

Projet Bataille Navale

L2 - Info 2

*Laïter Léonard
Lefranc Joaquim*

*Réalisé en 2015 dans le cadre du
projet informatique.*



Sommaire

I. Présentation et déroulement

- 3. Interface*
- 8. Déroulement du jeu*
- 11. Connexion Serveur / Client*

II. Mécanismes internes

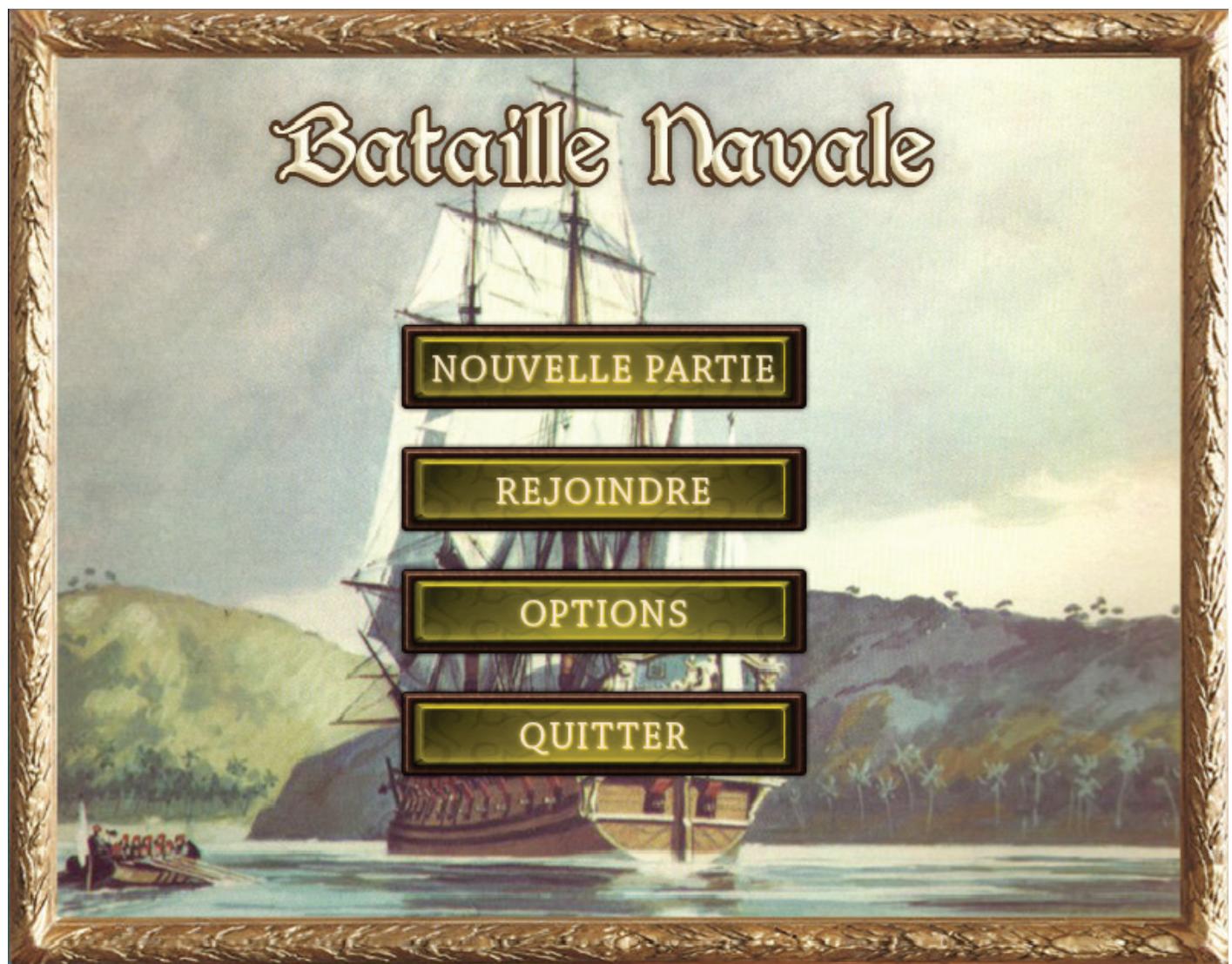
- 12. Fonctionnement du protocole*
- 14. Imbrication des JPanel*
- 15. Gestion des sons*
- 16. Diagramme UML*

Interface

Lancement du jeu

Lors du lancement le menu apparaît. On y trouve différents boutons :

- **Nouvelle Partie** : Permet de créer une partie
- **Rejoindre** : Permet de rejoindre une partie créée
- **Options** : Permet de modifier les options audio
- **Quitter** : Quitte le jeu



Interface

Nouvelle partie

L'écran de création de partie permet de lancer le serveur et la recherche d'un joueur lorsque l'on a placé les navires.



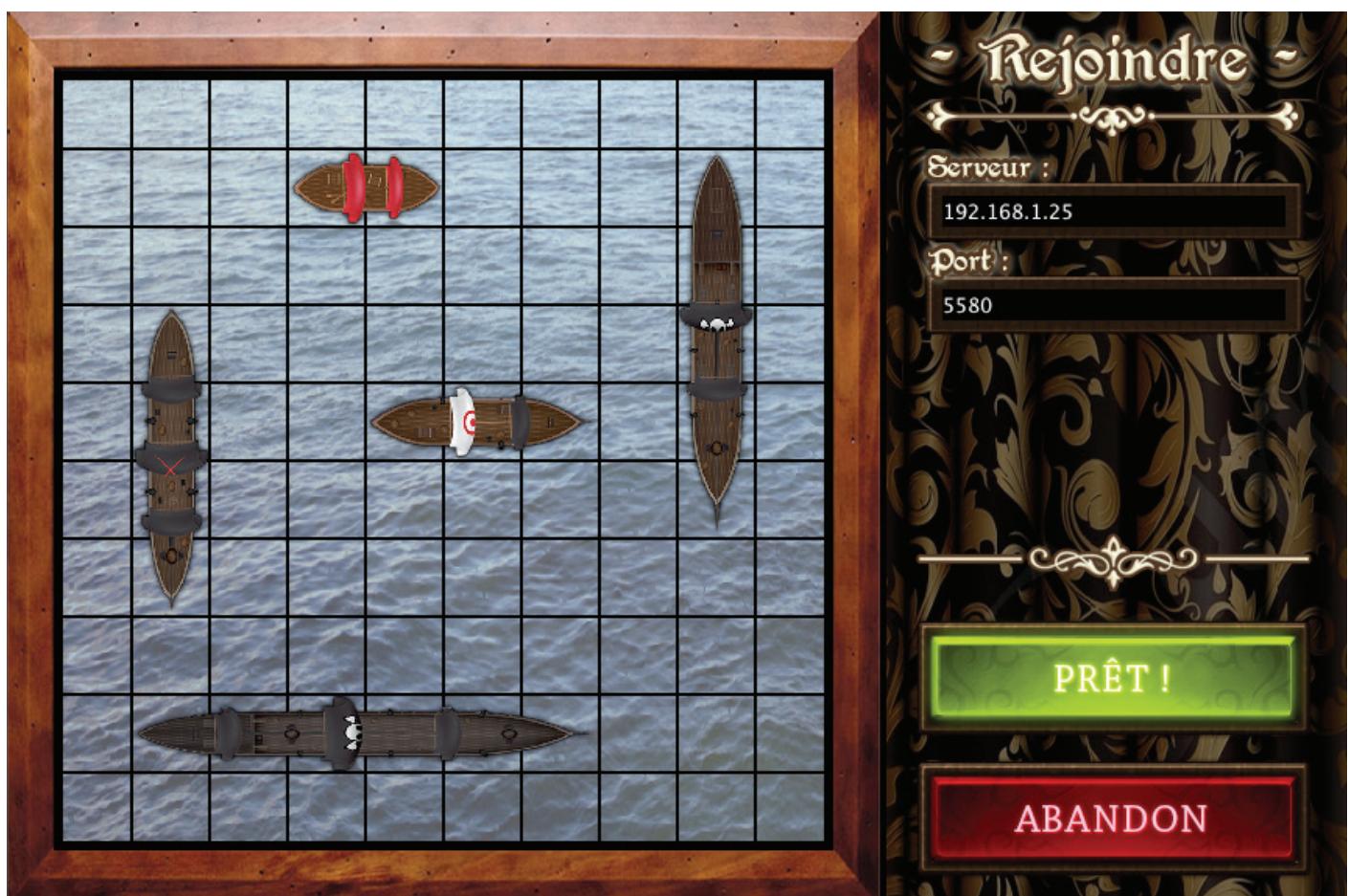
- **Serveur** : est à laisser tel quel, il est automatiquement mis à jour en récupérant l'adresse ip courante.
- **Port** : lui peut être modifié par défaut le 5580 est utilisé.
- **Connexion locale** : Permet de choisir si l'on utilise l'ip du réseau local ou bien l'ip externe.
- **Jouer tout seul** : Lance un jeu pour tirer sur ses propres navires, un puzzle game de mémorisation.

Une fois les navires placés, il suffit donc de faire «Prêt !» pour lancer la recherche d'un joueur.

Interface

Rejoindre une partie

L'écran pour rejoindre une partie est similaire à la création, il permet de rejoindre la partie créée par un joueur.



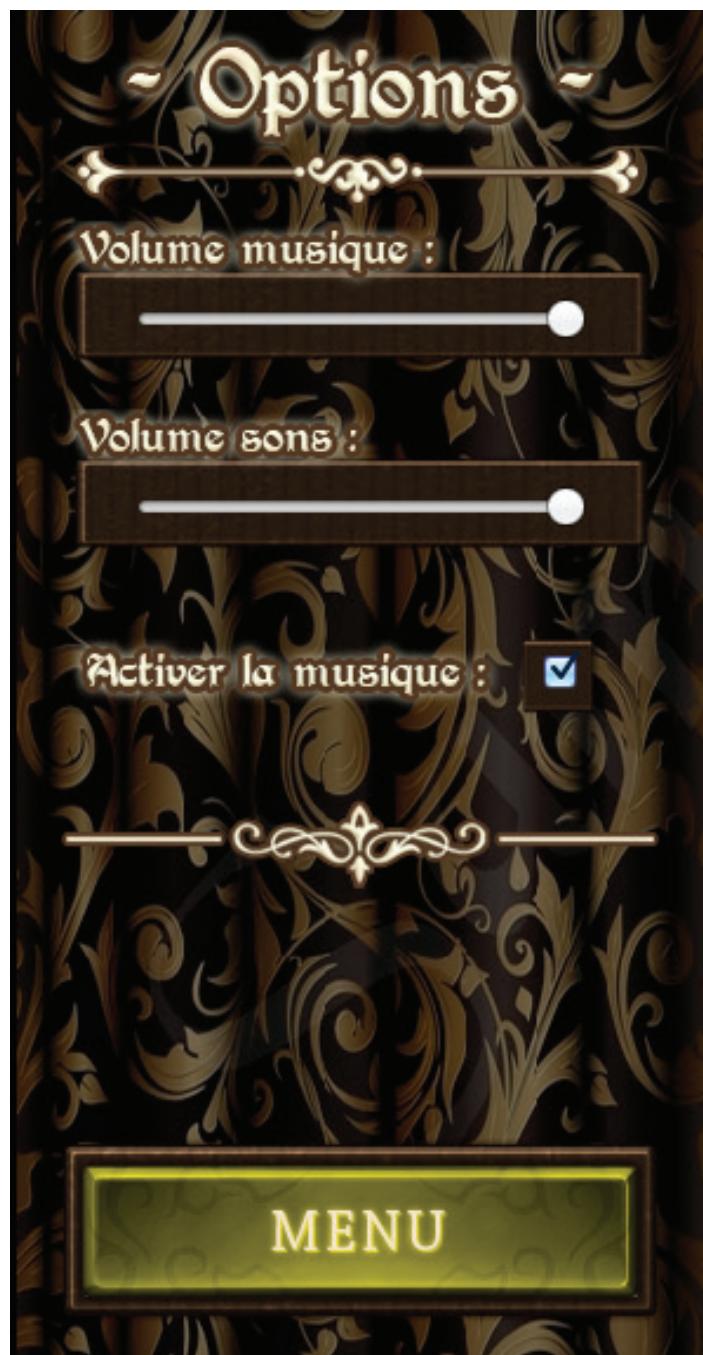
- **Serveur :** il faut indiquer l'adresse du serveur auquel le jeu doit se connecter.
- **Port :** le port utilisé par défaut est 5580

Une fois les navires placés, il suffit donc de faire «Prêt !» pour lancer la connexion à la partie distante.

Interface

Options

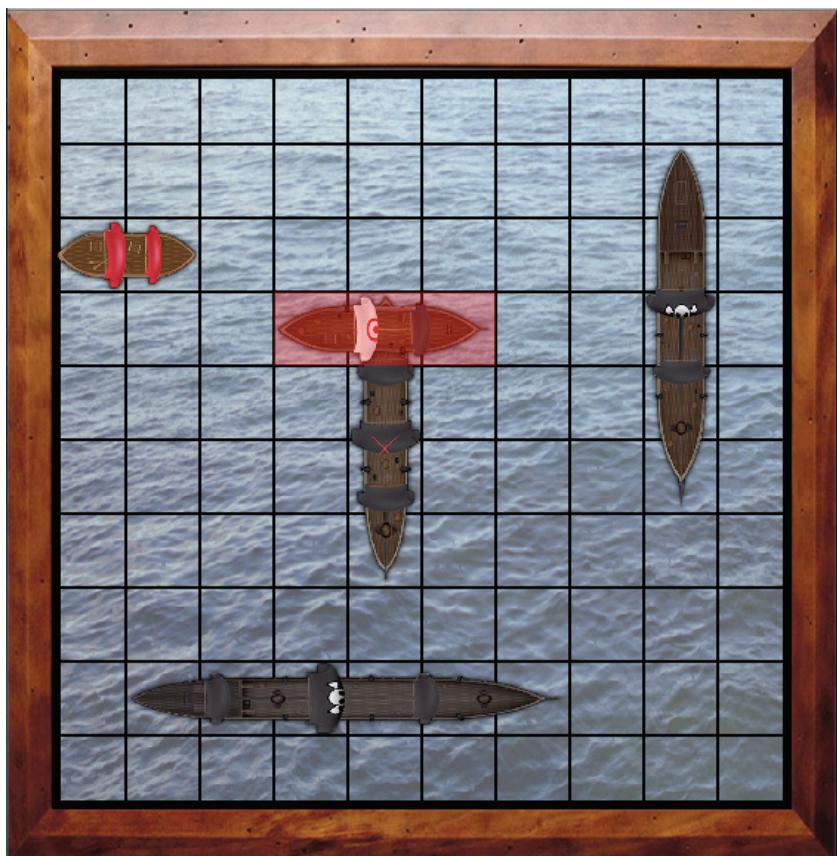
Le panel des options permet d'activer ou non la musique et de régler le volume de sortie de la musique et des sons.



Interface

Le placement des navires

Deux navires ne peuvent être superposés, si un tel cas se produit, un des navires passe en mode erreur (il devient rouge) et le bouton «Prêt !» n'est plus disponible.



L'abandon

Le bouton abandon permet comme son nom l'indique d'abandonner une partie ou bien la création de partie. Une confirmation est demandée pour éviter la déconnexion inopinée.

Déroulement du jeu

L'attente d'une connexion

Après avoir lancé la création d'une partie, on arrive sur un écran de recherche de joueur. Il y a un délai de 60 sec pour que le joueur adverse rejoigne la partie, sinon la création échoue et le serveur est fermé.

Un abandon entraîne une fermeture prématuée du serveur.



Déroulement du jeu

L'écran de partie

Lors du début de partie, un joueur est tiré au hasard pour commencer. Celui qui commence à le bouton «Tirer !» disponible, le second doit attendre de recevoir un tir.



- **Ma flotte** : ce sont les navires du joueur, ce qui permet de savoir à tout moment l'état de sa propre flotte.
 - **La grille** : deux types d'impact sont symbolisés sur la grille, les touchés (Explosion) et les ratés (Éclaboussure).
 - **Le curseur de visé** : Permet de déterminer les coordonnées du tir.

Déroulement du jeu

Les scores

Dès qu'un des joueurs n'a plus aucun navire en état de flotter, le panel des scores est déclenché sur l'interface des deux joueurs.

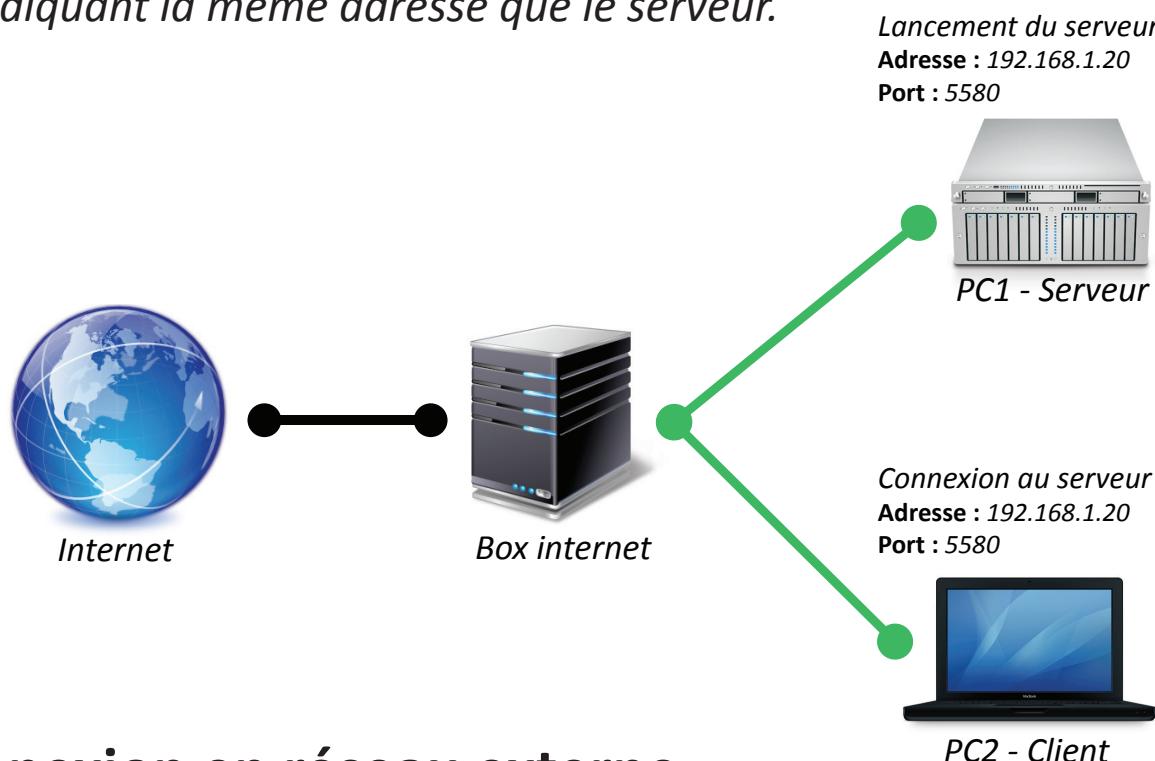


- **Navires restants** : *nombre de navires non détruits par l'adversaire.*
- **Nombre de tirs** : *total des tirs effectués.*
- **Nombre de ratés** : *total des tirs ratés.*

Connexion Serveur / Client

Connexion en réseau local

En connexion locale, il suffit juste que le client rejoigne la partie en indiquant la même adresse que le serveur.



Connexion en réseau externe

Pour une connexion distante, il faut rajouter une règle NAT sur la box du créateur de partie pour indiquer vers quel ordinateur du réseau local les requêtes vers le port 5580 doivent être redirigées.

Connexion au serveur
Adresse : 65.75.85.200
Port : 5580



PC2 - Client

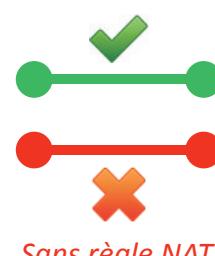
Lancement du serveur
Adresse : 65.75.85.200
Port : 5580



PC1 - Serveur

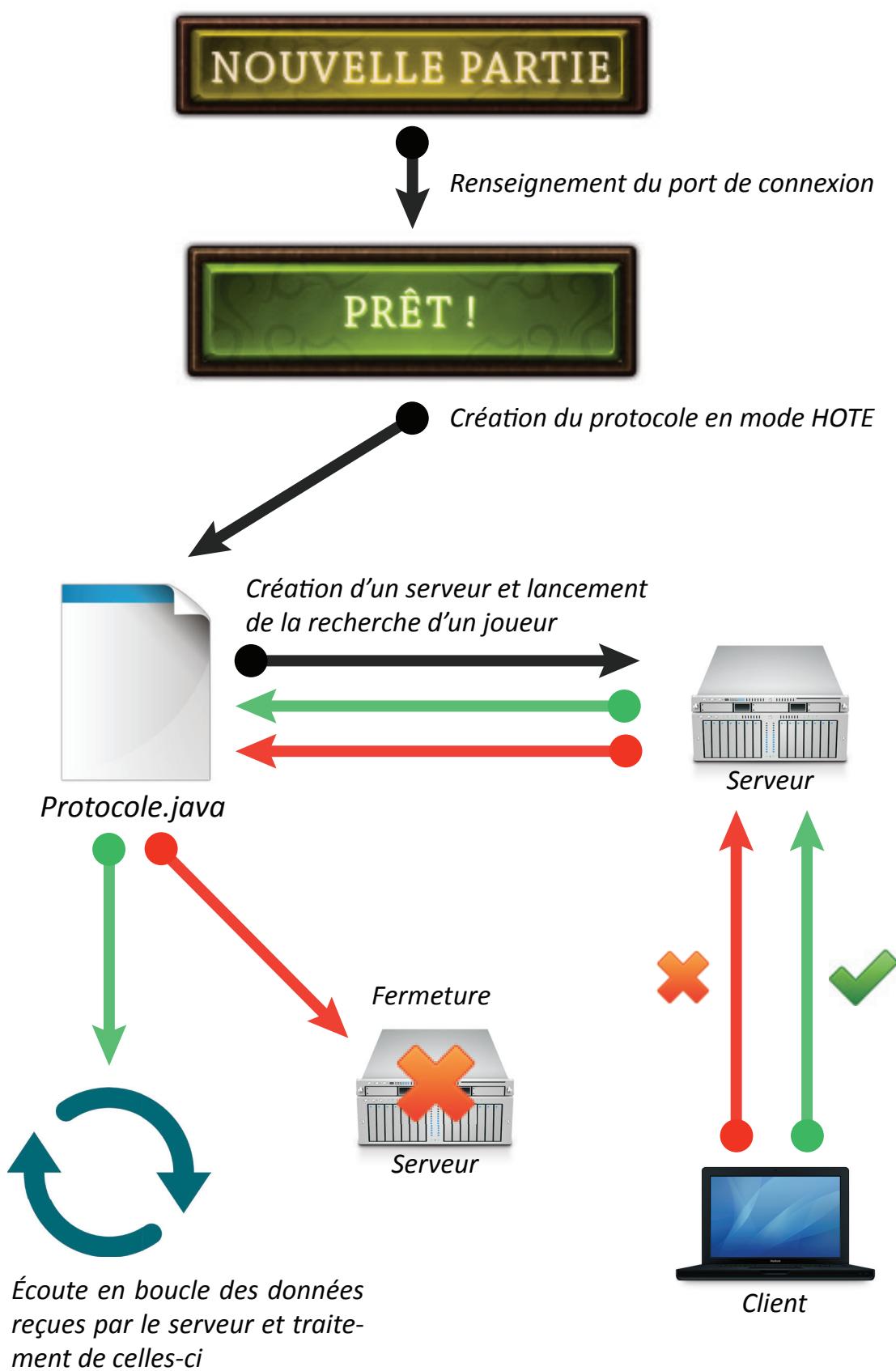


Règle NAT
5580 → PC1 (192.168.1.XX)



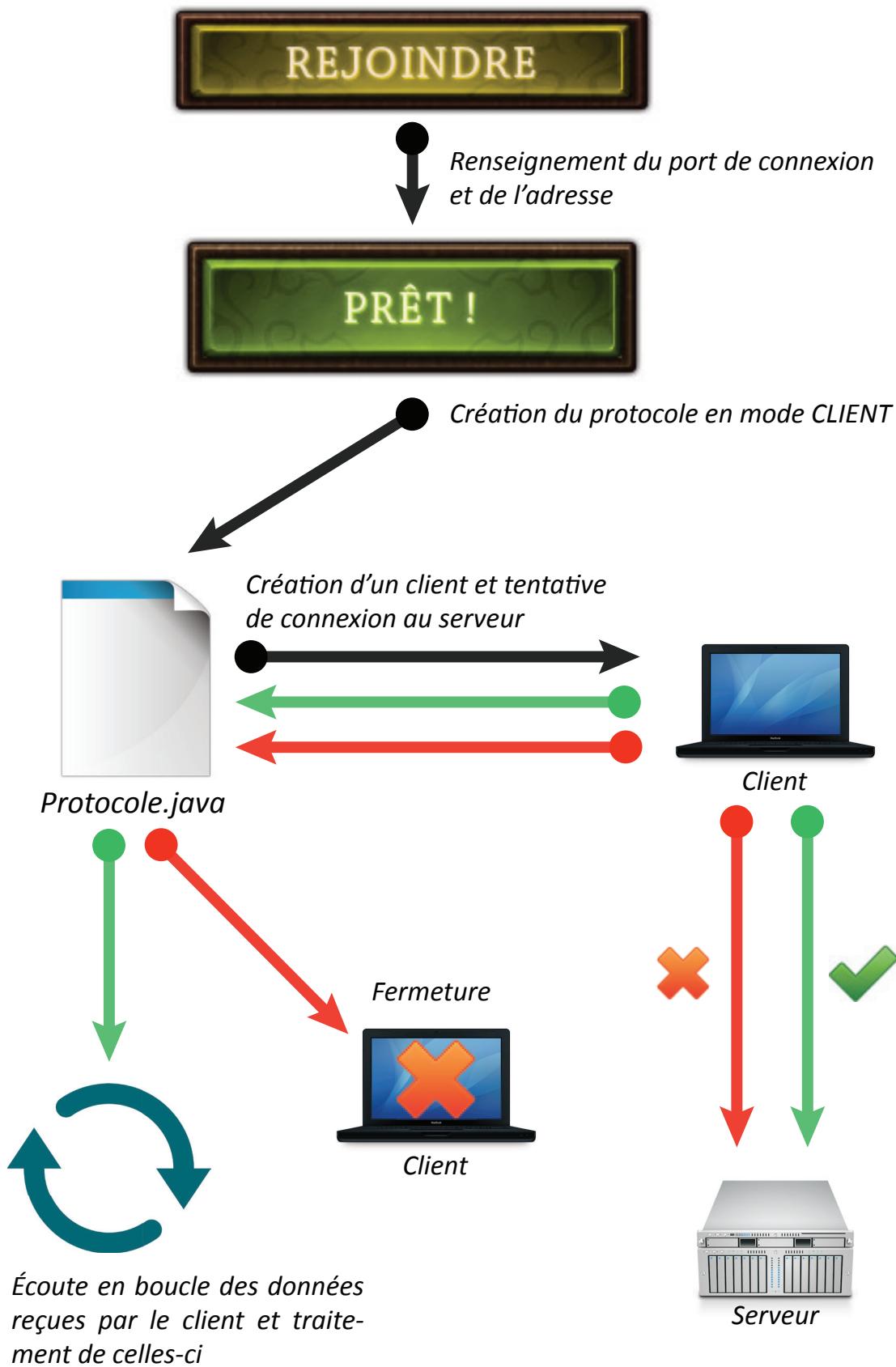
Fonctionnement du protocole

Création d'une partie (Serveur)



Fonctionnement du protocole

Rejoindre une partie (Client)



Imbrication des JPanel

Découpe de l'interface



Plateau.java



GrilleGUI.java



CaseEmpty.java



CaseWater.java



CaseExplosion.java



SideBarre.java



InformationPanel.java



ActionsPanel.java



GhostShipGUI.java



CaseCross.java

Gestion des sons

Sounds.java, AudioPlayer.java

La gestion des sons de l'interface à été pensée pour une simplicité d'utilisation. Le but principal étant d'avoir une banque de sons prédéfinis dans la class Sounds.java grâce à des méthodes statiques. Ce qui permet une utilisation facile dans tout le projet en utilisant un simple appel de méthode comme indiqué ci-dessous.



Sounds.java



La class Sounds contient des fonctions statiques chargées d'instancier un objet AudioPlayer et de le lancer.

Dans le programme, il suffit donc ensuite d'appeler une des méthodes contenue dans Sounds.java pour jouer un son.

Exemple : Sounds.eventSoundExplosion();



AudioPlayer.java

InputStream(fichier.wav)



Fichier audio WAV

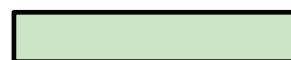
BatNav (GUI)

v.5

- Update I.I -

Légendes :

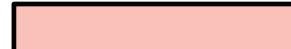
Classes mères :



Classes héritées :



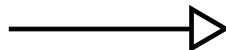
Interfaces, abstraites :



Enumérations :



Héritage :



Implémentation :

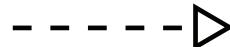


Diagramme UML

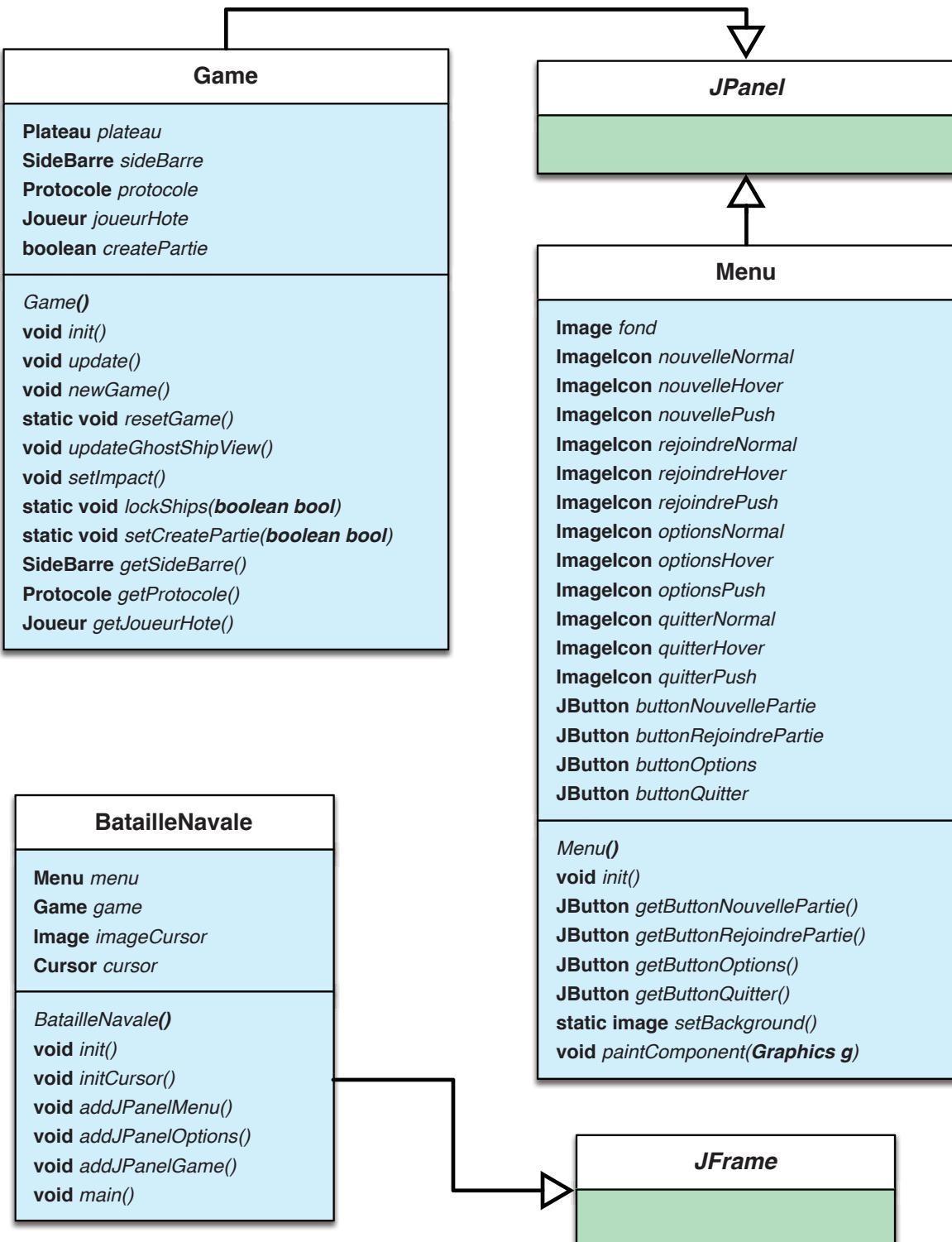


Diagramme UML

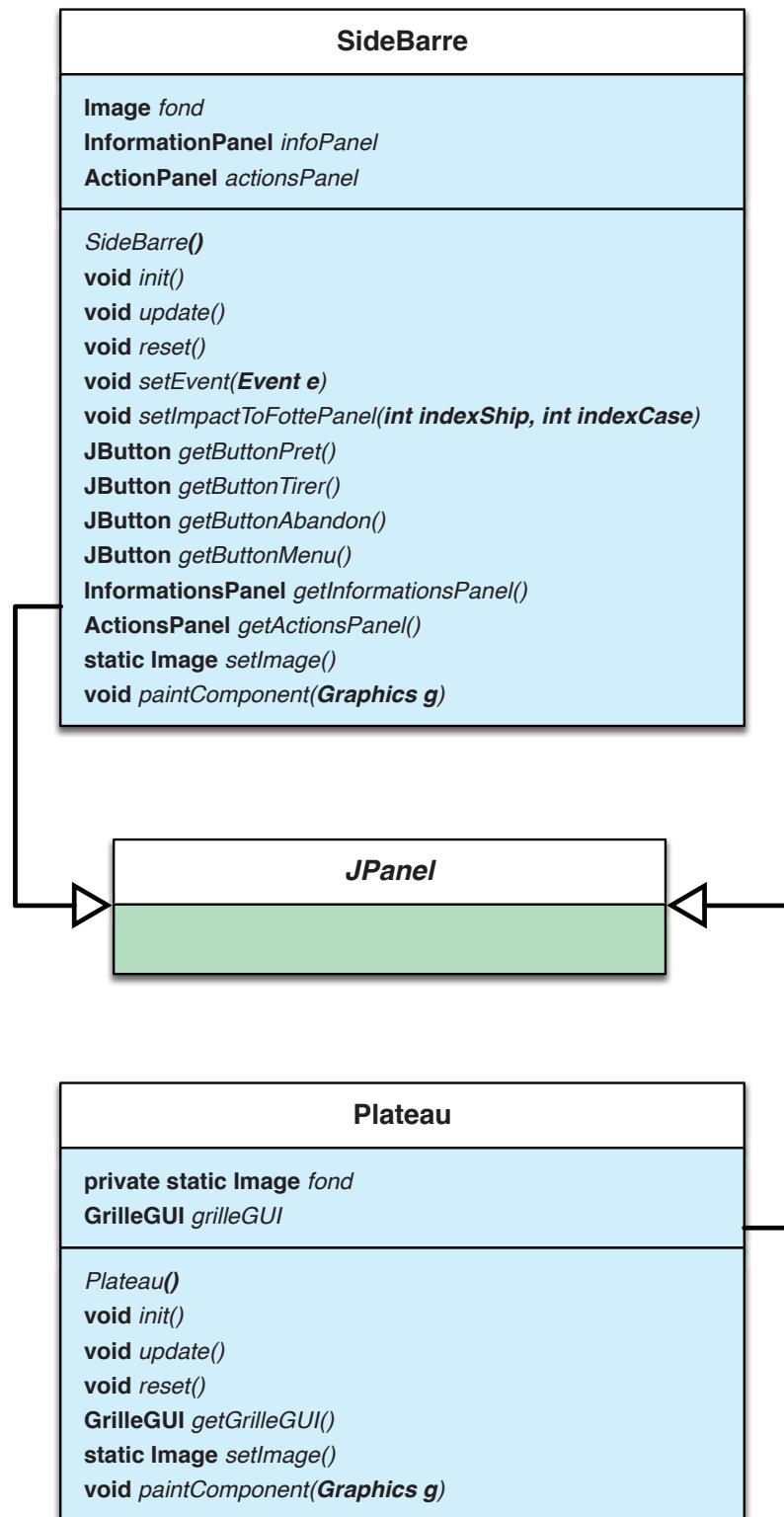


Diagramme UML

ActionsPanel	ActionsPanel
<pre>ImageIcon tirerNormal ImageIcon tirerHover ImageIcon tirerPush ImageIcon pretNormal ImageIcon pretHover ImageIcon pretPush ImageIcon abandonNormal ImageIcon abandonHover ImageIcon abandonPush ImageIcon menuNormal ImageIcon menuHover ImageIcon menuPush ImageIcon ouiNormal ImageIcon ouiHover ImageIcon ouiPush ImageIcon nonNormal ImageIcon nonHover</pre>	<pre>ImageIcon nonPush ImageIcon img_waitPlayer ImageIcon img_scores ImageIcon img_abandonConfirm static JButton buttonPret static JButton buttonTirer JButton buttonAbandon JButton buttonMenu JButton buttonOui JButton buttonNon JLabel waitPlayer JLabel scores JLabel abandonConfirm boolean confirm TypeActionsBox typePanel BoxNumber scoreRestants BoxNumber scoreNbTirs BoxNumber scoreNbRates</pre>
<pre>ActionsPanel() void init() void setTypePanel(TypeActionsBox type) void setScores(int restants, int nbTirs, int nbRates) void setEventConfirmAbandon() void setAbandonConfirm(boolean bool) static void setEnabledButtonPret(boolean bool) static void setEnabledButtonTirer(boolean bool) JButton getButtonPret() JButton getButtonTirer() JButton getButtonAbandon() JButton getButtonMenu() JButton getButtonOui() JButton getButtonNon()</pre>	<pre>JPanel InformationsPanel</pre>
	<pre>InformationsPanel() void init() void setTypePanel(TypeInfoPanel type) CreationPanel getCreationPanel() FlottePanel getFlottePanel() OptionsPanel getOptionsPanel() ScorePanel getScorePanel()</pre>

Diagramme UML

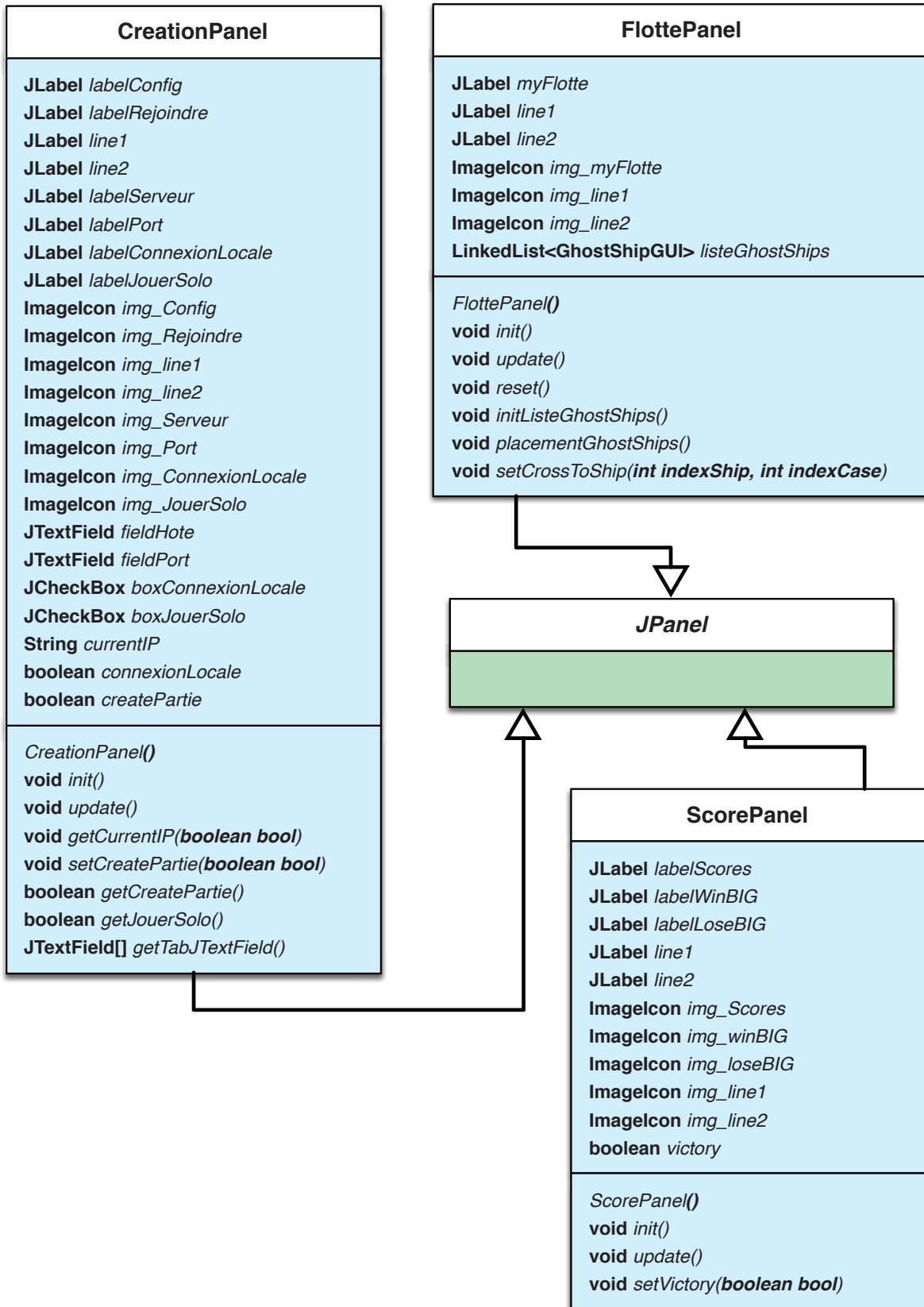


Diagramme UML

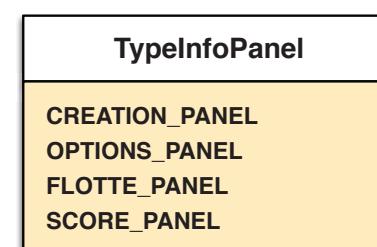
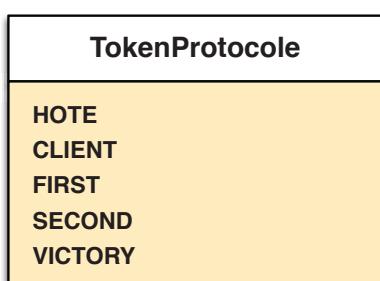
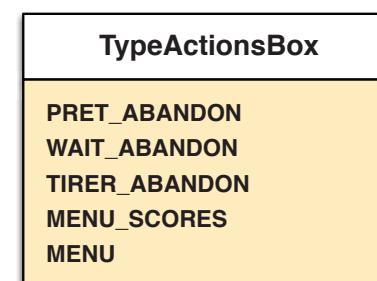
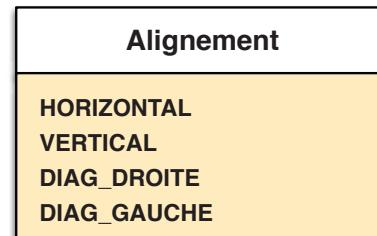
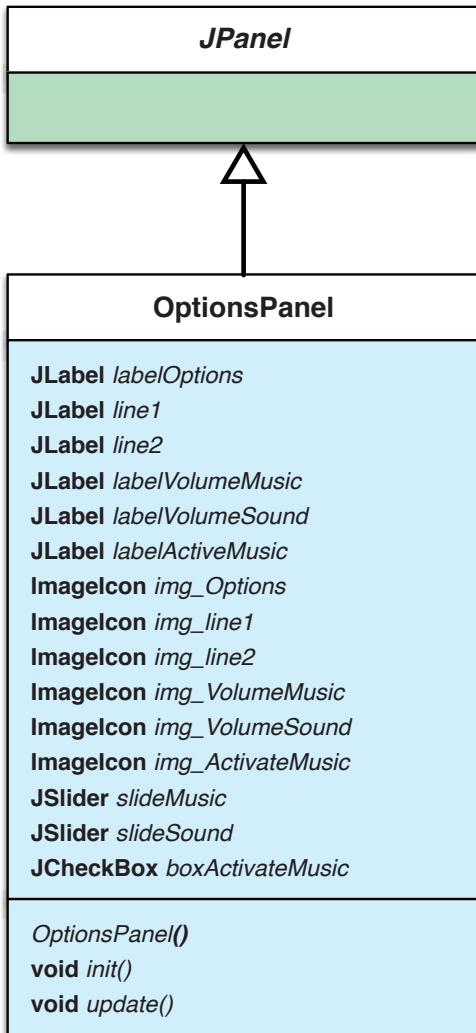


Diagramme UML

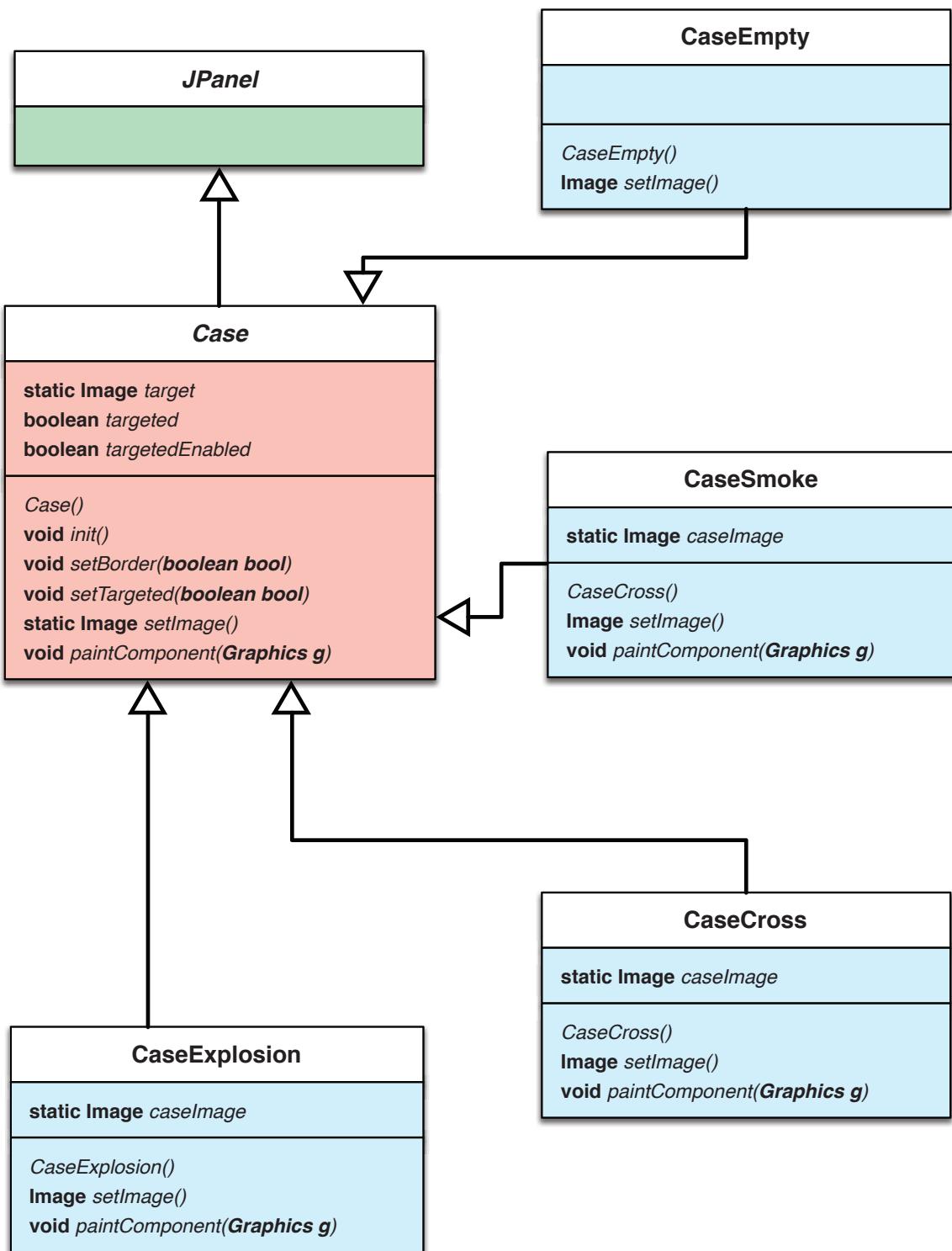


Diagramme UML

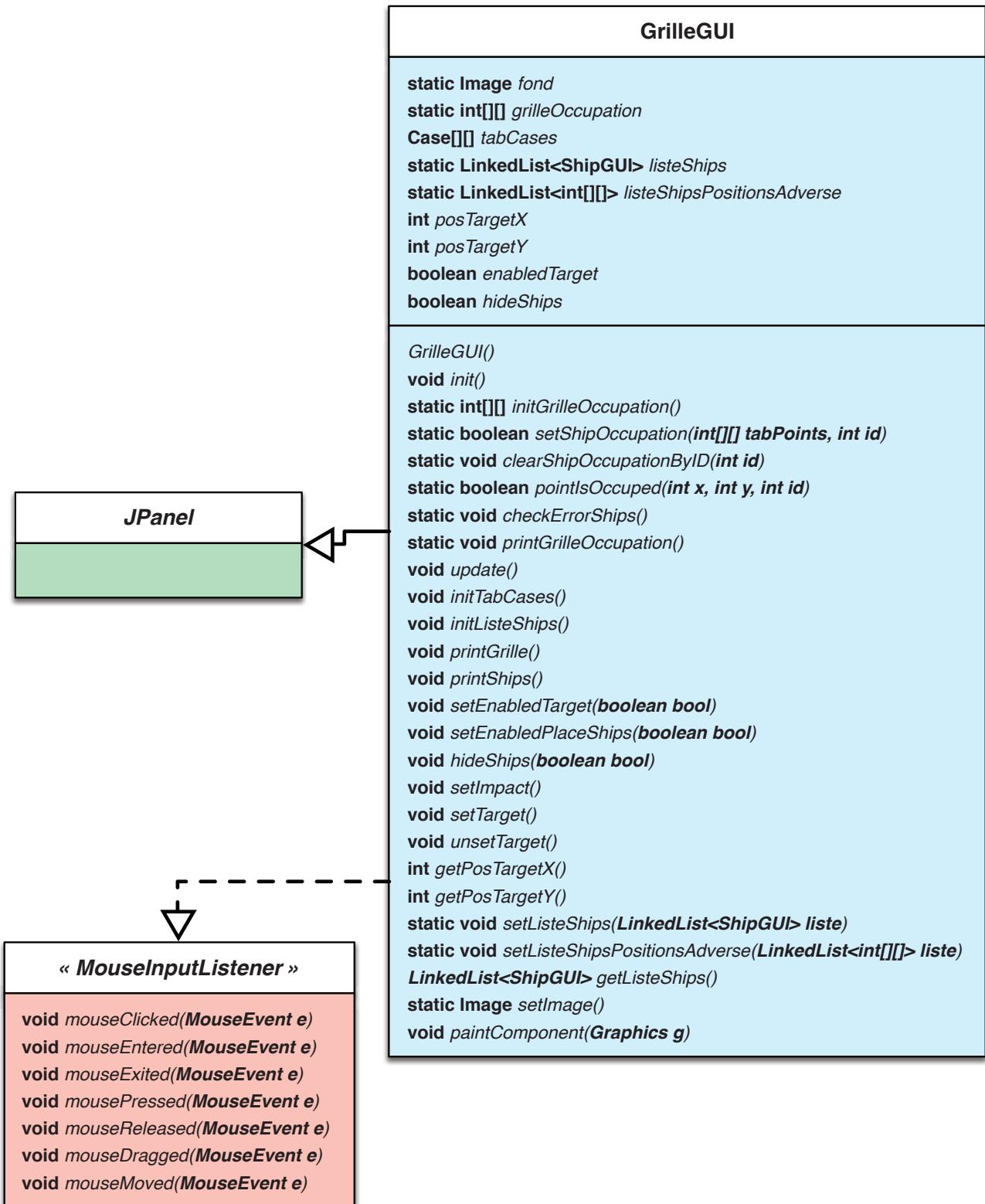


Diagramme UML

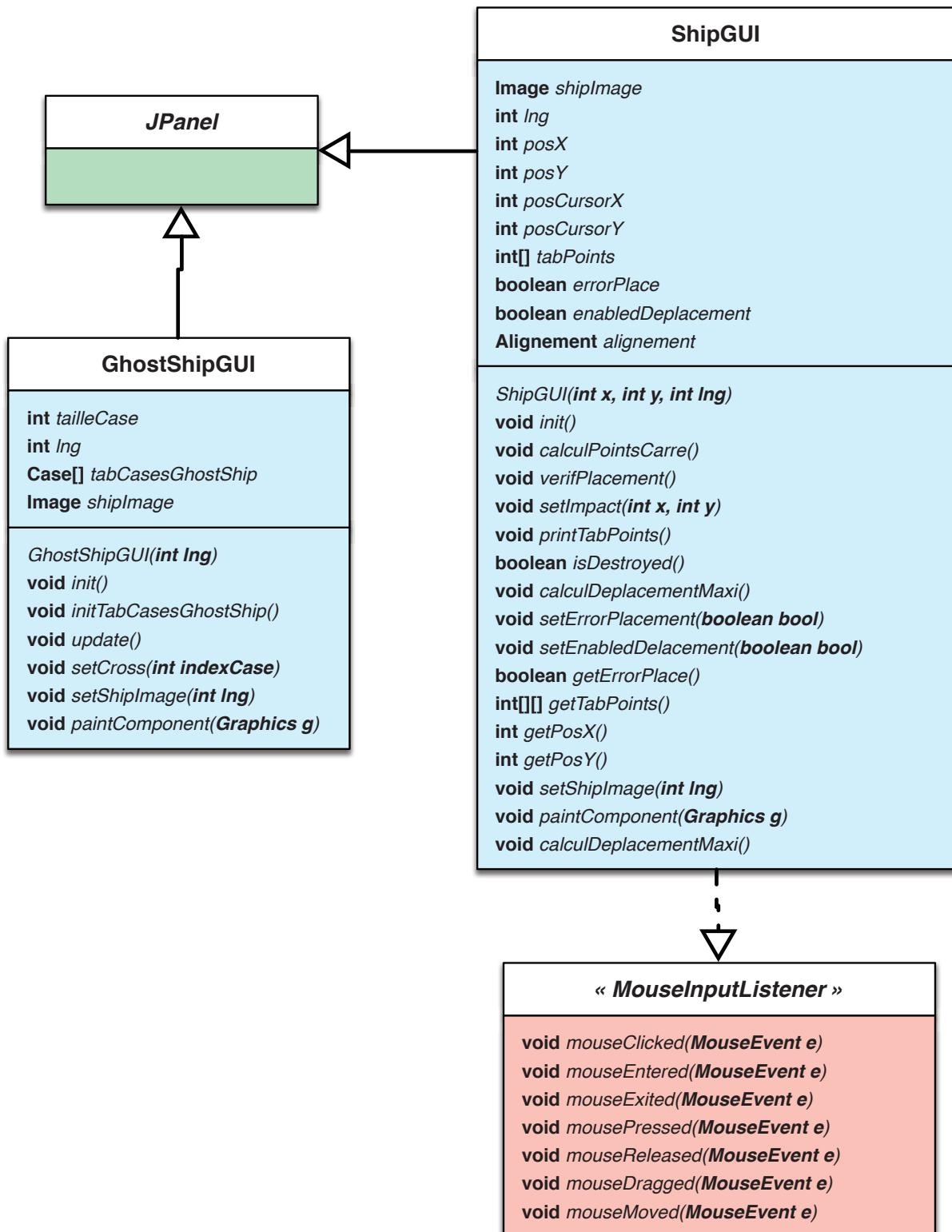


Diagramme UML

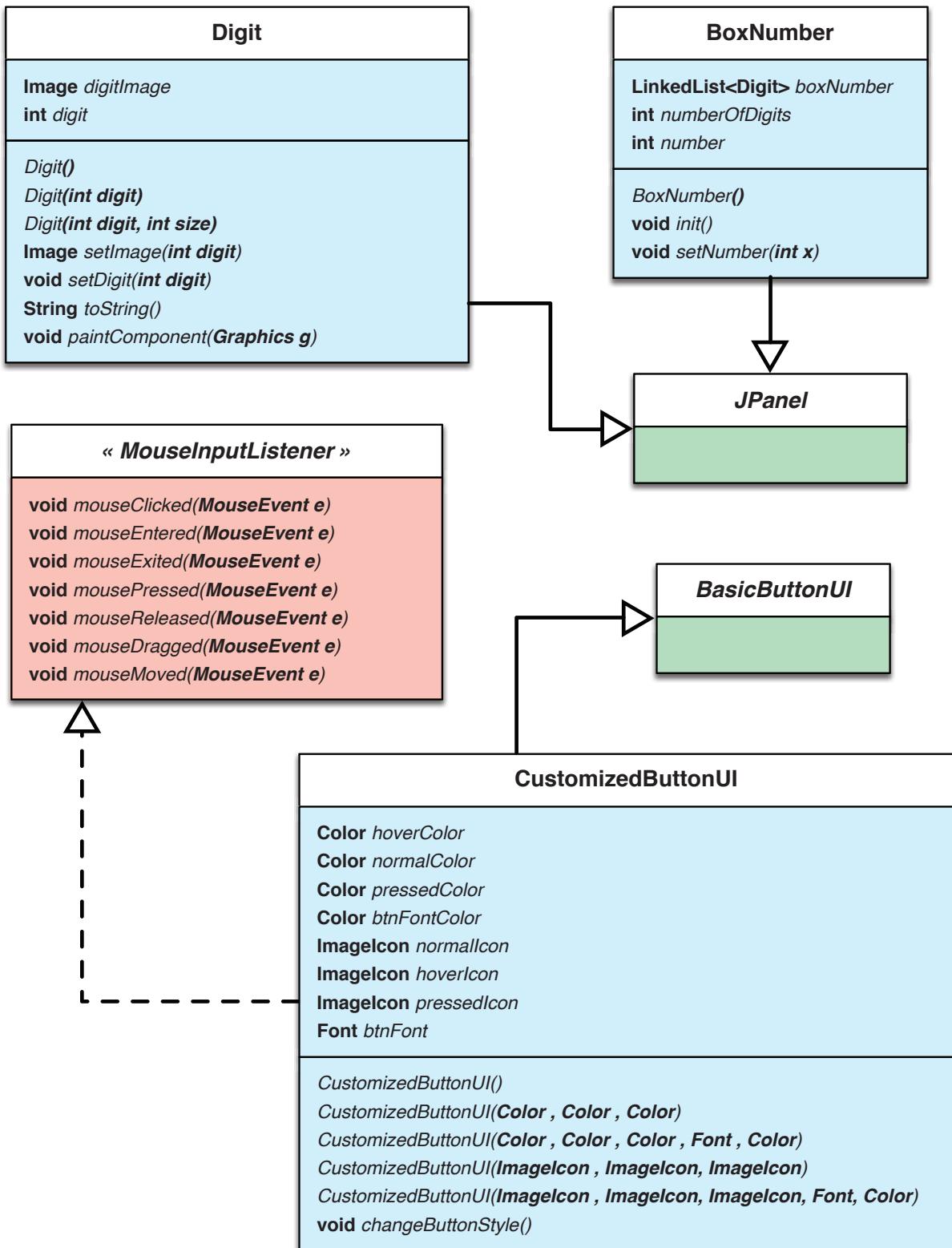


Diagramme UML

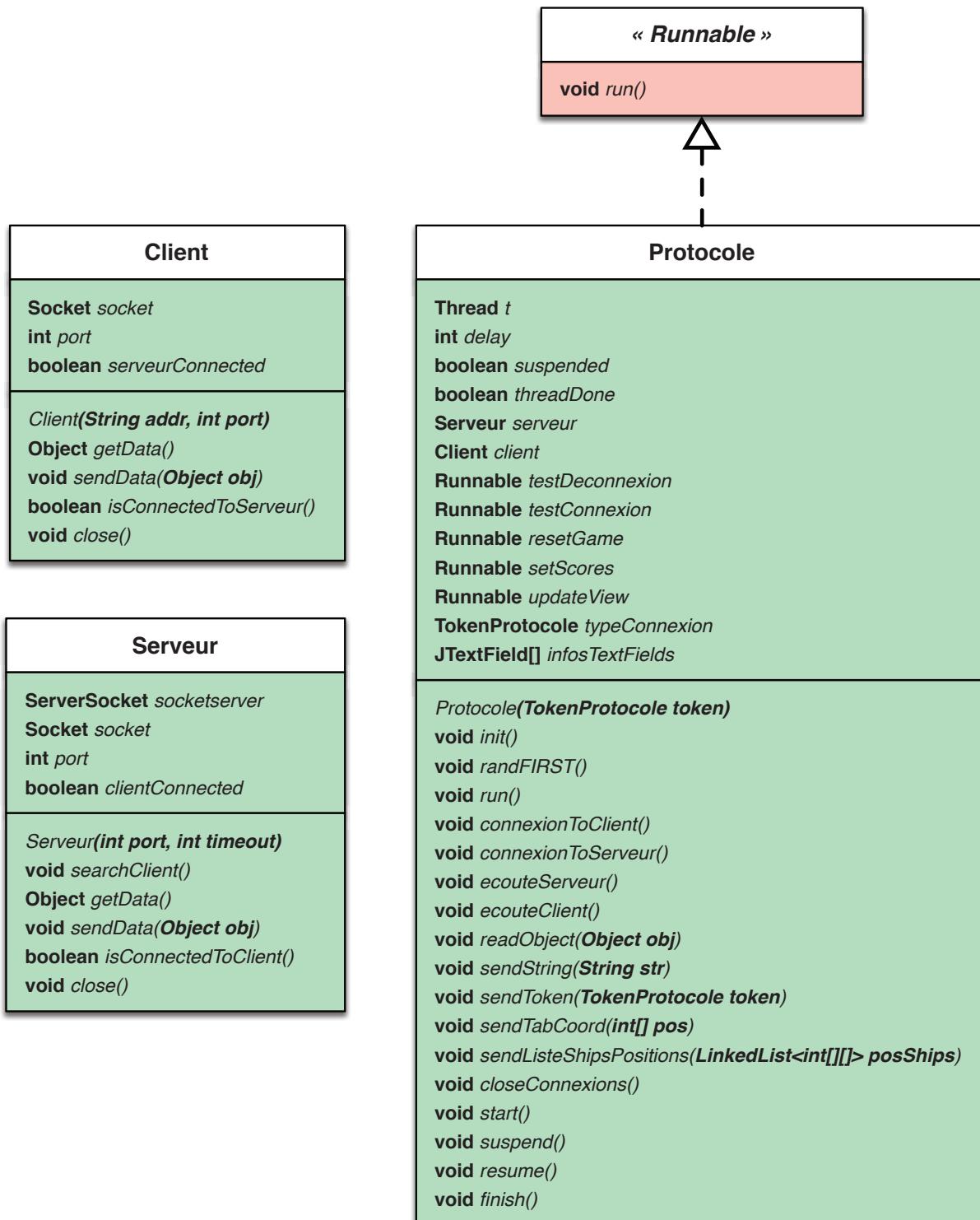


Diagramme UML

AudioPlayer	Sounds
AudioInputStream <i>audioInputStream</i> InputStream <i>file</i> FloatControl <i>gainControl</i> Clip <i>clip</i> boolean <i>loop</i>	static float <i>volumeMusic</i> static float <i>volumeSound</i> static boolean <i>mutedMusic</i> static AudioPlayer <i>music1</i>
AudioPlayer(InputStream file) AudioPlayer(InputStream file, boolean loop) void playSound() void stopSound() void setVolume(Float volume)	static AudioPlayer <i>initMusic()</i> static void <i>setVolumeMusic(int volume)</i> static void <i>setVolumeSound(int volume)</i> static float <i>getVolumeMusic()</i> static float <i>getVolumeSound()</i> static void <i>muteMusicIntro(boolean bool)</i> static boolean <i>musicIsMuted()</i> static void <i>eventMusicIntro()</i> static void <i>eventSoundShot()</i> static void <i>eventSoundReload()</i> static void <i>eventSoundButton()</i> static void <i>eventSoundButton2()</i> static void <i>eventSoundWood1()</i> static void <i>eventSoundWood2()</i> static void <i>eventSoundExplosion()</i> static void <i>eventSoundScores()</i> static void <i>eventSoundShotWater()</i> static void <i>eventSoundShipDoor()</i>