2017

Projet C++

JEUX DE L'ALLUMETTE

JIANG SHANGWEI / WANG JIANGLEI / TANG YUXIN / ZHA JIERUI

Document de Spécifications

Introduction

- **Introduction du projet**

  Le but du jeu est de soit ne pas prendre la dernière allumette, soit essayer de prendre la dernière allumette. Ce jeu se joue à deux. Les joueurs sont devant un certain nombre d'allumettes (qui peut varier d'une partie à l'autre). A chaque tour, il faut en enlever 1, 2 ou 3. Celui qui prend la dernière gagne ou perd suivant les versions.

- **Astuce du jeu**

  Si c'est vous qui commence, en respectant ce règle ci-dessous, vous serez sûrement gagné.

  Si celui qui prend la dernière perd, à votre tour, vous devez laisser un nombre d'allumettes correspondant à un multiple de 4 plus 1, c'est-à-dire : 1, 5, 9, 13, 17, 21, 25, 29, 33, ... Ainsi, le nombre d'allumettes diminuant, vous laisserez forcément la plus petite position possible, c'est-à-dire 1 allumette. L'autre aura perdu.

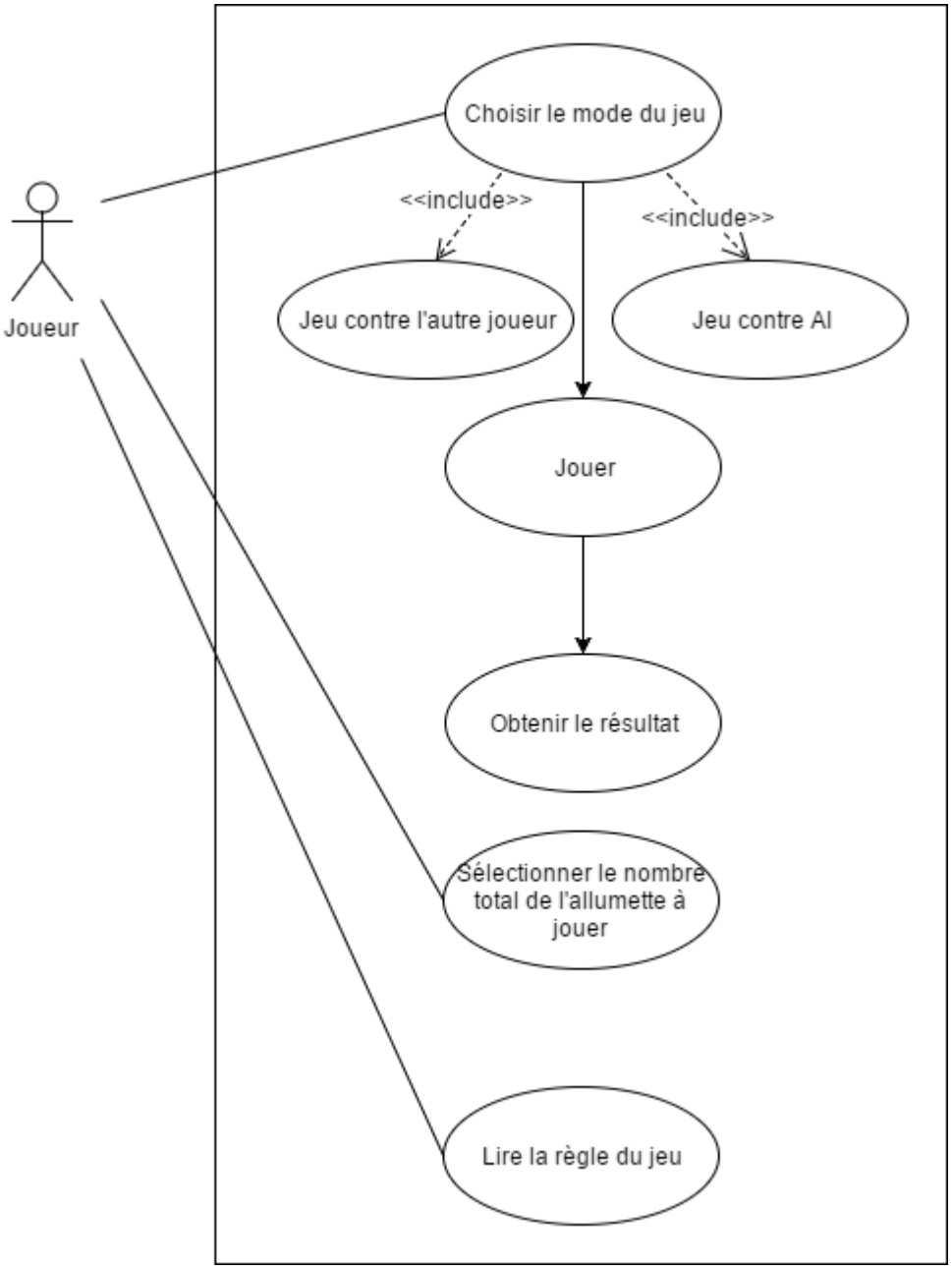
  Si celui qui prend la dernière gagne, à votre tour, laissez un nombre d'allumettes correspondant à un multiple de 4 : 4, 8, 12, 16, 20, ... Ainsi, vous êtes forcé de gagner.

  Que faire si l'autre joue comme vous joueriez ? Il y a deux possibilités : soit votre adversaire connaît aussi le truc. Dans ce cas, vous ne pouvez rien faire. Espérez une erreur de sa part. Soit votre adversaire ne le connaît pas. Dans ce cas, il a de la chance et joue au hasard (ou avec une mauvaise stratégie) mais bien. Toutefois, il y a peu de chances que cela dure et il finira par se tromper. Vous pouvez aussi forcer la chance en enlevant, tant qu'il gagne, une allumette à chaque tour. Ainsi, vous faites durer la partie et augmentez les chances d'erreur de sa part.

- **Réalisation dans notre projet**

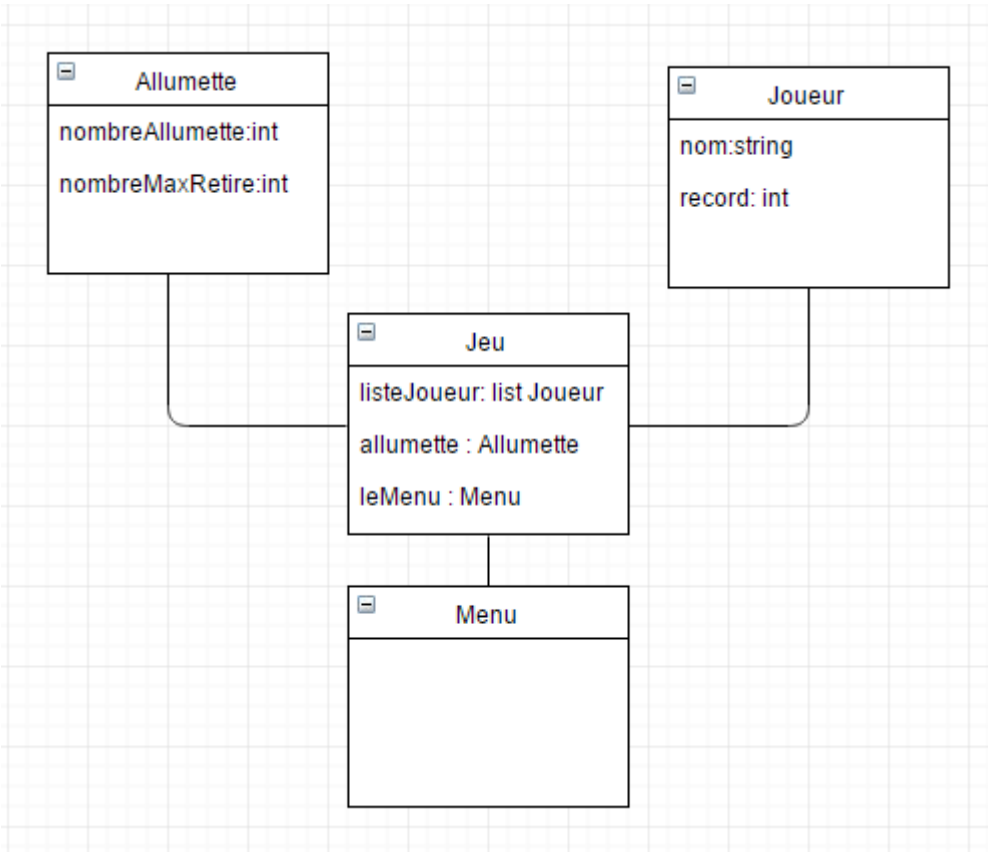
  Nous choisissons la version celui qui prend la dernière allumette perdue. Nous avons l'intention de faire deux genres du jeu : humain contre humain (en changeant de la place sur la même machine) et humain contre machine. Pour la partie machine, nous allons appliquer l'astuce introduit ci-dessus. Nous fixons certain nombre d'allumettes aux certains niveaux, plus nombreuse correspond plus haut niveau comme il complique le calcul pour humain.

Diagramme de cas d'utilisation



Le diagramme de cas d'utilisation représente les cas d'utilisation du joueur. Le joueur pourrait choisir le mode du jeu qui inclue choisir de jouer contre l'autre joueur et de jouer contre la machine. Et puis le joueur pourrait commencer le jeu et obtenir le résultat après finir une partie du jeu. Il pourrait aussi sélectionner le nombre total de l'allumette à jouer et lire la règle du jeu.

Diagramme de modèle du domaine



Nous décidons de définir 4 classes principaux : Allumette, Joueur, Menu et Jeu.

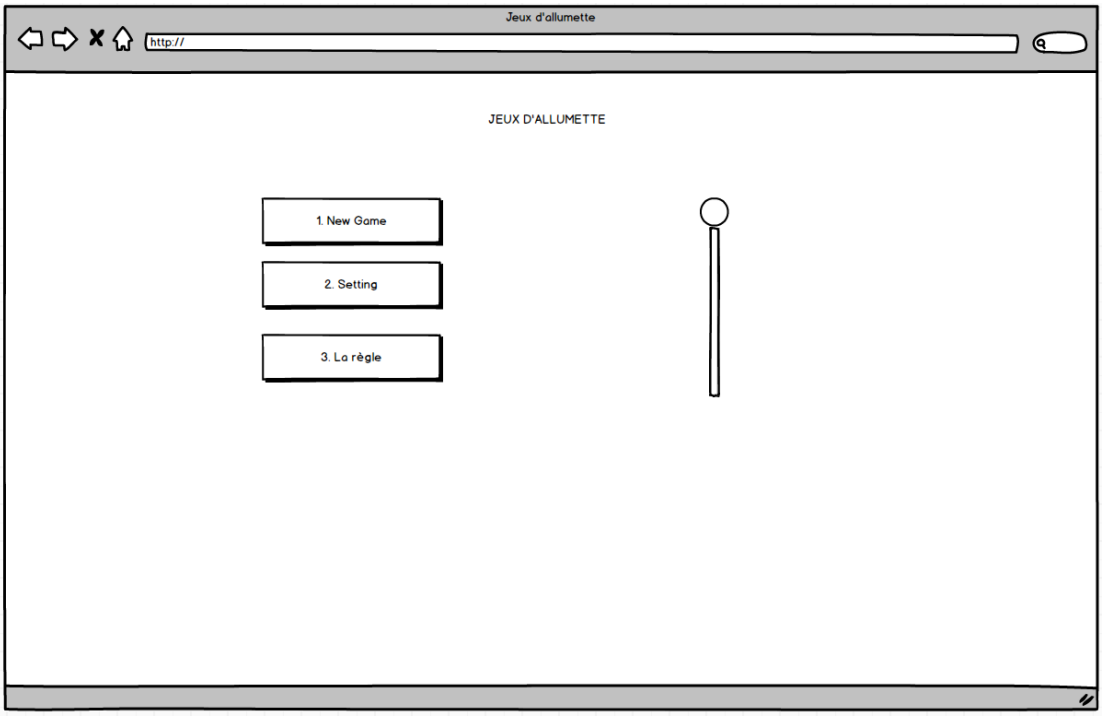
La classe Allumette représente les allumettes qui peut être retirer par les joueurs, et elle contient la méthode qui permet de réaliser l'affichage graphique.

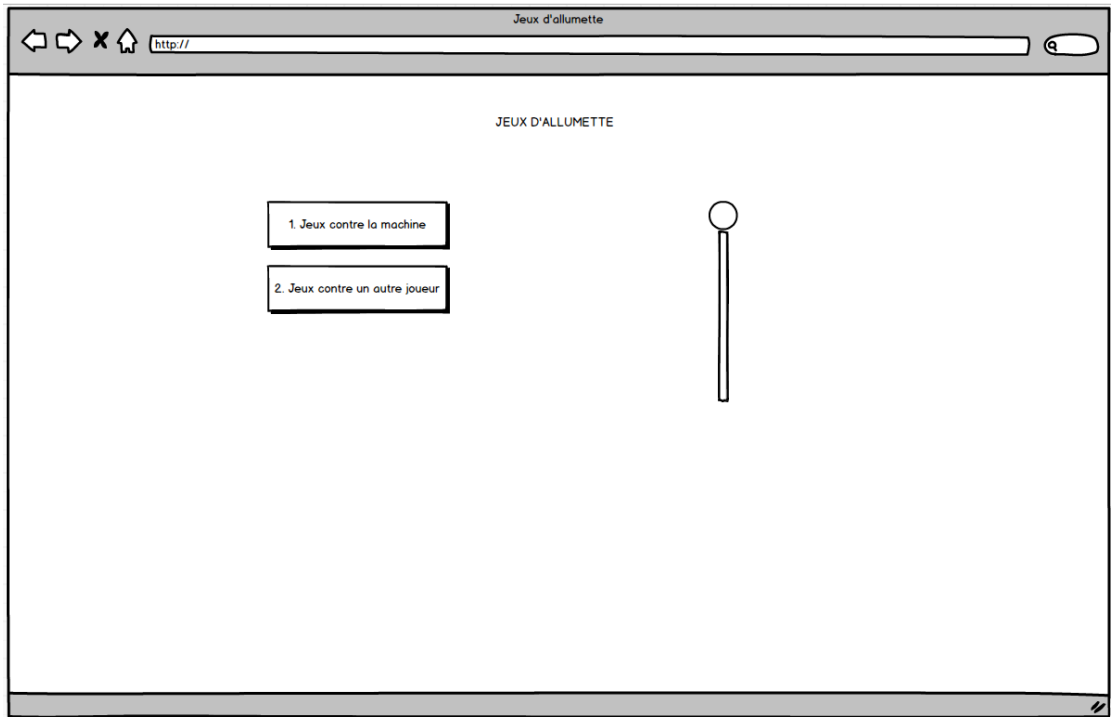
La classe Joueur représente un joueur qui participe au jeu , un joueur peut être un humain ou bien l'ordinateur . Nous enregistrons le nom du joueur et son record. Une fois que le joueur a gagné une partie du jeu , un point est ajouté à son record .

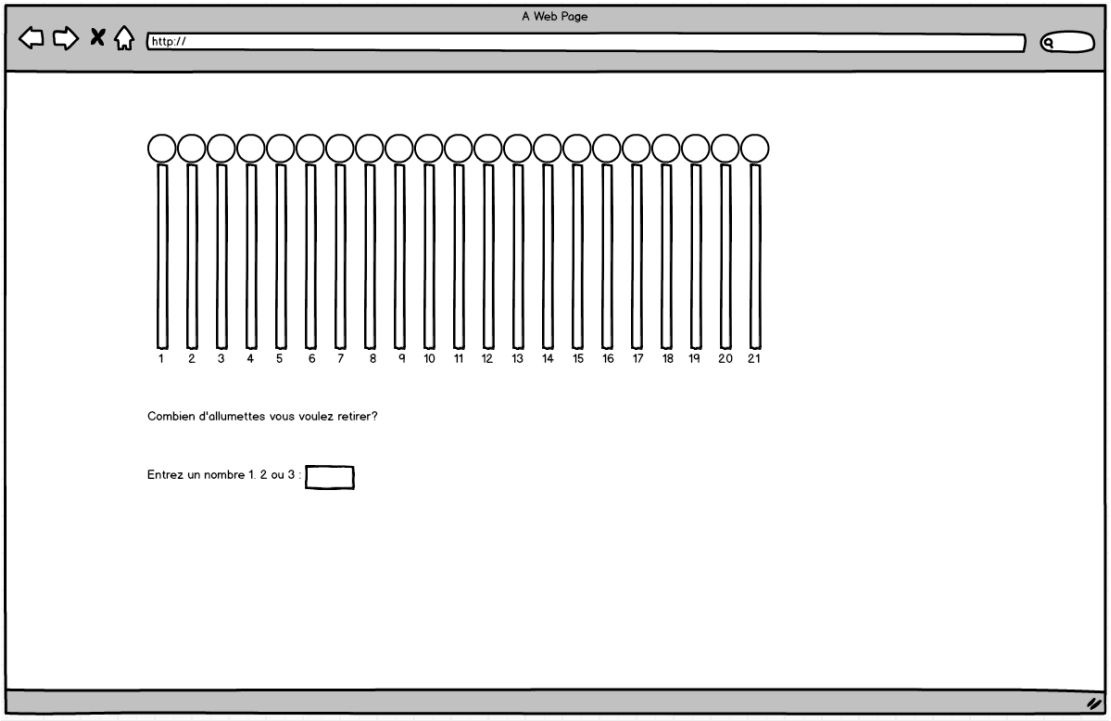
La classe Menu représente le menu avant le démarrage du jeu. Elle a une interface simple affichée dans le terminal. Nous pouvons faire notre option dans le menu : démarrer une partie du jeu , voir les règles et changer les paramètres du jeu .

La classe Jeu consiste le plus essentiel logique de ce projet, il réalise le processus principal du jeu. Il y a une liste de joueur dans l'attribue de la classe, la liste enregistre tous les joueurs qui ont participé au jeu. Les méthodes dans cette classe nous permettent d'initialiser et jouer une partie du jeu.

Maquettes

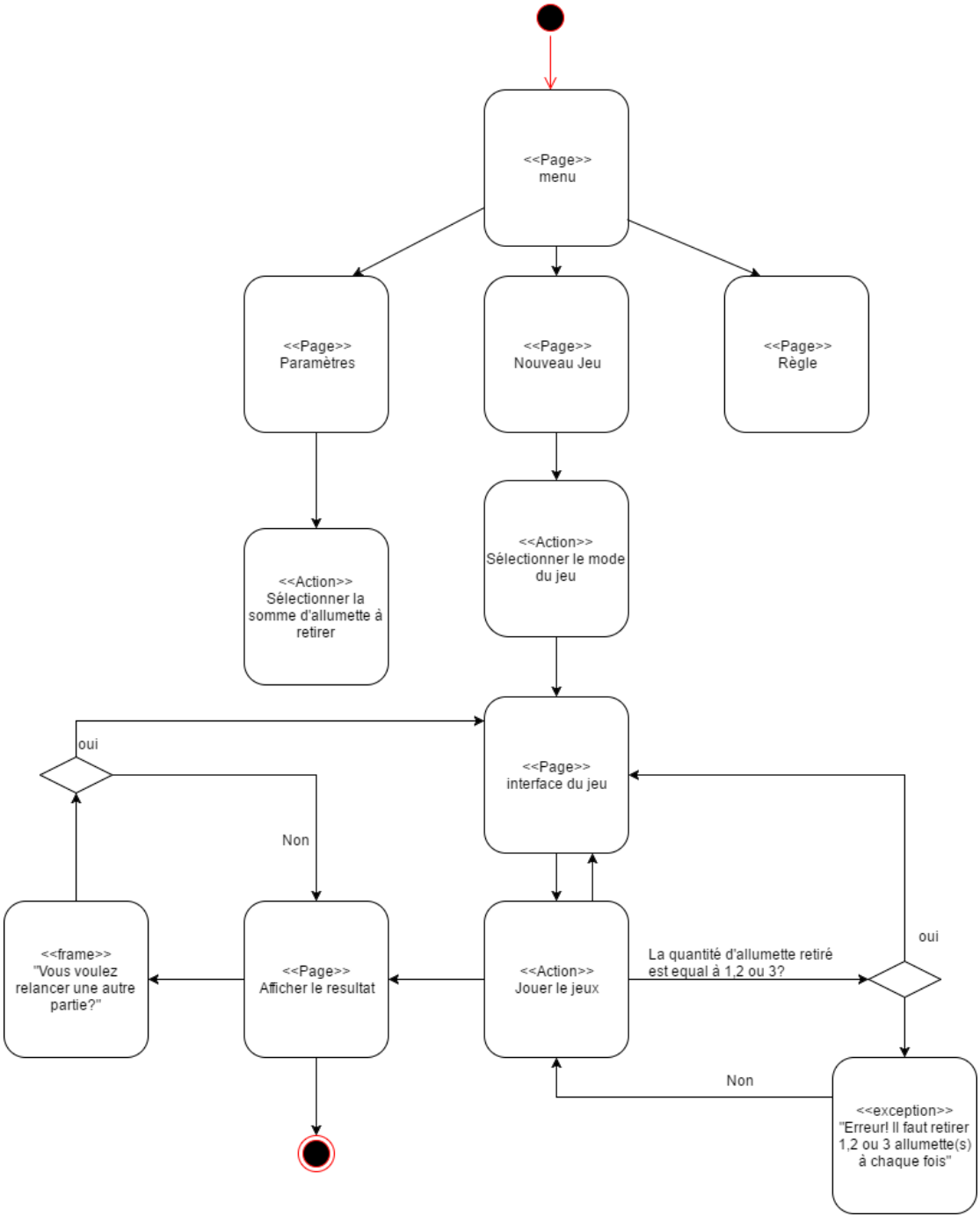

Jeux d'allumette
http://
JEUX D'ALLUMETTE
1. New Game
2. Setting
3. La règle


Jeux d'allumette
http://
JEUX D'ALLUMETTE
1. Jeux contre la machine
2. Jeux contre un autre joueur



Le premier graphe représente le menu principal du jeu.
Le deuxième graphe représente l'interface de sélectionner le mode du jeu.
Le dernier graphe représente l'interface du jeu.
Les maquettes au-dessus sont des graphes concepts. On ne fera pas les interfaces graphiques concrètement. On utilisera les symboles pour représenter l'interface du jeu.

Diagramme de navigation



Le diagramme de navigation représente le déroulement du programme. Le début du jeu est la page « menu ». Dans cette page, il y a 3 options : « Nouveau Jeu », « Règle » et « Paramètres ». En choisissant « Paramètres », le joueur pourrait sélectionner le nombre total de l'allumette à jouer ; « Règle » permet de lire la règle du jeu et en fin, « Nouveau Jeu » pour commencer un le jeu. Après choisir « Nouveau Jeu », le jeu demandera les joueurs à sélectionner le mode du jeu : soit contre avec un autre joueur, soit contre la machine. Et puis on arrive à l'interface du jeu et pourrait commencer jouer. Pendant le processus de retirer les

allumettes, si le nombre de l'allumette que le joueur prenne n'est pas égal à 1,2 ou 3, le programme retournera une exception « Erreur ! Il faut retirer 1,2 ou 3 allumette(s) à chaque fois » pour rappeler les joueurs et ils devront alors ré-retirer les allumettes. Quand il ressort de gagnant, le score des joueurs s'affichera et le programme eux demandera si ils veuillent relancer une autre partie.

Document de Conception

Introduction

- **Introduction du projet**

Le but du jeu est de soit ne pas prendre la dernière allumette, soit essayer de prendre la dernière allumette. Ce jeu se joue à deux. Les joueurs sont devant un certain nombre d'allumettes (qui peut varier d'une partie à l'autre). A chaque tour, il faut en enlever 1, 2 ou 3. Celui qui prend la dernière gagne ou perd suivant les versions.

- **Astuce du jeu**

Si c'est vous qui commence, en respectant ce règle ci-dessous, vous serez sûrement gagné.

Si celui qui prend la dernière perd, à votre tour, vous devez laisser un nombre d'allumettes correspondant à un multiple de 4 plus 1, c'est-à-dire : 1, 5, 9, 13, 17, 21, 25, 29, 33, ... Ainsi, le nombre d'allumettes diminuant, vous laisserez forcément la plus petite position possible, c'est-à-dire 1 allumette. L'autre aura perdu.

Si celui qui prend la dernière gagne, à votre tour, laissez un nombre d'allumettes correspondant à un multiple de 4 : 4, 8, 12, 16, 20, ... Ainsi, vous êtes forcé de gagner.

Que faire si l'autre joue comme vous joueriez ? Il y a deux possibilités : soit votre adversaire connaît aussi le truc. Dans ce cas, vous ne pouvez rien faire. Espérez une erreur de sa part. Soit votre adversaire ne le connaît pas. Dans ce cas, il a de la chance et joue au hasard (ou avec une mauvaise stratégie) mais bien. Toutefois, il y a peu de chances que cela dure et il finira par se tromper. Vous pouvez aussi forcer la chance en enlevant, tant qu'il gagne, une allumette à chaque tour. Ainsi, vous faites durer la partie et augmentez les chances d'erreur de sa part.

- **Réalisation dans notre projet**

Nous choisissons la version celui qui prend la dernière allumette perdue. Nous avons l'intention de faire deux genres du jeu : humain contre humain (en changeant de la place sur la même machine) et humain contre machine. Pour la partie machine, nous allons appliquer l'astuce introduit ci-dessus. Nous fixons certain nombre d'allumettes aux certains niveaux, plus nombreuse correspond plus haut niveau comme il complique le calcul pour humain.

<u>Fonctionnalité</u>

Dans ce jeu, on va d'abord entrer dans la page « menu », où on peut sélectionner de :

1. Jouer un nouveau jeu
   → sélectionner une mode : simplejoueur / doublejoueur
   → commencer à jouer
   →relancer un nouveau jeu dans la même mode avec la même paramètre

2. Changer les paramètres
3. Lire la règle du jeu

explication en détaille

Si on connait pas encore la règle de ce jeu, tape 3 et puis entrer pour lire la règle spécifique et il y a aussi quelque information pratique sur notre jeu.

Pour changer les paramètres, tape 2. Ici, le paramètre à changer est le nombre d'allumette, on peut saisir n'importe combien d'allumette qu'on voudra. Par défaut, il y aura 21 dans le jeu.

Une fois qu'on commence le jeu, on entre d'abord à sélectionner de jouer avec une autre personne (doublejoueur) ou avec la machine (simplejoueur).

Si on joue avec une autre personne, il faut les deux personnes soient actuellement devant la même machine et ils décident qui commence sur place et puis ils saisient leur nom l'un après l'autre. Une fois tout ça c'est fait, ils peuvent commencer jouer (tirer les allumettes) tour à tour.

Si on joue avec la machine, l'ordre est fixé, c'est le joueur qui commence le jeu. Il saisie son nom et puis directement joue tour à tour avec la machine.

A la fin de chaque partie du jeu, le résultat sera affiché sur l'écran disant qui est gagné et combien de point chacun est accomulé. Il s'agit de garder le résultat de chaque partie, c'est à dire il compte combien de parties que vous avez gagné. Une partie gagnée égale à un point. Mais il n'enregistre rien si on quitte le jeu.

Après il demande si on veut continuer dans cette mode et jouer une autre fois, sinon on quitte le jeu. Il ne supporte pas de modifier la mode ou changer le nombre d'allumette après lancer la première fois de jeu.

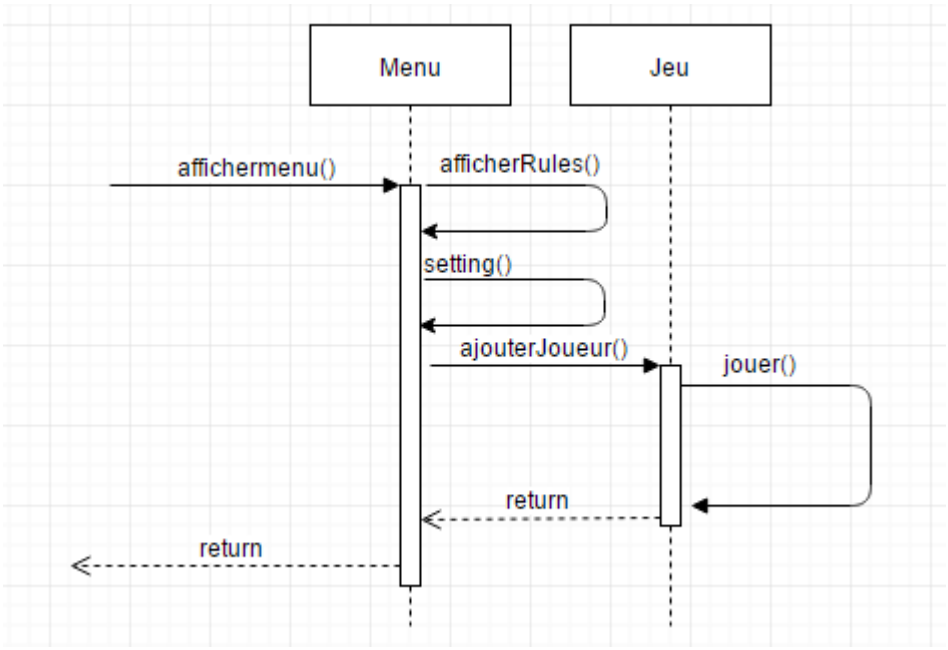
- **Les contraintes matérielles et logiciels**

Le logiciel qu'on va utiliser est eclipse c++ version 1.7 qui ne supporte pas de créer une interface et de jouer sur réseau. Donc notre jeu sera joué dans le console d'eclipse, et le joueur joue en saisissant le chiffre et appuyant sur Entrer pour confirmer et changer de tour.

Conception préliminaire

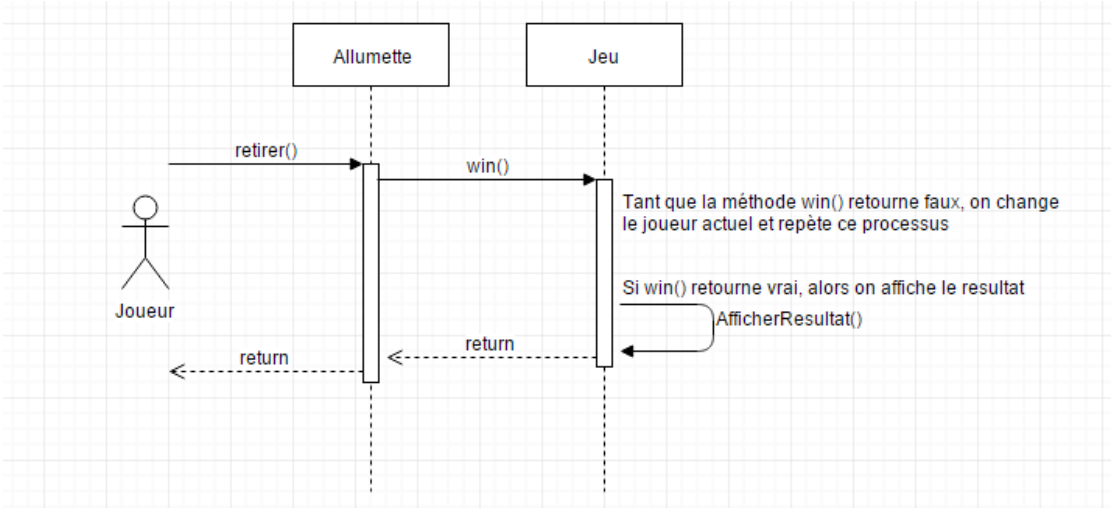
Les diagrammes de séquence système

Les interactions de classes apparaissent quand on démarre le jeu , et quand on joue une partie du jeu .



Démarrer le jeu

L'interface menu permet d'afficher 3 options : une pour afficher les règles du jeu, une pour changer les paramètres du jeu, et l'autre pour démarrer le jeu. Il existe deux modes de jeu : soit jouer avec l'ordinateur, soit jouer avec un autre joueur. Si le premier mode est choisi, une instance de joueur et une instance de machine sont créés et ajoutées dans le jeu. Dans le deuxième cas, deux joueurs sont créés. La méthode jouer est invoquée ensuite.



Jouer

Si un joueur retire un certain nombre de l'allumette, la méthode win() est appelée afin de déterminer si le joueur a perdu. Si oui, le résultat du jeu sera affiché. Sinon l'autre joueur va reprendre la main pour continuer le jeu.

*

- Le diagramme de modèle du domaine
- Des maquettes
- Le diagramme d'activités de navigation

cf. le document spécification

Conception détaillée

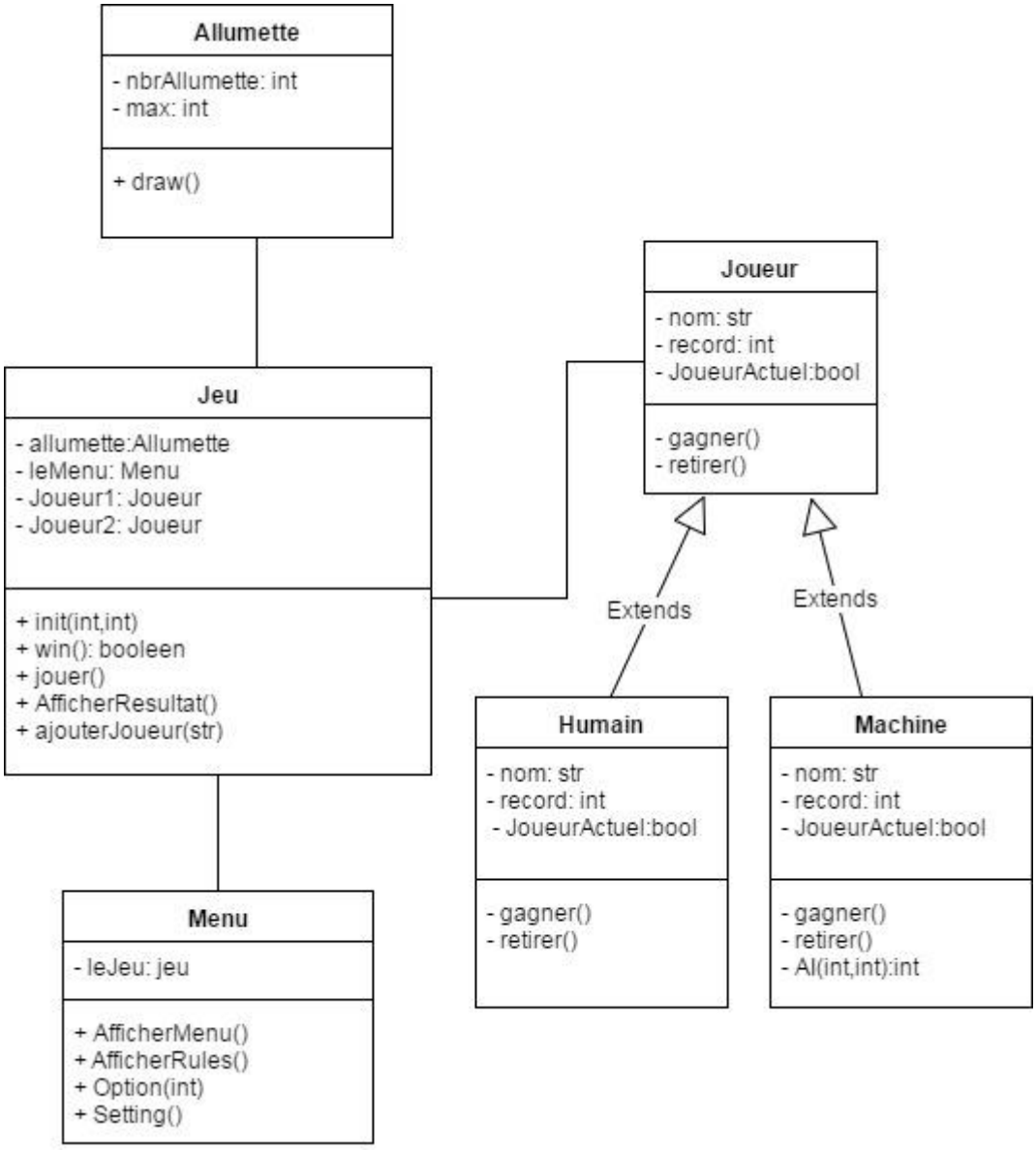
Diagramme de classe

Comparé avec le modèle du domaine, le diagramme de classe est plus précis, et afin de simplifier le travail de la programmation, nous avons fait quelques changements :

-Dans la classe de Jeu, nous avons supprimé la liste de joueurs et la remplaçons par Joueur1 et Joueur2, car nous ne sommes pas très intéressant à savoir tous les joueurs qui ont participé au jeu mais juste les deux joueurs qui jouent.

-Nous avons considéré la classe Joueur comme une classe abstraite et nous avons ajouté deux classes Humain et Machine qui héritent la classe Joueur pour bien distinguer les cas différents.

Le diagramme de classe final est comme la suite :



- **Allumette:**

Cette classe contient deux attributs entiers : **nbrAllumette** et **max**. Le **nbrAllumette** représente le nombre total de l'allumette à jouer à chaque partie du jeu. Le **max** représente le nombre maximum de l'allumette peut être retiré à chaque fois.

Il y a une méthode dans cette classe : **draw()**. La méthode **draw()** permet de afficher une interface graphique simple composée par les caractères dans le terminal .

- **Joueur** :

Joueur est une classe abstraite qui contient trois attributs : un string : **nom**, un entier : **record** et un booléen : **JoueurActuel**. Le **nom** est une variable qui représente le nom du joueur, **record** indique le score d'un joueur et **JoueurActuel** représente si le joueur soit dans son tour ou pas.

Il y a deux méthode dans cette classe abstraite : **gagner()** et **retirer()**. La méthode **gagner()** réalise de rajouter d'un point pour chaque fois qu'un joueur a gagné une partie. La méthode **retirer()** est un processus qui permet au joueur de faire le tirage des allumettes dans chaque tour.

Humain et **Machine** sont deux classes héritent la classe **Joueur**, ils représentent le joueur actuel et le joueur d'ordinateur respectivement. Dans la classe **Machine**, il y a une méthode qui s'appelle **AI(int, int) :int**. Cette fonction calcule le nombre de l'allumette à retirer dans le tour de l'ordinateur.

- **Menu** :

La fonction de cette classe est de créer un menu du jeu. Il contient un attribut **leJeu** du type jeu. Il y a quatre méthodes : **AfficherMenu()**, **AfficherRules()**, **Option(int)** et **Setting()**. La méthode **AfficherMenu()** crée un menu du jeu et l'affiche sur écran , elle aussi nous demande de faire une option: soit démarrer le jeu , soit voir les règles du jeu ou faire changement des paramètres du jeu (c'est-à-dire changer le nombre max d'allumette ou le nombre d'allumette qu'on peut retirer). Si on choisit à démarrer le jeu , le programme nous demande de choisir un mode : 1. mode humain vs humain ou 2. mode humain vs machine. Puis cette décision qu'on a fait est prise en paramètre par la méthode **Option(int)** , et après les Créations de joueurs , une partie du jeu commence. La méthode **AfficherRules()** contient les règles du jeu et permet de l'affiche sur écran . La méthode **Setting()** permet de faire les changements de paramètres du jeu .

- **Jeu** :

La classe Jeu consiste le plus essentiel logique de ce projet, il réalise le processus principal du jeu. Il contient quatre attributs : **allumette** du type Allumette, **leMenu** du type Menu et deux variables **Joueur1** et **Joueur2** du type Joueur respectivement.

D'abords, on doit initialiser le jeu par la méthode **init(int,int)**. Puis on pourrait ajouter un joueur dans le jeu en appelant la méthode **ajouterJoueur(str)**.

Et la méthode **win()** qui sort un booléen pour déterminer si l'un des joueurs soit gagné ou pas. La méthode **jouer()** permet de réaliser le processus d'un joueur contre l'autre. Après avoir fini une partie du jeu, les joueurs pourront voir son point par la méthode **AfficherResultat().**

Pseudo-code pour les méthodes principaux

Dans la classe Jeu:

```
Jouer()
Début
    int nb
    tant que nbrAllumette>0
        allumette.draw()
        si(J2.getactuel()) alors
            si (J2=machine)
                alors J2.retirer(J2.calculer(allumette.getnb(),allumette.getmax()))
            sinon
                écrire('Vous voulez retirer combien d allumette ?')
                lire(nb)
                J2.retirer(nb)
            finsi
            si win() alors
                J1.gagner()
                érire(J2.nom' avez perdu...' )
            sinon
                écrire('Il reste ', nbrAllumette,'  allumettes')
                J1.setactuel(vrai)
                J2.setactuel(faux)
            finsi
        sinon
            écrire('Vous voulez retirer combien d allumette ?')
            lire(nb)
            J1.retirer(nb)
            si win()
                érire(J1.nom' avez perdu...' )
                J2.gagner()
            sinon
                écrire('Il reste ', nbrAllumette,'  allumettes')
                J2.setactuel(vrai)
                J1.setactuel(faux)
            finsi
        finsi
    fintantque
Fin
```

Cette méthode permet de jouer une partie du jeu. D'abord, on décide qui est le joueur actuel. S'il est une machine, on retire un nombre calculé d'allumettes par la méthode AI(); sinon on demande au joueur combien d'allumette il veut retirer et changer le joueur actuel, jusqu'à toutes les allumettes sont retirés.

Dans la classe Menu:

AfficherMenu()
Afficher le menu sur l'écran et lire la commandement du joueur. Appeller la méthode AfficherRules ou Option ou encore Affichermenu selons la valeur retourné par le joueur.

AfficherRules()
Afficher les rules sur l'écran. Retourner au menu après les joueurs finissent de lire les rules par entrer un autre commandement.

```
Option(c)
Début
    Bool rejouer=vrai
    Si (c ==1)
        Alors    leJeu.AjouterJoueur (machine)
                 leJeu.AjouterJoueur()
    Sinon si (c==2)
        Alors    leJeu.AjouterJoueur()
                 leJeu.AjouterJoueur()
    Finsi
    Tantque rejouer==vrai
        leJeu.Jouer()
        leJeu.AfficherRésultat()
        Ecrire('Vous voulez rejouer?Y/N ')
        Lire(command)
        Si command==N
            Alors rejouer=faux
        Finsi
    Fintantque
Fin
```

Quand on lance le programme, d'abord, on applique la méthode AfficherMenu(). Après, on peut voir les règles ou changer le nombre d'allumette ou lancer le jeu selons le choix du joueur. Si on lance le jeu, il faut choisir le mode du jeu : soit on joue par deux personnes, on exécute Option(2); soit on joue contre l'ordinateur, on exécute Option(1). Ensuite on appelle la méthode jouer(). Lorsqu'une partie est fini, le programme va montrer le résultat. On peut choisir de jouer une autre fois par taper Y.

Dans la classe Machine:

```
AI(int reste, int max):int
Début
    dividende = max + 1
    si (reste mod dividende = 1)
        alors retourner 1
    sinon si (reste mod dividende = 0)
            alors retourner max
    sinon
        retourner (reste mod dividende - 1)
    finsi
Fin
```

La méthode AI est conçue pour réaliser l'action de l'ordinateur. Elle prend deux intègres en entrée, **reste** représente le nombre d'allumettes restantes du jeu et **max** représente le nombre d'allumettes au maximum que nous pouvons prendre chaque tour. La valeur retournée est la décision de l'ordinateur qui correspond à la nombre d'allumette qu'il veut prendre.
Comme indiqué précédemment, il existe un astuce du jeu. Si le max égale à trois, nous devons laisser un nombre d'allumette correspondant à un multiple de 4 plus 1. Dans la méthode, nous retournons un nombre qui permet de réaliser l'astuce. Mais il existe un cas exceptionnel, c'est-à-dire à la tour de l'ordinateur, il reste exactement un multiple de 4 plus 1, dans ce cas nous retournons simplement 1.