GAMEPLAY STUDIO

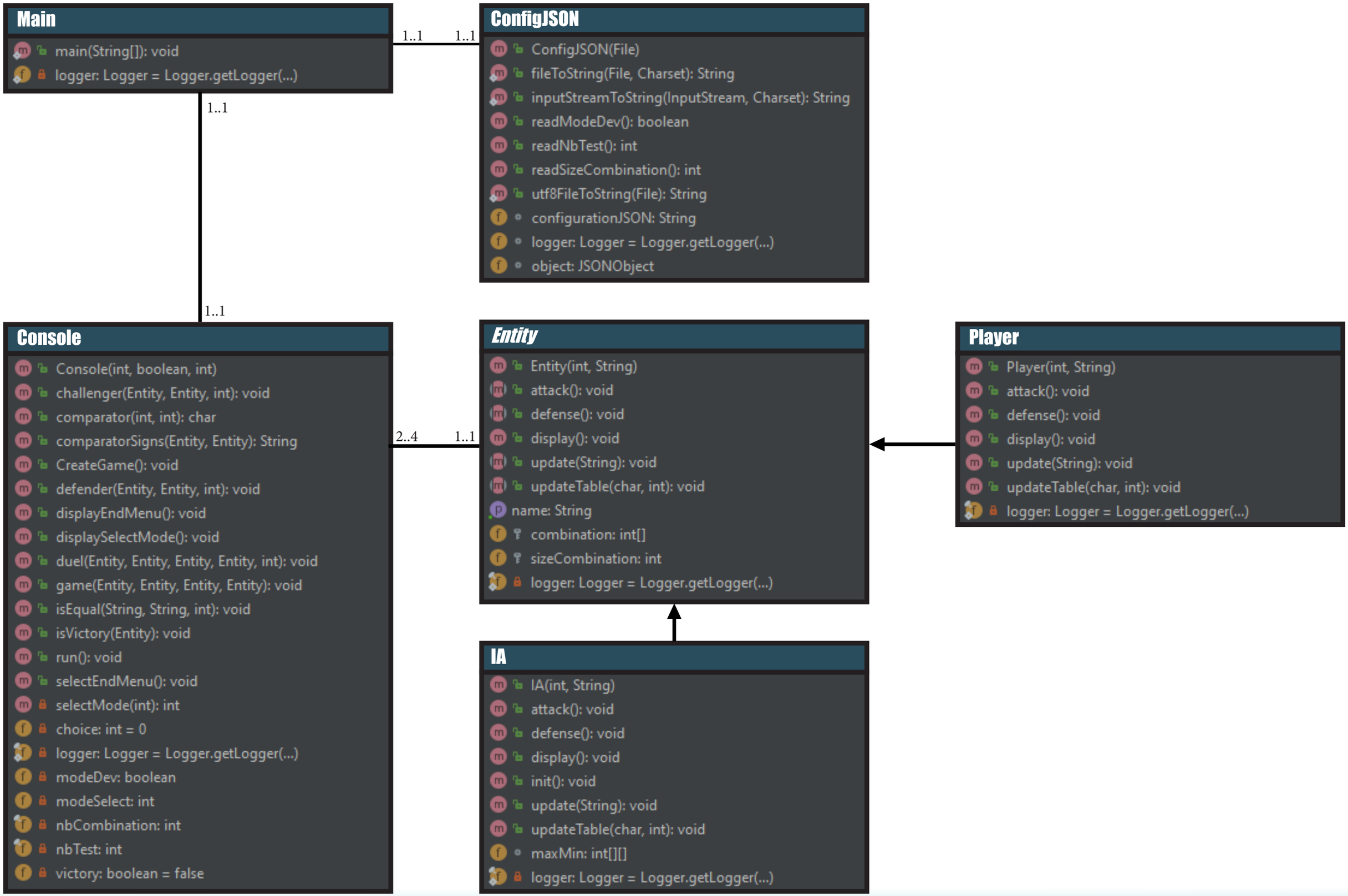
Projet : Escape Game ONLINE

« Création d'un mécanisme de recherche d'une combinaison à X chiffres »

Dubois Geoffrey

OPENCLASSROOMS

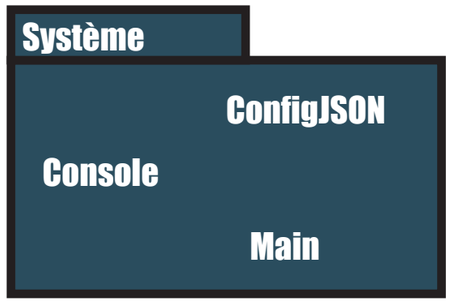
Diagramme de classe



Choix d'implémentation

2 packages :

- Acteurs (Entity) → Player / IA
- Système → Console / Main / ConfigJSON



Nommage des méthodes et des attributs :

- Anglais
- Explicite

```
public Console(int nbCombination, boolean modeDev, int nbTest) {
    this.nbCombination = nbCombination;
    this.modeDev = modeDev;
    this.nbTest = nbTest;
    logger.trace("Instantiation of an object Console ");
    logger.debug("nbCombinaison = "+nbCombination+" ,nbtest = "+nbTest+" ,modeDev = "+modeDev);
}
```

Méthodes minimaliste:

- Modularité du code
- Compréhension plus simple

```
public void run() {
    logger.trace("Input procedure Run");
    do {
        CreateGame();
        selectEndMenu();
    } while (choice != 0);
    logger.trace("Output procedure Run");
}
```

Algorithmes

```java
public void defender(Entity player1, Entity player2, int index) {
    String result;
    if (index == 0)
        player1.defense();
    player2.attack();
    player2.display();
    result = comparatorSigns(player2, player1);
    player2.update(result);
    System.out.println(result);
    isEqual(result,player2.getName(), index);
}


public String comparatorSigns(Entity player1, Entity player2) {
    char signs;
    String result = "";
    for (int i = 0; i < nbCombination; i++) {
        signs = comparator(player2.combination[i], player1.combination[i]);
        result += signs;
    }
    return result;
}


public char comparator(int num1, int num2) {
    if (num1 > num2)
        return '+';
    else if (num2 > num1)
        return '-';
    else
        return '=';
}


public void updateTable(char signs, int index) {
    if (signs == '+')
        maxMin[index][1] = combination[index] + 1;
    if (signs == '-')
        maxMin[index][0] = combination[index] - 1;
    if (signs == '=') {
        maxMin[index][1] = combination[index];
        maxMin[index][0] = combination[index];
    }
}
```

Defender / Challenger / Duel

if(index == 0)

if(modeDev)

Duel = defender + challenger

Suite de fonction

ComparatorSigns / comparator

Boucle de 0 à taille de la combinaison

comparaison de chiffres, return char

concatenation de signs dans result

IA.UpdateTable

IA.maxMin[ ] [ ]

Mise à jour

IA analyse le résultat et réduit la marge d'erreur

Axe d'amélioration

IA choisi :

```
Float middle = maxMin[index][1]+(maxMin[index][0]-maxMin[index][1]);
middle = middle / 2;
int value = middle.intValue();
```

Borné plus rapidement

Configuration JSON

Fichier de configuration:

Lecture simple

Modification des paramètres rapide

Fichier court

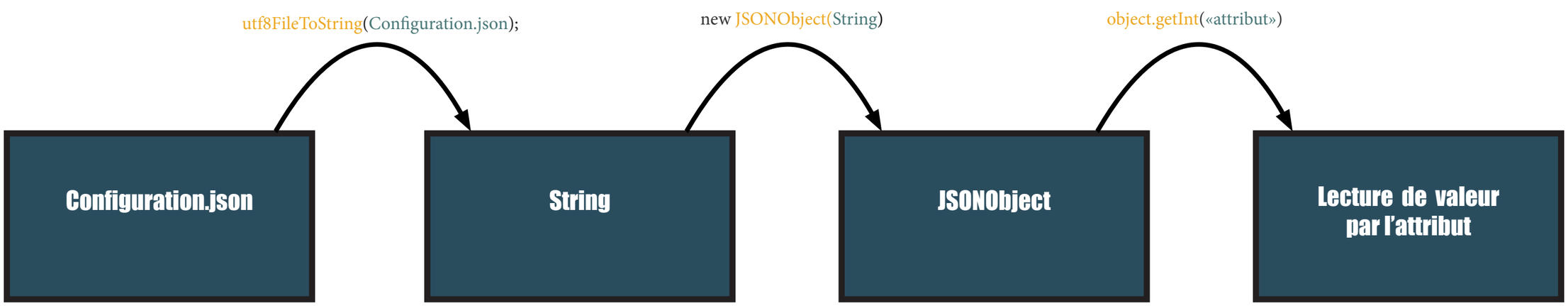
```
{
  "sizeCombination" : 8,
  "nbTest" : 5,
  "modeDev" : "false"
}
```

Classe ConfigJSON

Lecture de fichier ⟶ String

Instanciation de la classe ⟶ récupération du String et transformation en JSONObject

Méthodes readAttributs ⟶ return la valeur de l'attribut



Input / Output

Gestion des erreurs des «input»

IOException : Ouverture de fichier

Erreurs dans le Scanner :

SelectMode → boucle avec compteur

selectEndMenu → sortie direct

conbinaison → boucle infinie

Génération des sorties «Output»

Message d'info / jeu → Console

Message de fonctionnement → Fichier de log

Fichier de propriété : log4j2-test.properties

Gestion des levels de log à sauvegarder

Type d'appender : RollingFile

Pattern d'émission : %d %p %C{1.} [%t] %m%n

Pattern de sauvegarde : test1-%d{MM-dd-yy-HH-mm}-%i.log

Évolution possible

```
if(modeDev)
    logger.setlevel(Level.TRACE)
```