Programmation Orientée Objet (OBJET)

TP 2 : Début du projet « World of ECN »

Compréhension diagramme de classes UML – Écriture des premières classes Java

Jean-Marie Normand – Bureau E211
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

1RE PARTIE : CRÉATION D'UN PROJET JAVA

Création d'un projet

- En vous basant sur le slides du TP1 :
  - Créez un projet « ProjetTP »
  - Créez un package « `org.centrale.projet.objet` »
  - Rajoutez la classe `Point2D` écrite lors du TP1 à ce nouveau projet
  - Rajoutez la classe `TestPoint2D` écrite lors du TP1 à ce nouveau projet
- **Assurez vous du bon fonctionnement** de ce nouveau projet !
- **Illustrez** le dans votre rapport

2E PARTIE : COMPRÉHENSION D'UN DIAGRAMME DE CLASSES UML

Diagramme de classe UML

- En vous basant sur le diagramme UML suivant (disponible en version plus lisible dans le fichier `UML-Seance1.png`) :

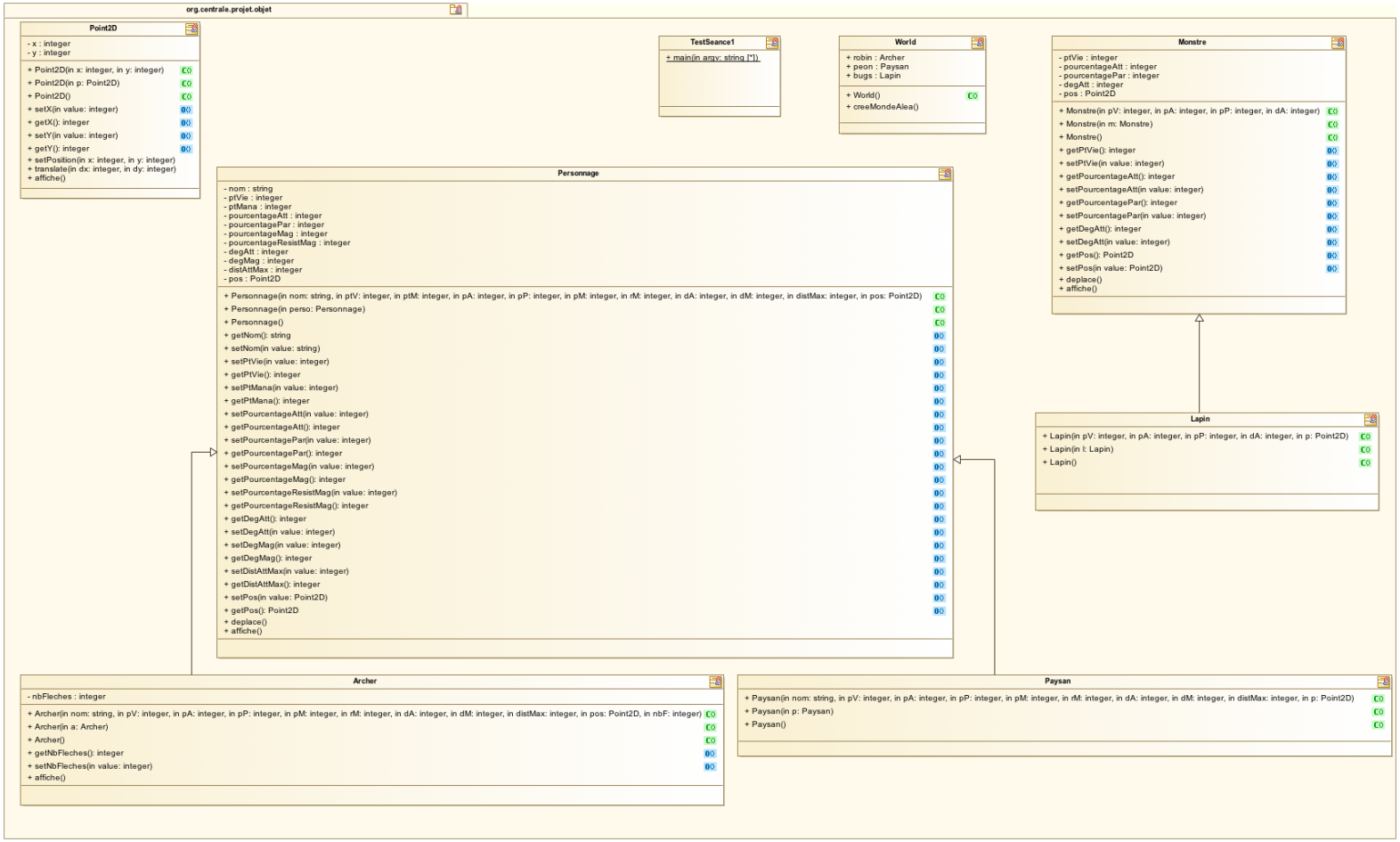
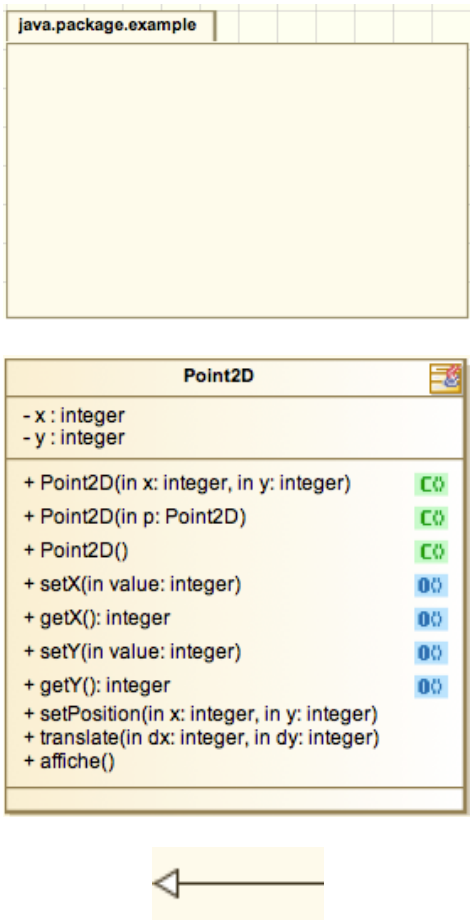


Diagramme de classe UML

- Rappels UML :
  - Package (rappel : vous pouvez voir les packages comme des dossiers permettant de stocker/ ranger nos classes)
  - Classes :
    - Attributs
    - Constructeurs
    - Accesseurs/Mutateurs
    - Méthodes
  - Lien d'héritage



World of ECN – WoE

- Le diagramme de classes représente un ensemble (incomplet pour l'instant) de classes basiques d'un mini jeu de rôle simpliste que nous allons écrire ensemble pendant les séances de TP
- Ce jeu de rôle est assez similaire à celui utilisé en cours dans les exemples illustrant les concepts de la POO !
- Nous allons détailler brièvement quelques classes de ce diagramme

Détails de quelques classes (1)

- La plupart des classes sont explicites et ne nécessitent pas de détails particuliers
- `Point2D` : représente un point à coordonnées entières en 2 dimensions
- `Personnage`, `Archer`, `Paysan`, `Monstre` et `Lapin` représentent les différents types de protagonistes possibles **dans cette version de WoE**

Détails de quelques classes (2)

- `World` : cette classe a pour but de représenter le « monde » dans lequel évolueront les différents protagonistes de WoE
  - On suppose le monde 2D « plat »
  - On ne donne pas de taille à ce monde (au moins pas pour le moment)
  - On suppose que les protagonistes ont tous une position de type `Point2D` dont on suppose les coordonnées correctes (au moins pour l'instant)
  - Pour ce TP, le monde contient uniquement 4 protagonistes :
    - Un `Archer`
    - Un `Paysan`
    - Un `Lapin` (vous pouvez en créer deux si vous voulez)
    - Pour faciliter les tests ces attributs seront publics
  - Une méthode `creeMondeAlea` doit permettre de positionner ces protagonistes de manière aléatoire, mais :
    - Ils ne doivent pas être sur la même position 2D
    - Assurez-vous qu'ils soient relativement proches les uns des autres (i.e. la distance entre deux éléments ne doit pas dépasser 5 unités)

Compléments sur la génération de nombres entiers pseudo-aléatoires en Java (1)

- Java fournit différents mécanismes pour la génération de nombres pseudo-aléatoires.
- Pour les entiers il faut utiliser la classe `Random` du paquetage `java.util`
- Pour ce faire, nous devons :
  - créer un objet de type `Random`
  - utiliser sa méthode `nextInt` : qui prend en paramètre un entier **N** et qui génère un nombre pseudo-aléatoire dans l'intervalle **[0, N[**

Compléments sur la génération de nombres entiers pseudo-aléatoires en Java (2)

- Vous trouverez un exemple de programme générant 10 nombres entiers pseudo-aléatoires entre 0 et 99 dans le fichier `RandomInteger.java` (voir sur Hippocampus)
- N'hésitez pas à copier/coller le code et/ou à le modifier pour tester le fonctionnement de l'objet `Random` et de ses méthodes
- Rappel : la **Javadoc** est très importante pour comprendre le fonctionnement des classes Java !
- URL : http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

Compléments sur la génération de nombres entiers pseudo-aléatoires en Java (3)

```java
// Dans un fichier RandomInteger.java
import java.util.Random;

// On genere 10 entiers pseudo-aleatoires dans l'intervalle [0,99]
public class RandomInteger {
  public static void main(String[] args) {
    System.out.println("Generation de 10 nombres entiers dans l'intervalle [0,99]");

    // NB : il est INUTILE de creer plusieurs objets de type Random
    // UN SEUL suffit pour generer plusieurs
    // nombres pseudo-aleatoires
    Random generateurAleatoire = new Random();

    // Boucle de generation des 10 nombres
    for(int i=0; i<10; i++) {
      // 100 ici definit la borne sup de l'intervalle
      int entierAlea = generateurAleatoire.nextInt(100);
      System.out.println("On vient de generer : "+entierAlea);
    }
  }
}
```

- Generation de 10 nombres entiers dans l'intervalle [0,99]
- On vient de generer : 67
- On vient de generer : 63
- On vient de generer : 46
- On vient de generer : 37
- On vient de generer : 28
- On vient de generer : 87
- On vient de generer : 23
- On vient de generer : 81
- On vient de generer : 3
- On vient de generer : 17

Détails de quelques classes (3)

- `TestSeance1` : cette classe a pour but de créer les différents objets nécessaires aux tests de la classe `World` et des autres classes implémentées lors de cette séance :
  - C'est une classe spéciale car elle contient une fonction principale `main`
  - Vous êtes responsables de son implémentation
  - Veillez à bien illustrer le bon fonctionnement des principales classes

Diagramme de classe UML

- A partir de ce diagramme UML (disponible en version lisible dans un fichier à part) :
  - **Décrivez** brièvement ce que représente chacune des classes pour vous (sauf les classes expliquées ci-avant)
  - **Décrivez** brièvement les différentes relations existantes entre les classes du diagramme
  - **Expliquez** pourquoi ces choix ont été faits d'après vous
  - Avez-vous oui ou non une alternative possible au diagramme à proposer ? **Justifiez**
  - Devez vous apporter des modifications à votre projet NetBeans par rapport à la configuration du slide 4 ? **Justifiez**
  - **Implémentez** les classes décrites dans le diagramme en respectant :
    - Les noms des classes
    - Les noms, portées et types des attributs
    - Les noms, portées et arguments des méthodes

Diagramme de classe UML

- Joignez à votre rapport (contenant les réponses aux questions précédentes) :
  - Le fichier **World.java**
  - Le fichier **TestSeance1.java**
  - **L'illustration du bon fonctionnement** de votre fonction principale (sortie textuelle des tests effectués)


Success
NEXT EXIT