

## TP7 : Héritage et implémentation d'interfaces

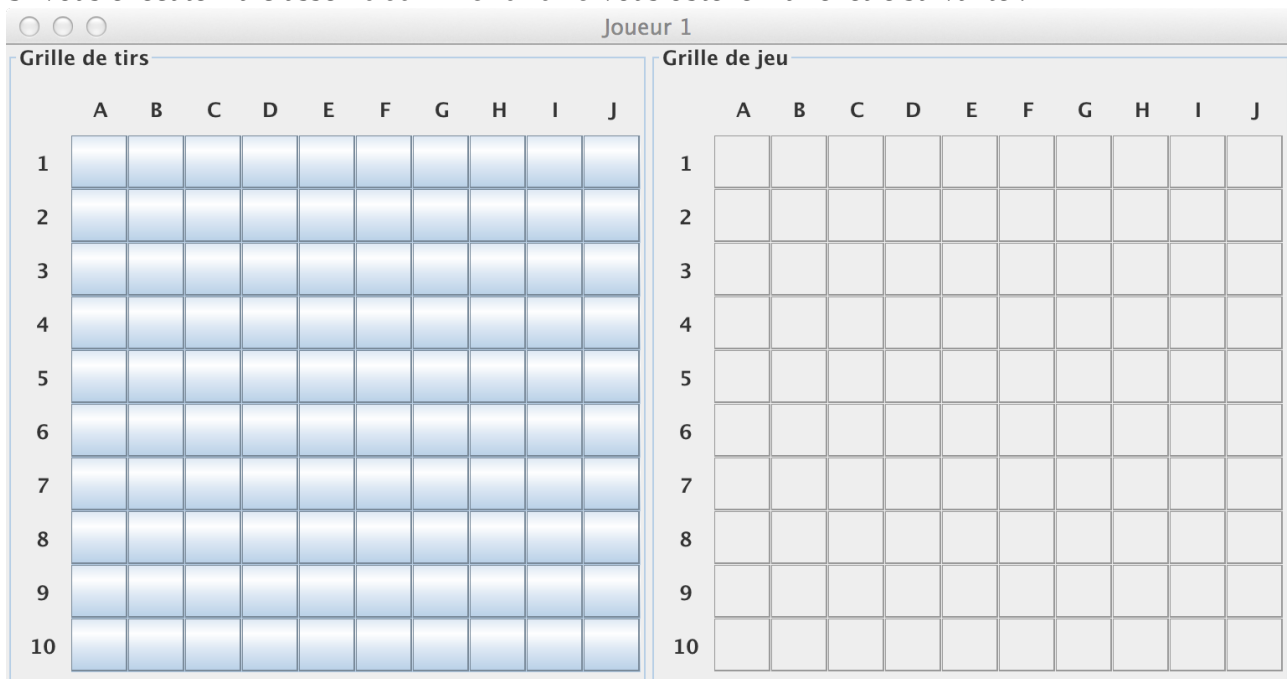
Le but de ce TP est de mettre en place l'interface graphique de la bataille navale. Pour cela, nous allons utiliser le composant GrilleGraphique qui permet de représenter graphiquement une Grille.

Les classes fournies sont les suivantes :

- GrilleGraphique
- BatailleNavale
- Joueur

### 1- La classe BatailleNavale

Si vous exécutez la classe BatailleNavale vous obtenez la fenêtre suivante :



Ce programme initialise une fenêtre de jeu pour un joueur. Un joueur possède deux grilles, sa grille de tir qui représentera les tirs envoyés sur son adversaire et leur état (touché ou à l'eau) et sa grille de jeu qui représentera ses navires et également l'ensemble des tir reçus.

Dans un premier temps, pour représenter graphiquement l'état de de la grille de jeu, nous allons devoir « connecter » les deux classes GrilleGraphique (qui dessine la grille) et GrilleNavale (qui contient l'état de la grille du joueur).

Pour cela, vous allez faire les étapes suivantes :

- Créer une classe GrilleNavaleGraphique qui hérite de GrilleNavale
- Ajouter un attribut de type GrilleGraphique à GrilleNavaleGraphique et l'initialiser dans le constructeur GrilleNavaleGraphique(int taille).

- Ajouter la méthode **public** GrilleGraphique getGrilleGraphique() qui retourne l'attribut de type GrilleGraphique
- Redéfinir les méthodes **public boolean** ajouteNavire(Navire n) et **public boolean** recoitTir(Coordonnee c) pour qu'en plus d'ajouter un navire ou un tir, elles colorient les cases correspondantes. Les navires seront coloriés en vert (Color.GREEN), les tir à l'eau en bleu (Color.BLUE), et les tirs touchés en rouge (Color.RED).

Modifiez le code de BatailleNavale pour tester votre nouvelle classe :

```
GrilleGraphique grilleAttaque = new GrilleGraphique(10);
GrilleNavaleGraphique grilleJoueur = new GrilleNavaleGraphique(10);
grilleJoueur.placementAuto(new int[]{5,4,3,3,2,2});
initFenetre("Joueur 1", grilleAttaque, grilleJoueur.getGrilleGraphique());
```

## 2- Gestion du Jeu

Etudiez la classe abstraite Joueur fournie.

Créez la classe JoueurGraphique, sous-classe de Joueur. Et déclarez toutes les méthodes héritées abstraites en laissant les corps de méthodes vides pour l'instant (normalement Eclipse permet de la faire automatiquement).

La classe JoueurGraphique doit avoir un seul constructeur qui prend en paramètre une instance de GrilleNavaleGraphique (la grille de jeu) et une instance de GrilleGraphique (grille de tirs) qui seront ajoutés en attribut.

Maintenant, modifiez votre programme principal pour qu'il y ait deux instances de JoueurGraphique. Puis appelez, la méthode jouerAvec(...) du premier joueur en passant en paramètre le deuxième Joueur.

```
public static JoueurGraphique initJoueur(String nomJoueur) {
    GrilleGraphique grilleAttaque = new GrilleGraphique(10);
    GrilleNavaleGraphique grilleJoueur = new GrilleNavaleGraphique(10);
    grilleJoueur.placementAuto(new int[]{5,4,3,3,2,2});
    initFenetre(nomJoueur, grilleAttaque, grilleJoueur.getGrilleGraphique());
    return new JoueurGraphique(grilleJoueur, grilleAttaque);
}
```

A l'exécution, vous devriez obtenir deux fenêtres (superposées). On ne peut pas faire d'interaction pour le moment.

### 2.1- Gestion de la réception des attaques (méthode defendre)

Réalisez le corps de la méthode int defendre(Coordonnee c) pour que le tir sur la coordonnée c soit envoyé sur la grille de jeu. En fonction de l'impact de ce tir vous devrez retourner l'une des valeurs suivantes ; Joueur.TOUCHE, Joueur.COULE, Joueur.A\_L\_EAU ou JOUEUR.GAME\_OVER

### 2.2- Gestion des retours d'attaque (coloriage de la grille de tir)

Maintenant, on va s'occuper de colorier la grille de tirs en fonction du retour de l'attaque.

Dans l'algorithme de jeu (c.f. méthode `jouerAvec(...)` de `Joueur`), après que la méthode `defendre(Coordonnee c)` d'une instance de `Joueur` soit appelée, la méthode `retourAttaque(Coordonnee c, int etat)` de l'adversaire est appelée avec comme paramètre la coordonnée `c` attaquée et le résultat de l'attaque `etat`. En fonction du résultat du tir, ce paramètre pourra être égal aux constantes suivantes : `Joueur.TOUCHE`, `Joueur.COULE`, `Joueur.A_L_EAU` ou `JOUEUR.GAME_OVER`

Réalisez la méthode `retourAttaque(...)` pour qu'elle colorie la case attaquée dans la grille de tir. Si l'attaque est un succès la case sera coloriée en rouge, si l'attaque tombe à l'eau alors la case sera coloriée en bleu.

Modifiez votre programme principal pour que l'un des joueurs réalise une attaque sur son adversaire et vérifiez que cela fonctionne comme prévu.

### 2.3- Gestion du choix de l'attaque

Le but va être d'envoyer un tir sur la grille de l'adversaire lorsque l'on clique sur une case de la grille de tir. Pour cela, réalisez la méthode `Coordonnee choisirAttaque()` pour qu'elle renvoie la coordonnée sélectionnée de la grille de tir (allez voir dans la classe `GrilleGraphique` la méthode qui permet de retourner la coordonnées sélectionnée).

Testez et vérifiez que cela fonctionne bien.

### 2.4- Ajout de messages informatifs

En fonction des résultats d'attaque et de défense, placez dans les méthodes : `retourAttaque(Coordonnee c, int etat)` et `retourAttaque(Coordonnee c, int etat)` des messages informatifs. Pour cela, utilisez la fonction suivante :

```
JOptionPane.showMessageDialog(grilleTirs  
, "Bateau touché en "+c);
```