Proyecto Kartel

# Objetivo Principal:

* Modelizar el juego de mesa Kartel para PC bajo Java

# Objetivos Específicos:

* Crear una estructura de clases que permitan modelizar el proyecto
* Utilizar java y la programación orientada a objetos (POO) para crear el juego
* Crear una inteligencia artificial para el juego

# Recursos usados:

* Lenguaje de programación: JAVA, versión 1.8.0\_241
* IDE de desarrollo: NET BEANS 8.2

# Estructura del proyecto:

El proyecto cartel cuenta actualmente con 16 clases públicas, entre ellas 2 clases abstractas y una clase principal (Partie) con el método main, y también cuenta con 2 interfaces (Showable, Reutilisable). El esquema grafico UML de relaciones y herencias de clases e interfaces se puede visualizar en el archivo adjunto KartelUML.pdf

# Lista de clases (c), clases abstractas (ca), interfaces (i), clase principal (cm):

|  |  |
| --- | --- |
| Boss (c)  De (c)  Detective (c)  Gang (c)  GangMember (ca)  Gangster (c)  Joueur (c)  KartelIntelligence (c)  LesGangs (c) | LesJoueurs (c)  Lire (c)  Partie (cm)  Plateau (c)  PotDeVin (c)  Prison (c)  Recompense (ca)  Reutilisable (i)  Showable (i) |

# Especificación detallada de clases:

## GangMembre

### Cabecera

public abstract class GangMember implements Showable

### Definición

Esta clase se encarga de modelar todos los miembros de una pandilla [un miembro de una pandilla es todo aquel o aquello que pertenece a esta sin importar su forma, estatus o cargo. Una pandilla puede tener muchos miembros, pero un miembro solo puede pertenecer a una sola pandilla, un miembro de una banda es inmutable, es decir si, no puede cambiar de tipo de membro] entre ellos los Bosses, Gangsters y los Pots de vin

### Atributos

String type

define el tipo de miembro, por ejemplo “BOSS”, “GANGSTER” o “POT DE VIN”

Gang gang

hace referencia a la banda a la que pertenece este miembro

### Métodos

#### Constructor

public GangMember (Gang gang, String type)

inicializa el GangMembre perteneciente a una pandilla y con su tipo pasados por parámetro

#### Getters y Setters

public Gang getGang()

devuelve la referencia al objeto gang que pertenece

public String getType()

Devuelve el tipo de gang mebre

## Boss

Cabecera

public class Boss extends GangMember

### Definición

Esta clase se encarga de modelar todos los Bosses de una pandilla [el boss de una pandilla es un miembro de esta con la cualidad de que es el jefe y dependiendo de su estado de libertad los otros miembros pueden cambiar sus valores]

### Atributos

boolean prisonnier

define el estado del Boss true si está en pision

public static final String TYPE = "BOSS"

cadena de texto con el tipo de Gang member establecido como BOSS

### Métodos

#### Constructor

public Boss(Gang gang)

inicializa el Boss llamando al constructor de su clase padre (GangMembre) pasándole por parámetro la pandilla y el tipo. Por defecto el Boss no es prisionero

#### Getters y Setters

public boolean isPrisonnier()

devuelve si el boss es prisionero o no

public void setPrisonnier(boolean prisonnier)

Establece si el boss es prisionero o no

#### Overrides

public String toString()

devuelve una cadena de texto con la palabra “Boss” y el nombre de la pandilla

public String show()

devuelve la representación del boss para el juego (abreviación de la pandilla encerrada entre corchetes)

## Recompense

Cabecera

Public abstract class Recompense extends GangMember

### Definición

Esta clase se encarga de modelar a todos los miembros que puedan tener una recompensa [una recompensa es un valor que pueda aportar el miembro al jugador que lo tiene, la recompensa varía en función del estado de libertad del jefe de la pandilla]

### Atributos

int valeurBossPrisonnier

contiene valor de la recompensa si e Boss de la pandilla está en prisión

int valeurBossNonPrisonnier

contiene valor de la recompensa si e Boss de la pandilla NO está en prisión

### Métodos

#### Constructor

public Recompense(Gang gang, String type, int valeurBossPrisonnier, int valeurBossNonPrisonnier)

inicializa la Recompensa llamando al constructor de su clase padre (GangMembre) pasándole por parámetro la pandilla y el tipo, además inicializa los distintos valores de la recompensa con los parámetros correspondientes

#### Otros

public int getRecompense()

Devuelve el valor de la recompensa dependiendo si el estado de libertad del jefe de la pandilla

#### Getters y Setters

public int getValeurBossPrisonnier()

devuelve si el valor de la recompensa si el jefe de la pandilla está en prisión

public void setValeurBossPrisonnier(int valeurBossPrisonnier)

Estable si el valor de la recompensa si el jefe de la pandilla está en prisión

public int getValeurBossNonPrisonnier()

devuelve si el valor de la recompensa si el jefe de la pandilla NO está en prisión

public void setValeurBossNonPrisonnier(int valeurBossPrisonnier)

Estable si el valor de la recompensa si el jefe de la pandilla No está en prisión

## Gangster

Cabecera

public class Gangster extends Recompense

### Definición

Esta clase se encarga de modelar todos los Gangster de una pandilla [un gangster es un miembro de la pandilla que tiene un nivel y dependiendo de ese nivel tiene un valor de recompensa]

### Atributos

int niveau;

entero que contiene el nivel del gangster

public static final String TYPE = "GANGSTER"

cadena de texto con el tipo de Gang member establecido como GANGSTER

### Métodos

#### Constructor

public Gangster(Gang gang, int niveau)

inicializa el Gangster llamando al constructor de su clase padre (Recompense) pasándole por parámetro la pandilla, el tipo como y los valores de las recompensas dependiendo su nivel

#### Getters y Setters

public int getNiveau()

devuelve el nivel del gangster

public void setNiveau(int niveau)

Establece el nivel del gangster

#### Overrides

public String toString()

devuelve una cadena de texto con la palabra “Gangster”, su valor y el nombre de la pandilla

public String show()

devuelve la representación del Gangster para el juego (el nivel más abreviación de la pandilla)

## PotDeVin

Cabecera

public class PotDeVin extends Recompense

### Definición

Esta clase se encarga de modelar todos los PotDeVin de una pandilla [un PotDeVIn es un miembro de la pandilla que tiene un valor que establece su recompensa]

### Atributos

public static final String TYPE = "POT DE VIN"

cadena de texto con el tipo de Gang member establecido como POT DE VIN

### Métodos

#### Constructor

public PotDeVin(Gang gang, int recompense)

inicializa el PotDeVin llamando al constructor de su clase padre (Recompense) pasándole por parámetro la pandilla, el tipo, y los valores de las recompensas dependiendo su valor

public PotDeVin(Gang gang)

llama al constructor anterior pasándole por parámetro la pandilla y 3 como recompensa

#### Overrides

public String toString()

devuelve una cadena de texto con las palabras “Pot de vin”, su valor (si el Boss no está en prisión) y el nombre de la pandilla

public String show()

devuelve la representación del PotDeVin para el juego ($ más abreviación de la pandilla)

## Gang

Cabecera

public class Gang implements Reutilisable

### Definición

Esta clase se encarga de modelar todos los Gangs del juego [un Gang es un grupo de GangMembres con un nombre y una abreviación]

### Atributos

String nom

nombre de la pandilla

String abb

seudónimo de la pandilla

ArrayList<GangMember> gangMembers

Lista que contendrá todos los miembros de la pandilla

### Métodos

#### Constructor

public Gangster(Gang, int niveau)

inicializa la pandilla con un nombre y una abreviación, luego inicializa la lista con el jefe de la pandilla agregado como primer miembro

#### Otros

initDefaultGangMembres()

agrega 4 Gangster y 1 PotDeVin a la lista de miembros

public void bossInPrision(boolean etat)

le indica al jefe de la pandilla si está en prisión o no

public boolean isBossInPrison()

devuelve si el jefe de la pandilla está en prisión o no

public void addGangMember(GangMember g)

agrega un miembro nuevo a la pandilla, este no puede ser un Boss

public Boss getBoss()

Devuelve el jefe de la pandilla

public ArrayList<Gangster> getGangsters()

devuelve una lista con todos los gansgters de la pandilla

public ArrayList< PotDeVin> getPotDeVin ()

devuelve una lista con todos los PotDeVin de la pandilla

#### Getters y Setters

public ArrayList<GangMember> getGangMembresList()

Devuelve la lista con todos los miembros de la pandilla

public String getNom()

devuelve el nombre de la pandilla

public String getAbb()

Devuelve la abreviación de la pandilla

#### Overrides

public void reset()

resetea la pandilla para utilizarla d nuevo, lo que hace es indicarle al jefe que ya n está en prisión

public String toString()

devuelve una cadena de texto con el nombre, la abreviación y todos los miembros de la pandilla

public int hashCode()

cambia el hasCode de la instancia, calculándolo con respecto al seudónimo de la pandilla

public boolean equals(Object obj)

compara dos pandillas para ver si son las mismas [dos pandillas son iguales si se trata de la misma pandilla o si tienen el mismo seudónimo]

## LesGangs

Cabecera

Public class LesGangs

### Definición

Esta clase se encarga de modelar a un grupo de Gangs con los métodos necesarios para su tratamiento más cómodo en otras clases

### Atributos

ArrayList<Gang> gangs

Lista que va a contener a todos los Gangs

### Métodos

#### Constructor

public LesGangs ()

inicializa el grupo con una lista de gangs vacía

#### Otros

public boolean addGang(Gang g)

Agrega un Gang a la lista si este no está ya agregado y devuelve true si fue posible agregarlo

public int addGangs(LesGangs gs)

Agrega un grupo Gangs a la lista y devuelve la cantidad de agregados

public Gang getGang(int index)

Devuelve el Gang que está en la posición de la lista indicada por parámetro, si la posición no se encuentra devuelve null

public Gang getGang(String abb)

Devuelve el Gang de la lista de seudónimo abb, si este no existe devuelve null

public int getNbGangs()

Devuelve la cantidad de Gangs que hay en la lista

private boolean gangRepete(Gang g)

Verifica si el Gang existe en la lista

public void resetGangs()

hace la llamada del método reset de todos los gangs de la lista

public ArrayList<GangMember> getGangsMembers()

Devuelve una lista con todos los miembro de todos los Gangs de la lista

#### Getters y Setters

ArrayList<Gang> getGangs()

devuelve la lista de todos los gangs

#### Overrides

public String toString()

Un String con todos los Gangs y sus especificaciones

## Joueur

Cabecera

public class Joueur implements Comparable, Reutilisable, Showable

### Definición

Esta clase se encarga de modelar a los jugadores, estos tendrán un nombre y un grupo de recompensas de las cuales se calculará su socre.

Cada vez que un jugador gana un juego este suma un punto a los juegos ganados

Un jugador puede ser automático [Un jugador es llamado automático cuando es la inteligencia KartelIntelligence quien lo controla] o puede tener una inteligencia del nivel deseado y con un volumen deseado

### Atributos

String nom

Nombre del jugador

ArrayList<Recompense> recompenses

Lista que tendrá las recompensas del jugador en el juego

int jeuxGagnes

entero con la cantidad de juegos ganados

int iNiveau

Entero con el nivel de la inteligencia del jugador

int iVolumen

Entero con el nivel de volumen de la inteligencia del jugador

### Métodos

#### Constructor

public Joueur(String nom)

inicializa un jugador con un nombre y una lista de recompensas vacía, los juegos ganados por defecto son cero, el jugador no es automático y el volumen y nivel de la inteligencia es igual a cero

#### Otros

public void addJeuGange()

Suma al jugador otro juego ganado

public void addRecompense( Recompense recompense)

Agrega la recompensa a la lista de recompensas del jugador

public int getScore()

Devuelve el Score del jugador [el escore es la suma del valor de todas las recompensas que posee el jugador]

#### Getters y Setters

public ArrayList<Recompense> getRecompenses()

Devuelve la lista de recompensas del jugador

public String getNom()

Devuelve el nombre del jugador

public int getJeuxGagnes()

Devuelve la cantidad de juegos ganados

public boolean isAutomatique()

Devuelve true si el jugador es automático

public void setAutomatique(boolean automatique)

Establece si el jugador es automático

public int getiNiveau()

Devuelve el nivel de la inteligencia del jugador

public int setiNiveau(int iNiveau)

Establece el nivel de la inteligencia del jugador

public int getiVolumen ()

Devuelve el nivel del volumen de la inteligencia del jugador

public int setiVolumen (int iVolumen)

Establece el nivel del volumen de la inteligencia del jugador

#### Overrides

public String toString()

Devuelve una cadena de texto con el nombre del jugador y su score

public int compareTo(Object o)

Compara dos jugadores con respecto a su score, este método se usa para ordenar los jugadores al final del juego y obtener el ganador

public void reset()

Resetea al jugador para utilizarlo de nuevo, lo que hace es eliminar todas las recompensas de la lista

public String show()

Devuelve un String con el texto del jugador que se mostrara en el juego, este texto contiene su nombre, su score y la lista de recompensas

public int hashCode()

cambia el hasCode de la instancia, calculándolo con respecto al nombre del jugador

public boolean equals(Object obj)

compara jugadores para ver si son iguales [dos jugadores son iguales si se trata del mismo jugador o si tienen el mismo nombre]

## LesJoueurs

Cabecera

Public class LesJoueurs

### Definición

Esta clase se encarga de modelar a un grupo de Jugadores con los métodos necesarios para su tratamiento más cómodo en otras clases

### Atributos

private ArrayList<Joueur> joueurs

Lista que va a contener a todos los jugadores

private int indexJoueurActuel

índice que va a hacer referencia al jugador actual de la lista [El jugador actual se refiere al jugador que tiene el turno de juego]

### Métodos

#### Constructor

public LesJoueurs ()

inicializa el grupo de jugadores con una lista de vacía

#### Otros

public void nextJoueur()

Cambia el índice del jugador actual por el del siguiente en la lista, si el índice del jugador actual es el ultimo el siguiente será el cero

public String afficherPositions(int ctt)

Devuelve un String que va a mostrar todas las posiciones de los jugadores de la lista,

El String se calcula ordenando la lista de jugadores en orden descendente con respecto a su score y luego se van agregando con su posición con respecto a los jugadores, si hay dos jugadores repetidos estos aparecerán en la misma posición

public ArrayList<Joueur> getListResultats()

Devuelve una lista con los jugadores ordenados en orden descendente con respecto a su score

public boolean addJoueur( Joueur j)

Agrega un jugador a la lista de jugadores si este no está repetido, devuelve true si el jugador fue agregado

public int getNbJoueurs ()

Devuelve la cantidad de Jugadores que hay en la lista

public Joueur getJoueur(int index)

Devuelve el Gang de la posición indicada en la lista, si esta posición no existe devuelve null

public int getIndexJoueur(Joueur j)

Devuelve el la posición en la lista del jugador pasado por parámetro

public boolean remove(int index)

Elimina el jugador de la posición indicada por parámetro si este existe, si se eliminan todos los jugadores el jugador actual se convierte en -1, devuelve true si el jugador fue eliminado

public boolean remove(String nom)

Elimina el jugador cuyo nombre es pasado por parámetro, devuelve true si el jugador fue eliminado

private boolean joueurRepete (Joueur j)

Verifica si el Jugador existe en la lista

public Joueur getJoueurActuel()

Devuelve el jugador actual

public Joueur getJoueurGagnant()

devuelve el primer jugador con el score más alto, (SOLO PARA PRUEBAS)

public ArrayList<Joueur> getGagnant()

Devuelve una lista de jugadores ordenando los jugadores por su score (NO SE UTILIZA)

public void removeAll()

Elimina todos los jugadores de la lista

public void resetJoueurs()

llama al método reset de todos los jugadores de la lista

public String showJoueurs()

#### Getters y Setters

public ArrayList<Joueur> getJoueurs()

devuelve la lista de los jugadores

public int getIndexJoueurActuel()

devuelve la posición del jugador actual

public void setIndexJoueurActuel(int index)

establece la posición del jugador actual si esta posición está permitida

#### Overrides

public String toString()

Devuelve un String con todos los jugadores y sus especificaciones

## Detective

Cabecera

public class Detective implements Showable

### Definición

Esta clase se encarga de modelar el detective del juego [el detective es el jeton que mueven todos los jugadores en el plato de juego para capturar los miembros de la banda]

### Atributos

int position

entero que va a contener la posición del detective en el juego

int max

entero que va a contener la posición máxima que el detective puede alcanzar en el juego

### Métodos

#### Constructores

public Detective(int max)

inicializa el Detective con la posición máxima dada por parámetro y la posición igual a cero

public De()

llama al constructor anterior pasándole por parámetro cero para la posición máxima

#### Otros

public void avancer(int deplacement)

Método que hace avanzar al detective en el tablero teniendo en cuenta de volver a la posición inicial si se sobrepasa la posición máxima

#### Getters y Setters

public int getPosition()

devuelve la posición actual del detective

public void setPosition( int pos)

establece la posición actual del detective si esta está permitida

public int getMax ()

devuelve la posición máxima que el detective puede alcanzar

public void setMax ( int max)

establece la posición máxima que el detective puede alcanzar

#### Overrides

public String toString()

devuelve una cadena de texto con la palabra con las especificaciones del detective conteniedo su posición actual y su posición máxima

public String show()

devuelve una cadena de texto con el texto que se mostrara para representar al detective en el juego [el detective está representado por “<>”]

## De

Cabecera

public class De

### Definición

Esta clase se encarga de modelar el dado del juego

### Atributos

final int []faces

arreglo de enteros que contendrá todas las caras del dado, este no puede cambiar a lo largo del juego

int indexFaceCourante

entero el cual indicar la posición en el arreglo de caras de la cara se mostrará

### Métodos

#### Constructores

public De(int [] faces)

inicializa el Dado con las caras pasadas por parámetro, y establece el índice de la cara actual a cero

public De()

llama al constructor anterior pasándole por parámetro un arreglo con las caras (2,2,3,3,4,4)

#### otros

public int lancer()

elige aleatoriamente una de las caras del dado y la devuelve, además establece el índice de esta cara como el indicie actual

#### Getters y Setters

public int getFaceCourante()

devuelve la cara actual

public int getIndexFaceCourante()

Devuelve el índice de la cara actual

public void setIndexFaceCourante(int indexFaceCourante)

establece el índice de la cara actual

#### Overrides

public String toString()

devuelve una cadena de texto con la palabra “De” que contiene también la cantidad de caras y todas sus caras

## Prison

Cabecera

public class Prison implements Reutilisable, Showable

### Definición

Esta clase se encarga de modelar a la prisión del juego [la prisión es el lugar a donde van los Boss luego de ser capturados por el Detective]

### Atributos

ArrayList<Boss> bosses

Lista que contendrá todos los Boss de la prisión

int maxPrisonniers

cantidad máxima de prisioneros permitidos en la prisión

### Métodos

#### Constructores

public Prison(int maxPrisonniers)

inicializa la prisión con un número máximo de prisioneros y la lista de prisioneros vacía

#### otros

public boolean addBoss(Boss boss)

Agrega un Boss a la prisión si esta no está completa y devuelve false si la prisión esta completa

public int getNbBosses()

Devuelve la cantidad de prisioneros que hay en la prisión

public boolean isPrisonComplete()

Devuelve true si el número de jefes en la lista igual al máximo aceptado por la prisión

public Boss getBoss(int index)

devuelve el jefe con el indicio buscado en la lista de prisioneros

#### Getters y Setters

public int getMaxPrisoniers()

devuelve la cantidad máxima de prisioneros aceptados por la prisión

public void setMaxPrisonniers(int maxPrisonniers)

establece la cantidad máxima de prisioneros aceptados por la prisión

public ArrayList<Boss> getBosses()

Devuelve la lista de Boss en la prisión

#### Overrides

public void reset()

Resetea la prisión para usarla de nuevo, lo que hace es eliminar decirle a los Boss que ya no son prisioneros y eliminarlos de la lista

public String toString()

devuelve una cadena de texto con la palabra con las características de la prisión entre ellas el número máximo de prisioneros aceptados y todos los Boss que contiene

public String show()

Devuelve el texto que se va a mostrar de la prisión en el juego.

## De

Cabecera

public class De

### Definición

Esta clase se encarga de modelar el dado del juego

### Atributos

final int []faces

arreglo de enteros que contendrá todas las caras del dado, este no puede cambiar a lo largo del juego

int indexFaceCourante

entero el cual indicar la posición en el arreglo de caras de la cara se mostrará

### Métodos

#### Constructores

public De(int [] faces)

inicializa el Dado con las caras pasadas por parámetro, y establece el índice de la cara actual a cero

public De()

llama al constructor anterior pasándole por parámetro un arreglo con las caras (2,2,3,3,4,4)

#### otros

public int lancer()

elige aleatoriamente una de las caras del dado y la devuelve, además establece el índice de esta cara como el indicie actual

#### Getters y Setters

public int getFaceCourante()

devuelve la cara actual

public int getIndexFaceCourante()

Devuelve el índice de la cara actual

public void setIndexFaceCourante(int indexFaceCourante)

establece el índice de la cara actual

#### Overrides

public String toString()

devuelve una cadena de texto con la palabra “De” que contiene también la cantidad de caras y todas sus caras

## Plateau

Cabecera

public class Plateau implements Reutilisable, Showable

### Definición

Esta clase se encarga de modelar el tablero de juego, este va a contener a todos los jetones (miembros de todas la bandas), la prisión y el detective

### Atributos

LesGangs gangs

Grupo de Gang que van a estar en el juego

ArrayList <GangMember> jetons

Lista de jetons que estarán presente en el trablero

Detective detective

Detective del juego

Prison prison

Prision del juego

### Métodos

#### Constructores

public Plateau(LesGangs gangs, int maxPrisonniers)

inicializa el tablero con un grupo de gans y un máximo número de prisioneros posibles en la prisión, si la lista de gangs no esta vacía crea los jetons y luego inicializa el detective

public Plateau(LesGangs gangs)

llama al constructor anterior pasándole el grupo de Gangs y el máximo de prisioneros como 5

#### otros

public void setGangs(LesGangs gangs)

elimina el grupo de Gangs actual y establece el pasado por parámetro, luego crea los Jetons

private void createJetons()

Crea los jetons del plateau obteniéndolos del grupo de Gangs, luego los mezcla para ponerlos de forma aleatoria y establece los parámetros del detective como su posición actual y su posición máxima

public int getNbJetons()

devuelve la cantidad de jetons presente en el tablero

public GangMember next( int pas )

hace avanzar al detective en el tablero hasta una cierta posición de la lista de jetones y elimina al jeton de esa posición, devuelve el jeton eliminado y establece los nuevos parámetros del detective

public void melanger(List<GangMember> list)

Utiliza la clase Collections para mezclar la lista de jetones haciendo uso de la clase Random

#### Getters y Setters

public ArrayList<GangMember> getJetons()

devuelve la lista de jetones que están en el tablero

public Detective getDetective()

Devuelve el detective presente en el tablero

public Prison getPrison()

Devuelve la prisión del juego

public LesGangs getgangs()

Devuelve el grupo de Gangs presente en el juego del juego

#### Overrides

public String toString()

devuelve una cadena de texto que intenta modelizar el tablero

public String show()

Devuelve el texto que representara el tablero en el juego, este contiene la representación de los Jetons y la prisión

public void reset()

Resetea el tablero para usarlo nuevamente creando los Jetons otra vez y reseteando la prisión

## Partie

Cabecera

public class Plateau implements Reutilisable

### Definición

Esta es la clase principal del juego contiene el método main y el menú que permite interactuar con el o los usuarios, además contiene una instancia de KerteIntelligence que permite ayudar a jugar a los jugadores o jugar en contra de ella

### Atributos

Plateau

PLateau del juego

LesJoueurs joueurs

Grupo de jugadores presentes en el juego

De

Dado del juego

KartelIntelligence intelligence

Inteligencia presente en el juego

boolean rejouer = false

true luego de haber jugado la primera vez para activar la opción de rejouer

### Métodos

#### Constructores

public Partie()

inicializa la partida, inicializa el grupo de jugadores vacío, un tablero y una inteligencia, además de iniciar un dado por defecto y hacer la llamada del menú

#### otros

public void menu()

crea el menú del juego y permite elegir al usuario las distintas opciones que contiene para parametrizar el juego, hacer estadística o comenzar a jugar

private boolean ajouterJoueur(String nom, boolean automatique, int iNiveau, int iVolimen)

permite crear un nuevo jugador con un nombre, y establecer los parámetros de su inteligencia, como su nivel, y su volumen, también permite establecer si el jugador será o no controlado por la inteligencia y luego lo agrega a la lista

private void setGangs(LesGangs gangs, int maxP)

establece el grupo de gangs del tablero y parametriza la prisión estableciendo el número máximo de prisioneros y reseteándola

private void setGangs(LesGangs gangs)

establece el grupo de gangs del tablero

private void chargerIntelligencesTest()

elimina a todos los jugadores actuales y crea 4 jugadores controlados totalmente por las inteligencias cada una de distinto nivel y con volumen cero

private void chargerDefaultsJoueurs()

elimina a todos los jugadores actuales y crea 3 jugadores ayudados por inteligencias de distintos niveles

private void chargerDefaultsGangs()

Crea un 7 Gangs con sus miembros por defecto y llama a la función setGangs para agregarlos al tablero

public void modeStatistiques(int ctt)

Inicializa el modo estadísticas para poner a jugar a las inteligencias y medir sus niveles, carga las inteligencias y los Gangs por defecto para luego hacer la cantidad de juegos pasada por parámetro

Luego obtiene una lista clasificada con los jugadores dependiendo la cantidad de juegos ganados gracias a la clase Comparator y al método compare, luego muestra los jugadores con un porcentaje de juegos ganados

private void modeVs()

Inicializa el modo 1 a 1 que pone a jugar al utilizador contra la inteligencia de nivel 3

private void modifierPrison()

Permite al usuario de modificar la cantidad máxima de prisioneros que puede haber en la prisión

private void suprimerJoueur()

Permite al usuario de eliminar los jugadores

public void createJoueurs()

Permite al usuario de crear nuevos jugadores

public void createGangs()

Permite al usuario de crear nuevos Gangs

private void deModification()

Permite al usuario de modificar el dado, su cantidad de caras y los valores de estas

public void afficerLesParametres()

permite mostrar los parámetros actuales del juego, como los jugadores, la prisión, el dado…

private void jouer()

Método que permite jugar explicación detallada en el código

#### Overrides

public void reset()

permite reiniciar los jugadores y el tablero para jugar con los mismos parámetros actuales