COMPTE RENDU PROJET INFO4B

Auteur : *HAMIDOU Mohammed Nazim IE4-I911 (En Monôme)*

Mini Vidéo youtube non répertoriée ou je lance le jeu devant vous : https://youtu.be/8R2mSg7e1kA

I)

-Présentation de mon projet , Sujet 2 Burgers Time, j'ai dû travaillé en monôme, malheureusement je n'ai pas pu terminer tout le projet, je n'ai pas eu le temps pour faire le classement des joueurs, ainsi que la partie des équipes collaboratifs, ceci dit mon jeu en Solo est bien fonctionnel que ce soit en contrôlant un Cuisinier, un œuf ou une saucisse et en multi c'est possible de jouer en 1 contre 1,

Concernant le Burgers Time, c'est un jeu rétro assez simple, un «platformer»dans le quel on doit former des burgers et on est poursuivi par plusieurs ennemis (des Œufs et des Saucisses) 4 dans mon cas (2 de chaque), on a 5 vie au total et pour gagner la partie il ne doit plus rester aucun constituants de burger sur les plateformes dans l'autre cas la partie est perdue et donc remportée par les ennemis.

II)

-Concernant mes structures j'ai utilisé 1 tableau d'entier en 1 dimension pour le labyrinthe et 2 Listes de caractères pour les Entités et pour les Constituants du burgers.

-Concernant les threads j'en ai pas utilisé pour mon jeu principal, mais j'en ai utilisé pour le réseau.
-J'ai repris votre code du ServeurMC.java et ThreadedMC.java, l'architecture de mon réseau se repose sur le fait que tout mon jeu se passe sur le serveur, c'est assez simple comme architecture :

- Mon serveur reçoit les touches pour le déplacement (que ce soit pour le cuisinier ou pour un ennemie c'est la même chose), fait le déplacement en interne, il met à jour le labyrinthe et renvoie le nouveau labyrinthe en une seule variable string aux clients, je me suis totalement inspiré de votre programme de tchat pour réaliser mon réseau et en soit ça ne change pas grand-chose car le

déplacement va être envoyer au serveur via un string et le nouveau labyrinthe sera envoyé aux clients via un string. Pour cela j'ai utilisé une fonction jeuToString qui à la place de faire l'affichage via des System.out.print(), elle met chaque caractères présents dans l'affichage dans une variable.

-Le premier client créé sera un cuisinier et le deuxième un ennemi.

-Concernant mon jeu en plus de détails, pour les ennemis j'ai repris une fonction que j'avais codé pour mon projet en info4A qui calcule la distance minimal entre 2 id donnés au préalable, elle s'inspire de l'algorithme de djikstra trouvé sur wikipedia :

https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra

```
Entrées : G = (S, A) un graphe avec une pondération positive poids des arcs, s_deb un sommet de S

P := ∅
d[a] := +∞ pour chaque sommet a
d[s_deb] = 0
Tant qu'il existe un sommet hors de P
    Choisir un sommet a hors de P de plus petite distance d[a]
    Mettre a dans P
    Pour chaque sommet b hors de  P voisin de a
       Si d[b] > d[a] + poids(a, b)
           d[b] = d[a] + poids(a, b)
           prédécesseur[b] := a
    Fin Pour
Fin Tant Que
```

- Cela m'a prit une cinquantaine de lignes pour un résultat très satisfaisant, je pourrai expliquer plus mais c'est exactement le même algorithme mis à part le changement de poids(a,b) en +1 (car on a pas vraiment de poids d'arc) j'ai aucun changement.

-Concernant mes déplacements, j'ai une simple fonction deplaceEntite, qui pour un caractère et une touche entre z,q,s,d et un id (l'id du dit caractère) fait un déplacement, bien sûr si cela est possible, y'aura pas de déplacement dans le vide par exemple même si cela était l'intention du joueur.

-Concernant mes constituants, j'ai une fonction contactaliment qui déplace l'aliment (si en contact avec un cuisinier) vers le sol en bas, et si un aliment s'y trouve une récursion simple s'impose pour régler le problème.

-Les ennemis ne peuvent pas déplacer de constituant de ce fait j'ai opté pour afficher les ennemis plutôt que les constituants si ces deux là se trouve sur une même case.

-Le nombre de vie est égal à 5 et le nombre de poivre est égal à 3, on peut « poivrer » un ennemi à gauche ou à droite en utilisant un a ou un e respectivement, donc un « de » déplacera le cuisinier vers la droite et va « poivrer » l'ennemi s'y trouvant dans cette case, et c'est ici ou j'ai rencontré mon plus gros probléme. Comment enlever les Ennemis poivrés du labyrinthe temporairement(car ils finissent par remarcher un petit temps aprés, le poivre est donc simplement une diversion)

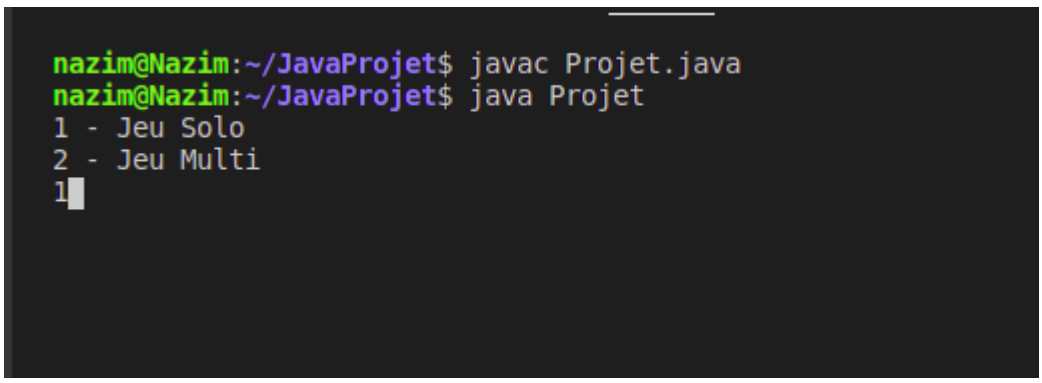
--La solution était les tables de hachages, j'en ai créé 2, une en int,int qui prend une id et le nombre de tour(s) avant que cette entité ne revienne en jeu, et la deuxième en int,string prend une id et le string de l'entité 'E' pour l'oeuf par exemple, comme mon jeu est en asynchrone à chaque tour le nombre de tour(s) avant qu'une entité « morte » ne revienne en jeu est décrémenté, et donc à 0 cette entité revient en jeu.

III)

-Concernant mes classes, j'en ai 3 pour le serveur au totales la classe main, la classe Burgers Time, la classe ThreadedClient qui est un thread ; et 2 pour le client la classe main et la case gererSaissie.

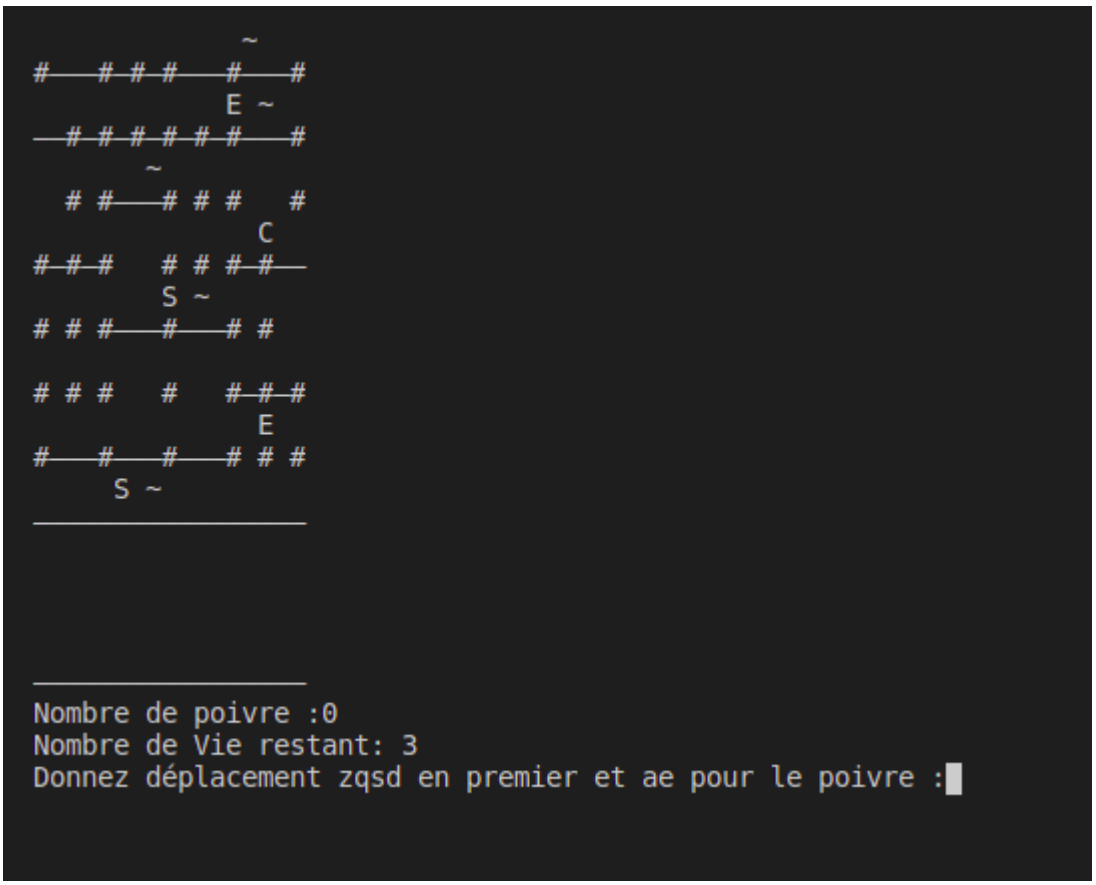
IV)

-Concernant mon lancement de jeu SOLO :



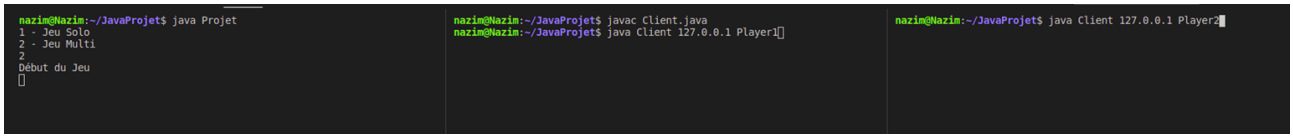
-Voilà ce que ça affiche aprés :

un « qa » déplace le



cuisinier à droite et met du poivre à droite .

-Concernant mon jeu en MULTI avec réseau:



Voilà pour le lancement

Voilà pour ce que ça affiche après : au milieu le cuisinier et a droite le « EGG »

V)

- Conclusion : je pense qu'avec un peu plus de temps je pourrai certainement mettre l'aspect des scores de joueurs en place, et c'est tout. Au final je suis assez fier et j'ai bien aimé faire ce projet. Merci à vous.

VI) Code Complet Du Projet :

```
import java.io.*;
import java.net.*;
import java.util.*;


class BurgersTime {
private int[] Decore;
private List<Character> Constituants;
private List<Character> Entites;
private Hashtable < Integer , Integer > ht1;
private Hashtable < Integer , Character > ht2;
private int NB_COLONNES;
private int NB_LIGNES;
private int Sommet = 0;
private char Echelle;// int 2
private char Sol;// int 1
private char Vide;// int 0
private int[] Pile;
private int Poivrenb;
private char Aliment;
private char Saucisse;
private char Poivre;
private char Egg;
private char Cuisinier;
private int[] Burger;
private int nbvie;
private boolean multiplayer = true;
public BurgersTime(int[] Decore, List<Character> Constituants, List<Character> Entites, int
NB_LIGNES,
int NB_COLONNES) {
this.NB_LIGNES = NB_LIGNES;
this.Decore = Decore;
this.NB_COLONNES = NB_COLONNES;
this.Constituants = Constituants;
```

```
this.Entites = Entites;
this.nbvie = 5;
this.Burger = new int[4];
this.Pile = new int[NB_LIGNES * NB_COLONNES];
this.Sommet = 0;
this.Echelle = '#';
this.Sol = '—';
this.Poivrenb = 3;
this.Vide = ' ';
this.Cuisinier = 'C';
this.Egg = 'E';
this.Poivre = 'P';
this.Saucisse = 'S';
this.Aliment = '~';
this.ht1 = new Hashtable<>();
this.ht2 = new Hashtable<>();
startJeu();
}

private int getID(int ligne, int colonne) {
return ligne * NB_COLONNES + colonne;
}

private int getLigne(int id) {
return id / NB_COLONNES;
}

private int getCol(int id) {
return id % NB_COLONNES;
}

private void modifie(int ligne, int colonne, int x) {
Decore[getID(ligne, colonne)] = x;
}

private void modifie(int id, int x) {
Decore[id] = x;
}

private int lit(int ligne, int colonne) {
return Decore[getID(ligne, colonne)];
}

private int lit(int id) {
```

```
return Decore[id];
}

private int getNbLignes() {
return NB_LIGNES;
}

private int getNbColonnes() {
return NB_COLONNES;
}

private void push(int x) {
Pile[Sommet] = x;
Sommet++;
}

private int pop() {
Sommet--;
return Pile[Sommet];
}

public void Affiche() {
for (int i = 0; i < NB_LIGNES; i++) {
for (int j = 0; j < NB_COLONNES; j++) {
if (Entites.get(getID(i, j)) != Vide)
System.out.print(Entites.get(getID(i, j)));
else
System.out.print(Constituants.get(getID(i, j)));

}
System.out.println("");

for (int j = 0; j < NB_COLONNES; j++) {
if (Decore[getID(i, j)] == 1)
System.out.print(Sol);
else if (Decore[getID(i, j)] == 0)
System.out.print(Vide);
else
System.out.print(Echelle);

}
System.out.println("");
}
for (int i = NB_LIGNES; i < NB_LIGNES + 4; i++) {
```

```
for (int j = 0; j < NB_COLONNES; j++) {
System.out.print(Constituants.get(getID(i, j)));
}
System.out.println("");
}
for(int j=0;j<NB_COLONNES;j++)
{
System.out.print(Sol);

}
System.out.println("");

}

private int randomIDConstituants() {
int max = (NB_COLONNES * NB_LIGNES) - 1;
int min = 0;
int range = max - min + 1;
int rand = (int) (Math.random() * range) + min;
while (Decore[rand] != 1 || Constituants.get(rand) != Vide)
rand = (int) (Math.random() * range) + min;
return rand;
}

private int randomIDEntites() {
int max = (NB_COLONNES * NB_LIGNES) - 1;
int min = 0;
int range = max - min + 1;
int rand = (int) (Math.random() * range) + min;
while (Decore[rand] != 1 || Entites.get(rand) != Vide)
rand = (int) (Math.random() * range) + min;
return rand;
}

private int idPlayer() {
return Entites.indexOf(Cuisinier);
}

private int distMin(int id1, int id2)
{
if(id1==id2)return 0;
int[] P=new int[NB_LIGNES*NB_COLONNES];
int[] utilise=new int[NB_LIGNES*NB_COLONNES];
int Comp=0,nbcasesb=0,min=NB_LIGNES*NB_COLONNES;
```

```
int idtemp = 0;
int l=0,c =0;
for(int i=0;i<NB_LIGNES*NB_COLONNES;i++)
{
if(Decore[i]!=0)
{P[i]=NB_LIGNES*NB_COLONNES;
nbcasesb++;
utilise[i]=0;
}
else
{P[i]=-1;
utilise[i]=-1;
}
}
P[id1]=0;
while(nbcasesb!=Comp)
{for(int i=0;i<(NB_LIGNES*NB_COLONNES);i++)
{
if(P[i]!=-1 && P[i]<min && utilise[i]==0)
{min=P[i];
idtemp=i;
}
}
utilise[idtemp]=1;
Comp++;
l=getLigne(idtemp);
c=getCol(idtemp);
if ( l-1>=0 && Decore[getID(l-1, c)] == 2 &&P[getID(l-1,c)]>(P[idtemp]+1) && utilise[getID(l-
1,c)]==0 )
P[getID(l-1,c)]=P[idtemp]+1;
if ( l+1<NB_LIGNES && Decore[getID(l, c)]==2 && P[getID(l+1,c)]>(P[idtemp]+1) &&
utilise[getID(l+1,c)]==0 )
P[getID(l+1,c)]=P[idtemp]+1;
if ( c-1>=0 && P[getID(l,c-1)]>(P[idtemp]+1) && utilise[getID(l,c-1)]==0 )
P[getID(l,c-1)]=P[idtemp]+1;
if ( c+1<NB_COLONNES && P[getID(l,c+1)]>(P[idtemp]+1) && utilise[getID(l,c+1)]==0 )
P[getID(l,c+1)]=P[idtemp]+1;
min=NB_LIGNES*NB_COLONNES;
}
return P[id2];
}


public void deplacementEntite(char dep, int id, char P) {
```

```
int ligj = getLigne(id);
int colj = getCol(id);
char x = ' ';
int idx = 0;
int idapresmodif =0;
boolean deplacement = false;
if (P == Cuisinier) {
if (dep == 'd' && colj + 1 < NB_COLONNES && Decore[getID(ligj, colj + 1)] != 0) {
x = Entites.get(getID(ligj, colj + 1));
idx = getID(ligj, colj + 1);
Entites.set(id, Vide);
Entites.set(getID(ligj, colj + 1), P);
deplacement = true;

} else if (dep == 'q' && colj - 1 >= 0 && Decore[getID(ligj, colj - 1)] != 0) {
x = Entites.get(getID(ligj, colj - 1));
idx = getID(ligj, colj - 1);
Entites.set(id, Vide);
Entites.set(getID(ligj, colj - 1), P);
deplacement = true;
} else if (dep == 's' && ligj + 1 < NB_LIGNES && (Decore[getID(ligj, colj)] == 2)) {
x = Entites.get(getID(ligj + 1, colj));
idx = getID(ligj + 1, colj);
Entites.set(id, Vide);
Entites.set(getID(ligj + 1, colj), P);
deplacement = true;
} else if (dep == 'z' && ligj - 1 >= 0 && (Decore[getID(ligj - 1, colj)] == 2)) {
x = Entites.get(getID(ligj - 1, colj));
idx = getID(ligj - 1, colj);
Entites.set(id, Vide);
Entites.set(getID(ligj - 1, colj), P);
deplacement = true;
}
if (x != Vide && deplacement)
Entites.set(idx, x);
} else if (P == Egg || P == Saucisse) {
if (dep == 'd' && colj + 1 < NB_COLONNES && Decore[getID(ligj, colj + 1)] != 0
&& (Entites.get(getID(ligj, colj + 1)) == Vide
|| Entites.get(getID(ligj, colj + 1)) == Cuisinier)) {
idapresmodif = getID(ligj, colj + 1);
if(ht1.get(idapresmodif) == null)
{Entites.set(id, Vide);
Entites.set(idapresmodif, P);
deplacement = true;
```

```
}

} else if (dep == 'q' && colj - 1 >= 0 && Decore[getID(ligj, colj - 1)] != 0
&& (Entites.get(getID(ligj, colj - 1)) == Vide
|| Entites.get(getID(ligj, colj - 1)) == Cuisinier)) {
idapresmodif = getID(ligj, colj - 1);
if(ht1.get(idapresmodif) == null)
{Entites.set(id, Vide);
Entites.set(idapresmodif, P);
deplacement = true;
}

} else if (dep == 's' && ligj + 1 < NB_LIGNES && (Decore[getID(ligj, colj)] == 2)
&& (Entites.get(getID(ligj + 1, colj)) == Vide
|| Entites.get(getID(ligj + 1, colj)) == Cuisinier)) {
idapresmodif = getID(ligj + 1, colj);
if(ht1.get(idapresmodif) == null)
{Entites.set(id, Vide);
Entites.set(idapresmodif, P);
deplacement = true;
}

} else if (dep == 'z' && ligj - 1 >= 0 && (Decore[getID(ligj - 1, colj)] == 2)
&& (Entites.get(getID(ligj - 1, colj)) == Vide
|| Entites.get(getID(ligj - 1, colj)) == Cuisinier)) {
idapresmodif = getID(ligj-1, colj);
if(ht1.get(idapresmodif) == null)
{Entites.set(id, Vide);
Entites.set(idapresmodif, P);
deplacement = true;
}


}
if(deplacement)
{ht1.put(idapresmodif,ht1.get(id));
ht1.remove(id);
ht2.put( idapresmodif , P);
ht2.remove(id);
}
}
```

```
}

private void contactAliment(int id) {
int ligj = getLigne(id);
int colj = getCol(id);
if(Envie())
{
if (Constituants.get(id) == Aliment) {
ligj++;
while (ligj < (NB_LIGNES) && Decore[getID(ligj, colj)] == 0)
ligj++;

if (Constituants.get(getID(ligj, colj)) == Aliment)
contactAliment(getID(ligj, colj));
if(ligj < NB_LIGNES && Entites.get(getID(ligj,colj))!=Vide)Entites.set(getID(ligj,colj),Vide);
Constituants.set(getID(ligj, colj), Aliment);

Constituants.set(id, Vide);
}
}
}

public boolean resteConstituants() {
if (getLigne(Constituants.indexOf(Aliment)) < NB_LIGNES)
return true;
return false;
}

public boolean Envie() {
if (Entites.indexOf(Cuisinier) != -1)
return true;
return false;
}
private char DepEnnemy(int id, int id2)
{ int lig = getLigne(id);
int col = getCol(id);
char dep = '0';
if(id == -1 || id2 == -1) return dep;
int distmin = distMin(id ,id2);
if(lig+1<NB_LIGNES && Decore[getID(lig, col)]==2 && distmin >= distMin(getID(lig+1, col),
id2) )
{
dep = 's';
```

```
distmin = distMin(getID(lig+1, col), id2);
}
if(lig-1>= 0 && Decore[getID(lig-1, col)]==2 && distmin >= distMin(getID(lig-1, col), id2) )
{
dep = 'z';
distmin = distMin(getID(lig-1, col), id2);
}
if(col+1<NB_COLONNES && Decore[getID(lig, col+1)]!=0 && distmin >= distMin(getID(lig,
col+1), id2) )
{
dep = 'd';
distmin = distMin(getID(lig, col+1), id2);
}
if(col-1>=0 && Decore[getID(lig, col-1)]!=0 && distmin >= distMin(getID(lig, col-1), id2) )
{
dep = 'q';
distmin = distMin(getID(lig, col-1), id2);
}
return dep;
}
private void poivrerEnnemy(int id)
{
Entites.set(id,Vide);
ht1.put(id, 1);
}
private void depoivrerEnnemy()
{
for (Map.Entry<Integer, Integer> e : ht1.entrySet())
{
if(e.getValue() < 5) ht1.put(e.getKey() ,e.getValue() + 1);
else Entites.set(e.getKey(), ht2.get(e.getKey()));
}
}
private void majTableHash()
{ int[] L = new int[4];
int i = 0;
for (Map.Entry<Integer, Integer> e : ht1.entrySet())
{ L[i] = e.getValue();
ht1.put(e.getKey(),e.getValue());
i++;

}
for(Map.Entry<Integer, Character> e : ht2.entrySet())
```

```
{
ht2.remove(e.getKey());
}

}
private void PoivreFunction(int dep, int id)
{ if(Poivrenb > 0)
{
if((dep =='e' && getCol(id+1)<NB_COLONNES && Entites.get(id+1)!=Vide) || (dep=='a' &&
getCol(id-1)>=0 && Entites.get(id-1)!=Vide))
{if(dep == 'e')poivrerEnnemy(id+1);
else poivrerEnnemy(id-1);
Poivrenb--;}
}
depoivrerEnnemy();
}
private void startJeu()
{ Entites.clear();
Constituants.clear();
ht1.clear();
ht2.clear();
for (int i = 0; i < NB_COLONNES * NB_LIGNES; i++) {
Entites.add(Vide);
}
for (int i = 0; i < NB_COLONNES * (NB_LIGNES + 4); i++) {
Constituants.add(Vide);
}
Entites.set(randomIDEntites(), Cuisinier);
Entites.set(randomIDEntites(), Egg);
Entites.set(randomIDEntites(), Egg);
Entites.set(randomIDEntites(), Saucisse);
Entites.set(randomIDEntites(), Saucisse);
Constituants.set(randomIDConstituants(), Aliment);
Constituants.set(randomIDConstituants(), Aliment);
Constituants.set(randomIDConstituants(), Aliment);
Constituants.set(randomIDConstituants(), Aliment);
Constituants.set(randomIDConstituants(), Aliment);
ht1.put(Entites.indexOf(Egg),5);
ht1.put(Entites.lastIndexOf(Egg),5);
ht1.put(Entites.indexOf(Saucisse),5);
ht1.put(Entites.lastIndexOf(Saucisse),5);
ht2.put(Entites.indexOf(Egg),Egg);
ht2.put(Entites.lastIndexOf(Egg),Egg);
ht2.put(Entites.indexOf(Saucisse),Saucisse);
```

```
ht2.put(Entites.lastIndexOf(Saucisse), Saucisse);

}
public void Jeu() {
Scanner s = new Scanner(System.in);
String Dep;
while (Envie() && resteConstituants()) {
System.out.print("\033[H\033[2J");
System.out.flush();
Affiche();
System.out.println("Nombre de poivre :"+Poivrenb);
System.out.println("Nombre de Vie restant: "+nbvie);
System.out.print("Donnez déplacement zqsd en premier et ae pour le poivre :");
Dep = s.nextLine();
while(Dep.length()<2)
Dep = Dep + '0';
PoivreFunction(Dep.charAt(1),Entites.indexOf(Cuisinier));
deplacementEntite(Dep.charAt(0), Entites.indexOf(Cuisinier), Cuisinier);
deplacementEntite(DepEnnemy(Entites.indexOf(Egg), Entites.indexOf(Cuisinier)),
Entites.indexOf(Egg), Egg);
deplacementEntite(DepEnnemy(Entites.indexOf(Saucisse), Entites.indexOf(Cuisinier)),
Entites.indexOf(Saucisse), Saucisse);

if(Entites.indexOf(Egg)!=Entites.lastIndexOf(Egg))
deplacementEntite(DepEnnemy(Entites.lastIndexOf(Egg), Entites.indexOf(Cuisinier)),
Entites.lastIndexOf(Egg), Egg);
if(Entites.indexOf(Saucisse)!=Entites.lastIndexOf(Saucisse))
deplacementEntite(DepEnnemy(Entites.lastIndexOf(Saucisse), Entites.indexOf(Cuisinier)),
Entites.lastIndexOf(Saucisse), Saucisse);
contactAliment(idPlayer());

if(!Envie())nbvie--;
if(!Envie() && nbvie!=0)startJeu();


}
System.out.print("\033[H\033[2J");
System.out.flush();
Affiche();
if(!Envie())
System.out.println("You Lost");
else
System.out.println("You Won");
}
```

```
public boolean JeuMulti(String s, int i) {
System.out.print("\033[H\033[2J");
System.out.flush();
if (Envie() && resteConstituants()) {
System.out.println("Nombre de poivre :"+Poivrenb);
//System.out.print("Donnez dep zqsd :");
//System.out.print("Donnez poivre ae :");
if(i == 0)
{if(s.length()>=2) PoivreFunction(s.charAt(1),Entites.indexOf(Cuisinier));
else PoivreFunction('0',Entites.indexOf(Cuisinier));
if(s.length()==0)deplacementEntite('0', Entites.indexOf(Cuisinier), Cuisinier);
else deplacementEntite(s.charAt(0), Entites.indexOf(Cuisinier), Cuisinier);}
if(i!=0) {deplacementEntite(s.charAt(0),Entites.indexOf(Egg), Egg);}
deplacementEntite(DepEnnemy(Entites.indexOf(Saucisse), Entites.indexOf(Cuisinier)),
Entites.indexOf(Saucisse), Saucisse);
if(Entites.indexOf(Egg)!=Entites.lastIndexOf(Egg))
deplacementEntite(DepEnnemy(Entites.lastIndexOf(Egg), Entites.indexOf(Cuisinier)),
Entites.lastIndexOf(Egg), Egg);
if(Entites.indexOf(Saucisse)!=Entites.lastIndexOf(Saucisse))
deplacementEntite(DepEnnemy(Entites.lastIndexOf(Saucisse), Entites.indexOf(Cuisinier)),
Entites.lastIndexOf(Saucisse), Saucisse);
contactAliment(idPlayer());
if(nbvie !=0 && !Envie()){nbvie--;startJeu();}
System.out.println("Nombre de vie restantes : "+nbvie);
if ( ! (Envie() && resteConstituants()) )return false;
}
return true;
}
public String jeuToString()
{
String S="";
for (int i = 0; i < NB_LIGNES; i++) {
for (int j = 0; j < NB_COLONNES; j++) {
if (Entites.get(getID(i, j)) != Vide)
S = S + Entites.get(getID(i, j));
else
S = S + Constituants.get(getID(i, j));

}
S = S + '\n';

for (int j = 0; j < NB_COLONNES; j++) {
if (Decore[getID(i, j)] == 1)
S = S + Sol;
```

```
else if (Decore[getID(i, j)] == 0)
S = S + Vide;
else
S = S + Echelle;

}
S = S + '\n';
}
for (int i = NB_LIGNES; i < NB_LIGNES + 4; i++) {
for (int j = 0; j < NB_COLONNES; j++) {
S = S + Constituants.get(getID(i, j));
}
S = S + '\n';
}
for(int j=0;j<NB_COLONNES;j++)
{
S = S + Sol;

}
S = S + '\n';
return S;

}
}

public class Projet {
static int port = 8080;
static final int maxClients = 50;
static PrintWriter pw[];
static int[] nbrecu = new int[maxClients];
static int numClient = 0;
public static void main(String[] args) throws Exception {
int [] L = {
2,1,1,1,2,1,2,1,2,1,1,1,2,1,1,1,2,
1,1,2,1,2,1,2,1,2,1,2,1,2,1,1,1,2,
0,0,2,0,2,1,1,1,2,0,2,0,2,0,0,0,2,
2,1,2,1,2,0,0,0,2,0,2,0,2,1,2,1,1,
2,0,2,0,2,1,1,1,2,1,1,1,2,0,2,0,0,
2,0,2,0,2,0,0,0,2,0,0,0,2,1,2,1,2,
2,1,1,1,2,1,1,1,2,1,1,1,2,0,2,0,2,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

};
/*int[] L = {
```

```
1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2,
1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1 };*/
/* int[] L = { 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1 };*/

List<Character> Constituants = new ArrayList<Character>();
List<Character> Entites = new ArrayList<Character>();
Scanner Sc = new Scanner(System.in);

BurgersTime Game = new BurgersTime(L, Constituants, Entites, 8, 17);// nb ligne impair
System.out.println("1 - Jeu Solo\n2 - Jeu Multi\n3 - Joueur du jeu multi");
char x = Sc.next().charAt(0);
if(x =='1')
Game.Jeu();
else if (x=='2'){
if (args.length != 0) {
port = Integer.parseInt(args[0]);
}
pw = new PrintWriter[maxClients];
// 1 - Ouverture du ServerSocket par le serveur
ServerSocket s = new ServerSocket(port);
System.out.println("Début du Jeu");
while (numClient < maxClients){
/* 2 - Attente d'une connexion client (la méthode s.accept() est bloquante
tant qu'un client ne se connecte pas) */
Socket soc = s.accept();
/* 3 - Pour gérer plusieurs clients simultanément, le serveur attend que les clients se connectent,
et dédie un thread à chacun d'entre eux afin de le gérer indépendamment des autres clients */
ConnexionClient cc = new ConnexionClient(numClient, soc,Game);
numClient++;
cc.start();
}
}
else
{ Scanner myObj = new Scanner(System.in);
String args0,args1;
```

```
args0 = "127.0.0.1";
System.out.println("Donnez le pseudo ");
args1 = myObj.nextLine();
Client c = new Client();
c.player(args0, args1);
}
}
}
class ConnexionClient extends Thread {
private int id;
private String pseudo;
private boolean arret = false;
private Socket s;
private BufferedReader sisr;
private PrintWriter sisw;
private BurgersTime Jeu;
private int idP ;
private boolean jeu=true;
private static int nbrecu=0;
private String Fin="";

public ConnexionClient(int id, Socket s,BurgersTime Jeu) {
this.id = id;
this.s = s;
this.Jeu = Jeu;
this.idP = nbrecu++;
/* 5a - A partir du Socket connectant le serveur à un client, le serveur ouvre 2 flux :
1) un flux entrant (BufferedReader) afin de recevoir ce que le client envoie
2) un flux sortant (PrintWriter) afin d'envoyer des messages au client */
// BufferedReader permet de lire par ligne
try {
sisr = new BufferedReader(new InputStreamReader(s.getInputStream()));
sisw = new PrintWriter( new BufferedWriter(
new OutputStreamWriter(s.getOutputStream())), true);
} catch(IOException e) {
e.printStackTrace();
}
Projet.pw[id] = sisw;
// 6 - Le serveur attend que le client envoie son pseudo
try {
pseudo = sisr.readLine();
System.out.println(pseudo+" est arrivé(e)");
} catch(IOException e) {
e.printStackTrace();
```

```
}
}

public void run(){
try {
while (true) {
// 10 - Le serveur envoie le message à tous les clients
for(int i=0; i<Projet.numClient; i++){
if (Projet.pw[i] != null ) {
if(i == 0 && Jeu.Envie() && !jeu){Fin ="You Won";}
if(i == 0 && Jeu.resteConstituants() && !jeu){Fin = "You Lost";}
if(i == 1 && Jeu.Envie() && !jeu){Fin = "You Lost";}
if(i == 1 && Jeu.resteConstituants() && !jeu){Fin = "You Won";}
Projet.pw[i].println("\033[H\033[2J"+Jeu.jeuToString()+Fin);
}
}
if(!jeu)System.exit(0);
/* 8 - Le serveur attend que le client envoie des messages avec le PrintWriter côté client
que le serveur recevra grâce à son BufferedReader (la méthode sisr.readLine() est bloquante) */
String str = sisr.readLine(); // lecture du message
if (str.equals("END")) break;
System.out.println("recu de " + id + "," + pseudo + "=> " + str);
// trace locale
jeu = Jeu.JeuMulti(str, idP);
}
// 12a - Le serveur ferme ses flux
sisr.close();
sisw.close();
s.close();
} catch(IOException e) {
e.printStackTrace();
}
}
}

class Client {
static int port = 8080;
static boolean arreter = false;
// Le client attend comme argument l'adresse du serveur et le pseudo (ex. : java Client 127.0.0.1
pseudo pour l'exécuter)
public void player(String args0,String args1) throws Exception {
String pseudo = args1;
// 4 - le client ouvre une connexion avec le serveur
Socket socket = new Socket(args0, port);
```

```
/* 5b - A partir du Socket connectant le serveur au client, le client ouvre 2 flux :
1) un flux entrant (BufferedReader) afin de recevoir ce que le serveur envoie
2) un flux sortant (PrintWriter) afin d'envoyer des messages au serveur */
BufferedReader sisr = new BufferedReader(
new InputStreamReader(socket.getInputStream()));

PrintWriter sisw = new PrintWriter(new BufferedWriter(
new OutputStreamWriter(socket.getOutputStream())),true);
// 7 - Le client envoie son pseudo au serveur
sisw.println(pseudo);
// Gestion des messages écrits via le terminal
GererSaisie saisie=new GererSaisie(sisw);
saisie.start();
while (!arreter) {
/* 9 - Le client attend les messages du serveur. La méthode sisr.ready() permet de vérifier
si un message est dans le flux, ce qui permet de rendre l'action non bloquante */
try {
if (sisr.ready()) {
String str = sisr.readLine();
System.out.println(str);

if(str.contains("You"))System.exit(0);
}
} catch (Exception e) {
try {
Thread.currentThread().sleep(100);
} catch (InterruptedException e1) {}
}
}
System.out.println("END"); // message de fermeture
// 11 - Le client envoie un message pour mettre fin à la connexion, qui fera sortir le serveur de son
while
sisw.println("END");

// 12b - Le client ferme ses flux
sisr.close();
sisw.close();
socket.close();
}
}

/**
* Utiliser un thread pour gérer les entrées clavier du client permet de dissocier les actions
* d'envoi et de réception de messages. C'est indispenseble dans un contexte de chat multi clients,
```

```
* puisqu'un client ne sait pas s'il sera le prochain à envoyer un message ou non, il doit
* être capable de gérer les 2 cas.
*/
class GererSaisie extends Thread {
private BufferedReader entreeClavier;
private PrintWriter pw;

public GererSaisie(PrintWriter pw) {
entreeClavier = new BufferedReader(new InputStreamReader(System.in));
this.pw = pw;
}

public void run() {
String str;
try{
while(!(str = entreeClavier.readLine()).equals("END")){
// 9bis - Le client envoie un message au serveur grâce à son PrintWriter
pw.println(str);
}
} catch(IOException e){
e.printStackTrace();
}
Client.arreter = true;
}
}
```