETARD ÉVENTUEL)**

# Avant toute chose

- Assurez vous d'avoir bien terminé les séries de TP précédentes
- Si ce n'est pas le cas, prenez le temps d'arriver dans un **état fonctionnel de WoE**
- En particulier avec une version permettant à un humain de contrôler un **Personnage** au clavier



# Interface Graphique (Minimaliste)

- Afin de pouvoir jouer à WoE il nous faut une **interface graphique** même **minimaliste** (le mode texte est bien évidemment suffisant car nous n'avons pas vu les GUI en cours)
- Proposez un moyen de **visualiser le plateau de jeu de manière textuelle** (éventuellement en proposant une légende, p. ex. G = Guerrier, etc.)
- En plus de l'affichage du monde de WoE, veillez à bien **présenter au joueur humain les choix qui s'offrent** à lui à chaque tour de jeu (**déplacement/combat/sauvegarde**)



# Interface Graphique (Minimaliste)

- Préparez un rapport avec les questions posées lors du TP 6
- Et avec un bilan de votre travail sur ce projet
- Listez en particulier :
  - Les problèmes rencontrés
  - Votre ressenti sur les progrès réalisés au cours de ce projet
  - Des modifications à proposer sur le sujet actuel



# Finalisation de WoE

- Et voilà ! Nous avons vu (presque) tous les concepts que nous souhaitions aborder dans le projet du cours de Programmation Orientée Objet
- Maintenant vous devez avoir une **version minimaliste mais fonctionnelle** de WoE !
- Il ne vous reste plus qu'à **finaliser** les dernières **classes et méthodes** que vous avez pu laisser de côté
- Veillez à bien **commenter votre code** et à bien **écrire la Javadoc**
- Veillez aussi à **nettoyer votre code** !



# Conclusion

- Ajoutez à votre rapport :
  - **L'illustration du bon fonctionnement** de votre fonction principale (sortie textuelle des tests effectués)
- Rendez une archive au format **.ZIP** nommée **OBJET-TP7-NomBinome1-NomBinome2.zip (avec NomBinome1 < nomBinome2 dans l'ordre alphabétique)** contenant :
  - Votre rapport au format **.pdf**
  - Tous vos fichiers **.java**
  - **Veillez à bien avoir écrit la Javadoc** de tous les **attributs** de vos classes et des **principales méthodes** (déplacer, combattre, etc.)
  - **Faites générer la Javadoc** par NetBeans, **joignez** l'ensemble des fichiers résultats à l'**archive .zip dans un dossier documentation**
- Le respect de ces consignes est pris en compte dans la note !



# Surprise !

- Attention nous allons garder les 2 dernières heures pour une évaluation par les pairs
- Pour pouvoir faire ce test il faut :
  - Cliquer droit sur votre projet NetBeans → Properties
  - Dans la catégorie **Run** Assurez vous que le nom de la classe principale « Main Class » soit bien celle qui correspond à votre main de démonstration (celui avec la version fonctionnelle de WoE)
  - Dans la catégorie « **Sources** » assurez vous que « **Source/Binary Format** » soit bien pour **11** et Encoding à **UTF-8**
  - Ce n'est pas fini ! Lisez la suite...

# Surprise !

- Attention nous allons garder les 2 dernières heures pour une évaluation par les pairs
- Pour pouvoir faire ce test il faut :
  - Modifier le fichier **POM.xml** de votre projet Maven, pour ce faire :
  - Cliquer droit sur le nom de votre projet et sélectionnez « **Open POM** », cela va ouvrir le fichier dans l'éditeur NetBeans
  - Rajouter après le `</properties>` et avant le `</project>` le code XML suivant

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <version>2.4.1</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <mainClass>NOM.DE.CLASSE.COMPLET</mainClass>
          </manifest>
        </archive>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
    </plugin>
  </plugins>
</build>
```

**Doit intégrer les packages !**



# Surprise !

- Attention nous allons garder les 2 dernières heures pour une évaluation par les pairs
- Pour pouvoir faire ce test il faut :
  - Attention à bien avoir mis le nom complet de votre classe (avec le packaging complet donc)
  - Cliquez sur « **Clean and Build** » pour re-générer votre exécutable
  - Aller dans le répertoire **target** de votre projet → normalement vous y trouverez 2 fichiers .jar
    1. NomDeVotreProjetNetBeans-XX.jar (où XX correspond à la **<version>** de **<artifactId>** dans le POM.xml, tout en haut de votre fichier)
    2. NomDeVotreProjetNetBeans-XX-jar-with-dependencies.jar ← c'est celui qu'on doit utiliser !
  - Renommer le fichier jar généré (qui doit s'appeler « **XXX-jar-with-dependencies.jar** ») en « **LettreQuiVoussaEteAttribuee.jar** »

# Surprise !

- Attention nous allons garder les 2 dernières heures pour une évaluation par les pairs
- Pour pouvoir faire ce test il faut :
  - Tester que l'exécution **fonctionne en mode terminal** :
    - **Copier** le fichier .jar dans un dossier (p. ex. dans un dossier sur votre bureau)
    - **Ouvrir un terminal** et **aller dans le dossier** contenant ce .jar (avec la commande « cd ... »)
    - **Lancer l'exécution** du .jar avec la commande suivante :
      - « **java -jar VotreLettre.jar** »

