

EF1

Bases de la Programmation Orientée Objet

- Mini-projet : la bataille de Faërun

Mini-projet : la bataille de Faërun

Bleus



Entraîne

Met en jeux



dé 3 faces

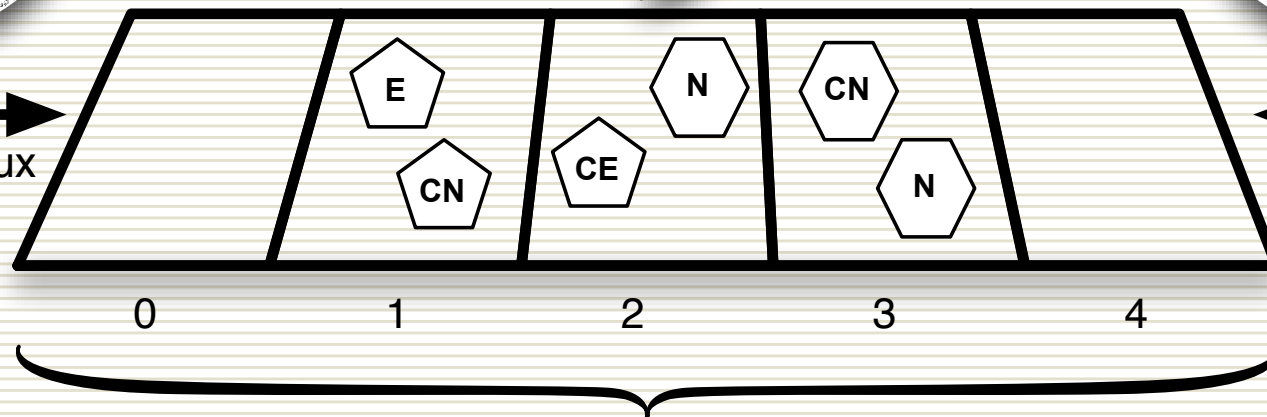
combat
sur ce
carreau

Rouges



Entraîne

Met en jeux



avancée des bleus

avancée des rouges

Mais avant de jouer ...

... Il faut concevoir et développer :

- Les **guerriers**

- ◆ Tester un combat entre deux guerriers

- Les **châteaux**

- ◆ Tester l'entraînement des guerriers

- Le **plateau**

- ◆ Tester le déplacement des guerriers
- ◆ Tester le combat entre guerriers sur le même carreau

- L'**application principale** (moteur de jeu)

- ◆ Tester le jeu



ETAPE 1 : LES GUERRIERS

Guerrier de base

✓ Caractéristiques de base :

■ **Force**

- ◆ 10 de base
- ◆ Utilisée pour calculer les dégâts lors d'un combat
 - Dégâts = somme de n lancers de dés de 3 faces (n égale à la force)

■ **Point de vie (PV)**

- ◆ 100 de base 100
- ◆ À 0 points de vie le guerrier meurt

Mais différents type de guerriers

✓ Nain

- subit 2x moins de dégâts qu'un guerrier de base

✓ Elfe

- a une force 2x supérieure à un guerrier de base

✓ Chef nain

- subit 2x moins de dégât qu'un nain

✓ Chef elfe

- a une force 2x supérieure à un elfe

Les nains encaissent mieux et les elfes tapent plus fort !

Combat entre guerriers

✓ Exemple

■ Nain tape un elfe

- ◆ Nain[PV=100] tape sur Elfe[PV=100]
- ◆ Dégâts : somme de 10 lancés de dés de 3 = 22
- ◆ Elfe subit 22 de dégâts et voit ses PV descendre à 78.

■ L'elfe se défend !

- ◆ Elfe[PV=78] tape sur Nain[PV=100]
- ◆ Dégâts : somme de 20 lancés (force x2) de dés de 3 = 38
- ◆ Nain subit 19 de dégât (dégât /2) et voit ses PV descendre à 81.

■ Fin de la passe d'arme : Nain[PV=81] et Elfe[PV=78]

Exercice 1

- ✓ Brainstorming : dessiner le schéma UML correspondant aux Guerriers.
- ✓ Ecrire les classes correspondantes.
- ✓ Ecrire un test de combat entre deux guerriers.
 - Itération jusqu'à la mort d'un des deux guerriers .

Dynamique vs statique

✓ Programmation orientée objet (POO)

■ Programmation **dynamique**

- ◆ En rapport avec la **création des objets**
- ◆ Association, encapsulation, héritage

VS

■ Programmation **statique**

- ◆ En rapport avec la **classe seulement**
- ◆ Attributs et méthodes de classe

Attributs et méthodes de classe (1/4)

- ✓ Un attribut ou une méthode de classe sont **exploitables même si la classe n'a pas été instanciée** (même s'il n'y a pas de création d'objet)
- ✓ **ATTENTION :**
 - Les attributs et les méthodes de classe ne peuvent pas accéder aux attributs et aux méthodes propres aux objets créés dans la classe

Attributs et méthodes de classe (2/4)

✓ Exemples extraits de l'API Java

■ Dans la classe Math :

```
public static final double PI
```

■ Dans la classe Integer :

```
public static String toString(int i)
```

✓ Utilisation :

```
double pi = Math.PI;    // donne la valeur de PI  
String chaine = Integer.toString(3);
```

Attributs et méthodes de classe (3/4)

```
public class PlateauUtilitaire {  
  
    // Attribut contenant un objet de type random  
    private static final Random RANDOM = new Random();  
  
    // Méthode qui simule le lancement d'un dé de 3 faces  
    // et retourne le résultat  
    public static int De3() {  
        return RANDOM.nextInt(3)+1;  
    }  
  
    // Méthode qui simule plusieurs lancers d'un dé et retourne le résultat  
    public static int De3(int nombreDes) {  
        int somme = 0;  
        for (int i = 0; i < nombreDes; i++) {  
            somme = somme + De3();  
        }  
        return somme;  
    }  
}
```

Attributs et méthodes de classe (4/4)

- ✓ Utilisation de la **méthode de classe** *De3(...)* dans la classe Guerrier

- `NomDeLaClasse.nomDeLaMethode(...)`

```
public void attaquer(Guerrier guerrier) {  
    int degat = PlateauFactory.De3(getForce());  
    ...  
}
```

- Rappel : dégâts = somme de n lancés de dés de 3 faces (avec n = force du guerrier)

Classe Utilitaire

- ✓ **Regrouper** au sein d'une classe des méthodes **statiques** permettant de manipuler des objets
 - afficher, transformer, etc.
- ✓ Permettre de simplifier l'usage des objets dans le projet.
- ✓ Exemple :

```
public class GuerrierUtilitaire {  
    public static void printlnGuerrier(Guerrier guerrier) {...  
    public static void printlnCombat(...
```



ETAPE 2 : LES CHÂTEAUX

Château et entraînement

- ✓ Un château contient des guerriers **novices**.
- ✓ Pour qu'un guerrier novice accède au plateau, il faut qu'il soit **entraîné** !
- ✓ À chaque tour, un château entraîne un nombre limité de guerriers **en fonction de ses ressources**.

Ressources nécessaires

✓ Coût d'entraînement des guerriers

- 1 de base
- Nain : coût de base
- Elfe : 2
- Chef nain : 3
- Chef elfe : 4

Pour ce faire, utiliser la notion de constante

Constantes (1/4)

✓ Mot clé **final**

```
public static final double PI
```

- Signification « cela ne peut pas changer »
- Utilisation : attributs, méthodes et classes

Constantes (2/4)

- ✓ Un attribut final est **immuable**

- ✓ Deux possibilités
 - Constante initialisée lors de la **compilation**
 - Constante initialisée lors de l'**exécution**

- ✓ **ATTENTION** lors de la référence à un objet.
 - La référence est immuable
 - L'objet est modifiable !

Constantes (3/4)

```
public class Data {  
  
    // Constante à la compilation  
    ... final int CONSTANCE_1 = 9;  
  
    // Constante à l'exécution  
    ... final int CONSTANCE_2 = (int) (Math.random() * 20);  
  
    // Référence constante  
    ... final Value CONSTANCE_3 = new Etudiant(...);  
  
    ...  
    public void modifier(...) {  
        // MAIS l'objet est modifiable  
        CONSTANCE_3.setLogin(...)  
    }  
}
```

Constantes (4/4)

✓ Méthodes final

- Empêche toutes sous-classes de la redéfinir
 - ◆ Préserve le comportement d'une méthode durant l'héritage
- Toutes méthodes privées sont implicitement final

✓ Classes final

- Une classe ne peut pas hériter d'une classe final

Exemple d'entraînement

✓ Initialisation :

- Château à 3 de ressources
- Ordre d'entraînement donné par le joueur :
 - ◆ 1:Nain, 2:Nain, 3:Elfe, 4:Elfe (File d'attente)

✓ Tour 1 : château possède 3 ressources

- Entraînement de 1:Nain **OK** (-1 ressource)
- Entraînement de 2:Nain **OK** (-1 ressource)
- Entraînement de 3:Elfe **PARTIEL** (-1 ressource)
- Etat final : château 0 ressources, 2 nains prêts à se battre, 1 elfe (manque 1 ressource)

✓ Tour 2 : château récupère 1 en ressource

- Entraînement de 3:Elfe **OK** (-1 ressource)
- Entraînement de 4:Elfe **IMPOSSIBLE POUR LE MOMENT**
- Etat final : château 0 ressources, 1 elfe prêts à se battre

✓ Tour3 : château récupère 1 en ressource

- Entraînement de 4:Elfe **IMPOSSIBLE POUR LE MOMENT**
- Etat final : château 1 ressources

Exercice 2

- ✓ Brainstorming : dessiner le schéma UML de la classe Château permettant l'entraînement des guerriers.
- ✓ Modifier les classes Guerriers.
- ✓ Ecrire la classe Château.
- ✓ Rappel : un château contient des guerriers novices.
A chaque tour, elle entraîne un nombre limité de guerriers en fonction de ses ressources.
 - Initial : 3 en ressources
 - +1 par tour

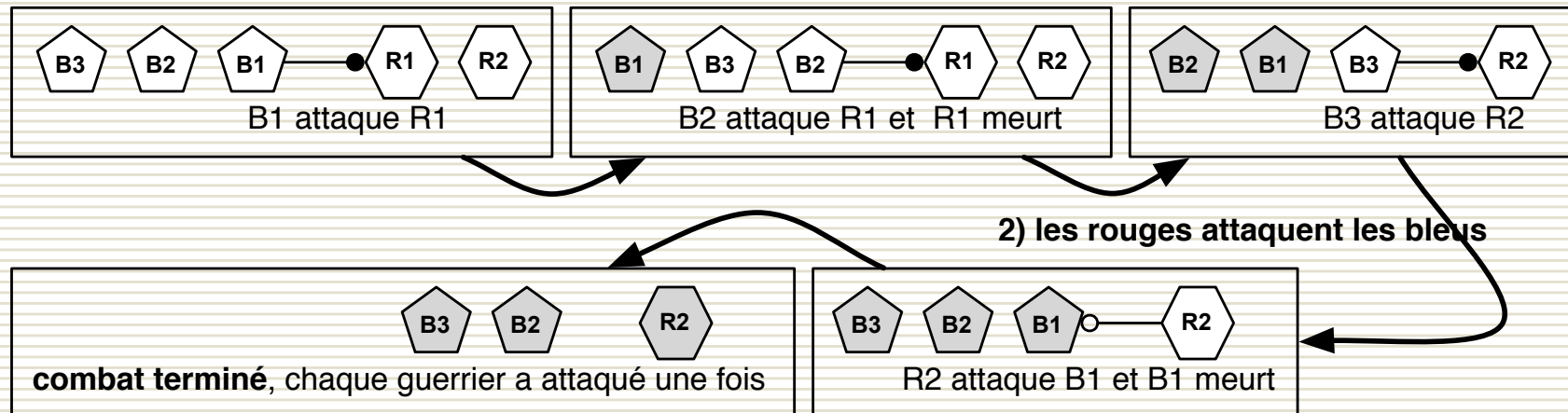


LE PLATEAU DE JEU

Plateau de jeu

- Plateau constitué de **carreaux**
 - ◆ éviter le mot case, mot clé en java
- Longueur 5, 10 ou 15 carreaux
- A chaque tour, les guerriers sur le plateau avancent vers le château ennemi
- Si des guerriers ennemis sont sur le même carreau, ils s'arrêtent et se battent !

1) les bleus attaquent les rouges



Exercice 3

- ✓ Brainstorming : dessiner le schéma UML correspondant au plateau de jeu.
- ✓ Ecrire les classes correspondantes.
- ✓ Brainstorming : déplacement des guerriers sur le plateau.



LE MOTEUR DE JEU

Classe Application : le moteur de jeu

- ✓ Init : les joueurs donnent les **ordres** d'entraînement
- ✓ Déroulement du jeu pour un tour
 - Les châteaux **entraînent** les guerriers
 - Les guerriers entraînés se **placent** de chaque côté du plateau
 - Les guerriers sur le plateau **avancent** d'un carreaux
 - ◆ Sauf si le carreau contient des guerriers bleus et rouges
 - Le carreau contenant des guerriers bleus et rouges lance un **combat**

Démonstration (à disposition sur l'intratek)

- ✓ L'équipe gagnante : premier guerrier arrivé aux portes du château ennemi