

DAMERGI MOHAMED ALI

PORTFOLIO

PROJETS UNIVERSITAIRES

ET EXTRA-UNIVERSITAIRES

**Ce PORTFOLIO est
maintenant
disponible en
siteWeb**

<https://d4l1ir3tr0.github.io/D4L1IR3Tr0-pf.github.io/Index>

INDEX

- 1. Estimation pseudo aléatoire de Pi**
- 2. Simulateur pseudo aléatoire d'une population de lapins**
- 3. Simulateur pseudo aléatoire du jeu de la vie**
- 4. Cryptographie code de Cesar**
- 5. Jeu de Puissance 4**
- 6. Site Web**
- 7. Distributeur Automatique de nourriture pour chat**
- 8. Véhicule a deux roues motrices**
- 9. Etude de la base de donnée de la société fictive AUVER**
- 10. Application IOS (transport de Clermont)**

**Ce PORTFOLIO est
maintenant
disponible en
siteWeb**

<https://d4l1ir3tr0.github.io/D4L1IR3Tr0-pf.github.io/Index>

ESTIMATION PSEUDO ALÉATOIRE DE PI

Introduction

Le but de ce projet est de déterminer la valeur de Pi en utilisant des nombres pseudo-aléatoires. Pour atteindre cet objectif, nous allons utiliser l'algorithme de Mersenne Twister, qui est une méthode courante de génération de nombres pseudo-aléatoires.

La méthode employée consiste à générer des nombres aléatoires X et Y compris entre 0 et 1. Pour ensuite appliquer des règles trigonométriques afin de calculer la surface d'un quart de cercle. En multipliant cette surface par 4, on obtient la surface totale du cercle, ce qui permet de déterminer la valeur de Pi.

Cette approche est une méthode connue pour estimer la valeur de Pi. Elle est basée sur le fait que la surface d'un cercle est directement proportionnelle à son rayon au carré. En utilisant des nombres pseudo-aléatoires pour estimer la surface d'un quart de cercle, cette méthode fournit une approximation de la valeur de Pi.

L'algorithme de Mersenne Twister est utilisé en raison de sa rapidité et de sa fiabilité dans la génération de nombres pseudo-aléatoires. Cela permet à l'équipe de générer rapidement un grand nombre de nombres aléatoires pour améliorer la précision de leur estimation de Pi.

```
1 #include <stdio.h>
2 #include <math.h>
3
4 /* -----
5  * ----- */
5 /* estimPi renvoie une estimation de pi*/
6 /*
7 /* En entrée: inNbPts un nombre de points de type
8 long */
9 /*
10 /* En sortie: une estimation de pi */
11 /*
12 double estimPi(long inNbPts)
13 {
14     int cpt = 0;
15     double Xr;
16     double Yr;
17
18     for(int i = 0; i<inNbPts;i++)
19     {
20         Xr = genrand_real1();
21         Yr = genrand_real1();
22         if((Xr*Xr) + (Yr*Yr) < 1 ){cpt++;}
23
24     }
25     return((double) cpt/(double)inNbPts *4);
26 }
27
28
29 /* -----
30  * ----- */
30 /* meanPi renvoie une moyenne des estimations de pi
31 */
32 /*
33 /* En entrée: nombre d'iterations de type int*/
34 /*
35 /* En sortie: une moyenne des estimations de pi*/
36 /*
36 double meanPi(int iter)
```

```
37 {
38
39     double Bin[iter];
40     double somme = 0.0;
41     for(int i = 0; i < iter; i++)
42     {
43         Bin[i] = estimPi(10000000);
44         somme +=Bin[i];
45
46         //printf(" Pi = %f ",Bin[i]); // debugage
47         //printf("total = %f %d \n",somme , i
48     ); // debugage
49 }
50
51
52 }
53
54 /* -----
----- */
55 /* relativeError renvoie l'erreure relative entre
   le pi estimé et Pi*/
56 /*
57 /* En entrée: rien*/
58 /*
59 /* En sortie: l'erreure relative entre pi estimé et
   Pi*/
60 /* -----
----- */
61
62 double relativeError()
63 {
64     return((meanPi(30) - M_PI)/M_PI);
65 }
66
67 /* -----
----- */
68 /* StudentLaw renvoie l'intervale de confiance*/
69 /*
70 /* En entrée: n de type int */
71 /*
72 /* En sortie: l'intervale de confiance */
```

```
73 /* -----
   ----- */
74
75 double StudentLaw(int n)
76 {
77     double valuesTab[30] = {12.706, 4.303, 3.182, 2.
776, 2.571, 2.447, 2.365, 2.308, // Tableau de la
    lois de student
78                         2.262, 2.228, 2.201, 2.179
    , 2.160, 2.145, 2.131, 2.120,
79                         2.110, 2.101, 2.093, 2.086
    , 2.080, 2.074, 2.069, 2.064,
80                         2.060, 2.056, 2.052, 2.048
    , 2.045, 2.042};

81
82     double value = valuesTab[n-1];
83     double tabPi[n];
84     double mean = meanPi(n);
85     double somme = 0.0;
86     double S2 = 0.0;
87     double R = 0.0;
88
89     if (n>30 && n<=40) {value = 2.021;}
90     if (n>40 && n<=80) {value = 2.000;}
91     if (n>80 && n<=120) {value = 1.980;}
92     if (n>120) {value = 1.960;}
93
94     for (int i = 0; i <n; i++)
95     {
96         tabPi[i] = estimPi(1000000000);
97
98         somme += (tabPi[i] - mean)*(tabPi[i] -
mean);
99     }
100
101
102
103     S2 = (somme/(n-1));
104
105
106     R = (value * (sqrt(S2/n)));
107
```

```
108     return(R);
109 }
110
111 /* -----
112    ----- */
113 /* intervaleConfiance affiche l'intervalle de confiance entre deux estimations de pi*/>
114 /* En entrée: n de type int */
115 /*
116 /* En sortie: void*/
117 /* -----
118    ----- */
119 {
120     double pi = meanPi(30);
121     double student = StudentLaw(n);
122     printf("Notre intervalle de confiance est entre
123           : %.10lf et %.10lf avec n = %d \n", pi + student
124           , pi - student, n);
125 }
126 /* -----
127    ----- */
128 /* graphe permet de faire un graphe des intervalles de confiance*/>
129 /*
130 /* En sortie: void*/
131 /* -----
132    ----- */
133 {
134     for(int i = x; i <y; i++)
135     {
136         intervalleConfiance(i);
137     }
138 }
```

SIMULATEUR PSEUDO ALÉATOIRE D'UNE POPULATION DE LAPINS

Introduction:

Le japon est un pays formidable, pour les geeks, les amateurs de sushis, les amateurs de randonnées et de beaux paysages mais aussi pour les férus de petites bêtes trop kawaii **tashirojima** et **okunoshima** sont deux îles uniques en leurs genre l'une est peuplée de chats et l'autre de lapins et aujourd'hui nous allons nous intéresser à la deuxième.

Dans le cadre de ce projet on se propose de faire une simulation sur plusieurs générations d'une population de lapins.

Nous allons en premier lieu demandé à l'utilisateur d'entrer le nombre de lapins souhaité ainsi que la durée de la simulation.

Ceci a fin d'afficher par la suite le résultat final des lapins ainsi qu'une moyenne et la marge d'erreur de notre programme.

I°) Nos Prédicats:

- Un lapin peut être soit male soit femelle (50%)
- Une femelle a 3 à 9 portées par an (en cloche).
- Chaque année 10% des lapines sont infertiles
- Une portée est de 3 à 5 bébés et plus rarement de 2 ou 6
- La maturité sexuelle est entre 5 et 8 mois
- Les jeunes lapins ont 50% de chance de survivre et Les lapins matures 75%.
- A partir de 7 ans les chances de survie diminuent de 15% par an jusqu'à atteindre 0% à 12 ans

II°) Rabbit:

La première fonction de la classe Rabbit porte le même nom que celle-ci et permet l'initialisation de nos lapins. Elle prend comme entrée le sexe du lapin, son âge et sa maturité pour ensuite vérifier sa stérilité si c'est une femelle et lui donner un nombre de portée initiale.

La seconde fonction **setAge** permet d'incrémenter l'age du lapin en mois de le rendre mature ou pas et pour les femelles de vérifier s'ils ne sont pas stériles à fin de les activer sexuellement.

La troisième fonction **setChancesSurvie** permet de modifier les chances de survie de nos lapins selon leur âge suivant les prédictats.

Les quatrième et cinquième fonctions **setPortée** et **reinitPortée** permettent dans l'ordre de décrémenter le nombre de portée et de la réinitialiser.

La sixième et dernière fonction de notre classe **Rabbit** permet de compter à elle de tuer les lapins.

III°) Main:

Notre classe principale est divisée en quatre parties:

Lapin :

La fonction **lapin** prend en entrée le nombre de lapins mâles, le nombre de lapines ainsi que la durée de la simulation en année.

On va commencer par créer une liste de lapins mâles et femelles.

Ensuite chaque mois nous allons incrémenter l'âge de tous nos lapins grâce à la fonction **setAge**, enlever les lapins morts de la liste, et nous allons, pour chaque lapine, enclencher la naissance (la création) des bébés, réinitialiser leur portée initiale.

Enfin chaque année on réinitialise le nombre de portée de chaque femelle, on vérifie les chances de survie de chaque lapin et on élimine les lapins qui n'ont pas eu de chance.

La fonction **lapin** retourne le total des lapins sur la période donnée.

Mean :

La fonction **Mean** permet d'avoir la moyenne de plusieurs estimations de notre nombre de lapins final.

relativeError :

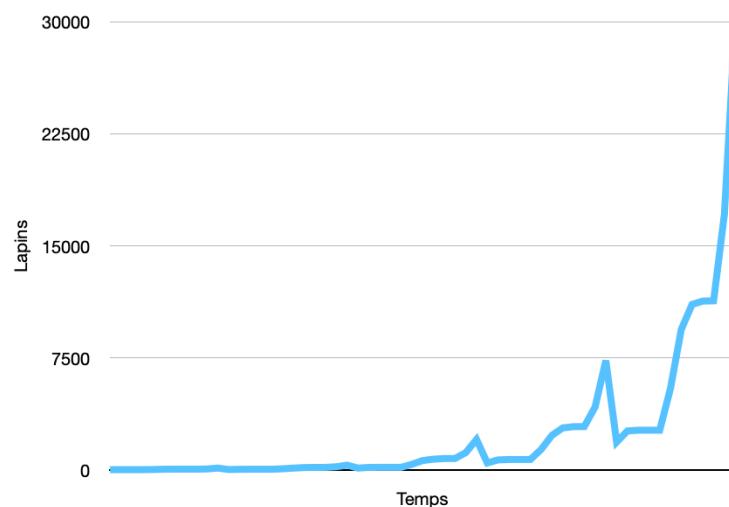
Comme son nom l'indique cette fonction permet d'avoir la marge d'erreur de notre programme soit la différence entre notre résultat et le résultat escompté

Main :

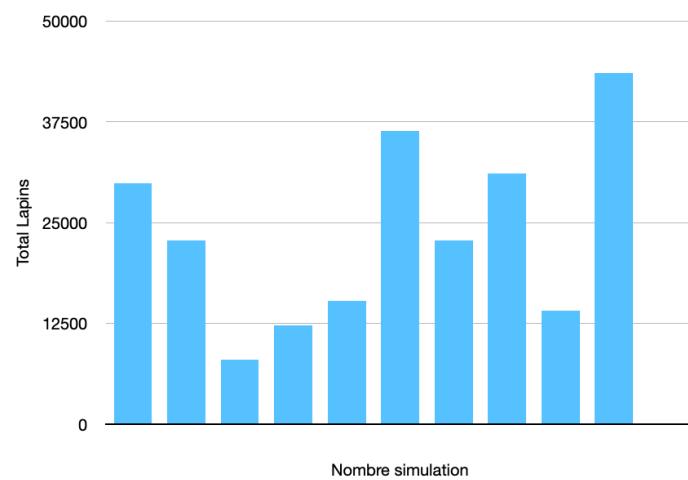
La fonction principale permet à l'utilisateur d'entrer le nombre de males de femelles ainsi que le nombre d'année.
Elle affiche les résultats à la fin.

IV°) Observations:

L'on remarque que sur 5 ans avec un male et une femelle le nombre de lapins augmente de façon exponentielle



Sur dix simulations:



Main.java

```
1 //Par Mohamed Damergi
2 package com.company;
3
4 import java.util.ArrayList;
5 import java.util.Scanner;
6
7 public class Main {
8     private static final MTRandom MT;
9
10    static {
11        MT = new MTRandom();
12    }
13
14
15    public static long lapin(int m, int f, int a){
16
17
18
19        //Bloc d'entrée
20        ArrayList<Rabbit> Rabbits = new ArrayList<>()
21            for (int i = 0; i < m; i++) {
22                Rabbits.add(new Rabbit('m', 12, true));
23            }
24            for (int i = 0; i < f; i++) {
25                Rabbits.add(new Rabbit('f', 12, true));
26            }
27
28
29        //Bloc Principale
30        for (int i = 1; i <= a * 12; i++) { // De
31            for (int k = 0; k < Rabbits.size(); k++)
32                Rabbits.get(k).setAge(); // incre
33                Rabbits.removeIf(rabbit -> !rabbit
34            }
35            for (int k = 0; k < Rabbits.size(); k++)
36                if (Rabbits.get(k).sexe == 'f' &&
37                    double y = MT.nextDouble();
38
39                    int n = 0;
40                    if (y <= 0.3) {
41                        n = 4;
42                    }
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039
5040
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054
5055
5056
5057
5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
50100
50101
50102
50103
50104
50105
50106
50107
50108
50109
50110
50111
50112
50113
50114
50115
50116
50117
50118
50119
50120
50121
50122
50123
50124
50125
50126
50127
50128
50129
50130
50131
50132
50133
50134
50135
50136
50137
50138
50139
50140
50141
50142
50143
50144
50145
50146
50147
50148
50149
50150
50151
50152
50153
50154
50155
50156
50157
50158
50159
50160
50161
50162
50163
50164
50165
50166
50167
50168
50169
50170
50171
50172
50173
50174
50175
50176
50177
50178
50179
50180
50181
50182
50183
50184
50185
50186
50187
50188
50189
50190
50191
50192
50193
50194
50195
50196
50197
50198
50199
50199
50200
50201
50202
50203
50204
50205
50206
50207
50208
50209
50210
50211
50212
50213
50214
50215
50216
50217
50218
50219
50220
50221
50222
50223
50224
50225
50226
50227
50228
50229
50230
50231
50232
50233
50234
50235
50236
50237
50238
50239
50240
50241
50242
50243
50244
50245
50246
50247
50248
50249
50250
50251
50252
50253
50254
50255
50256
50257
50258
50259
50260
50261
50262
50263
50264
50265
50266
50267
50268
50269
50270
50271
50272
50273
50274
50275
50276
50277
50278
50279
50280
50281
50282
50283
50284
50285
50286
50287
50288
50289
50290
50291
50292
50293
50294
50295
50296
50297
50298
50299
50299
50300
50301
50302
50303
50304
50305
50306
50307
50308
50309
503010
503011
503012
503013
503014
503015
503016
503017
503018
503019
503020
503021
503022
503023
503024
503025
503026
503027
503028
503029
503030
503031
503032
503033
503034
503035
503036
503037
503038
503039
5030310
5030311
5030312
5030313
5030314
5030315
5030316
5030317
5030318
5030319
5030320
5030321
5030322
5030323
5030324
5030325
5030326
5030327
5030328
5030329
5030330
5030331
5030332
5030333
5030334
5030335
5030336
5030337
5030338
5030339
50303310
50303311
50303312
50303313
50303314
50303315
50303316
50303317
50303318
50303319
50303320
50303321
50303322
50303323
50303324
50303325
50303326
50303327
50303328
50303329
50303330
50303331
50303332
50303333
50303334
50303335
50303336
50303337
50303338
50303339
503033310
503033311
503033312
503033313
503033314
503033315
503033316
503033317
503033318
503033319
503033320
503033321
503033322
503033323
503033324
503033325
503033326
503033327
503033328
503033329
503033330
503033331
503033332
503033333
503033334
503033335
503033336
503033337
503033338
503033339
5030333310
5030333311
5030333312
5030333313
5030333314
5030333315
5030333316
5030333317
5030333318
5030333319
5030333320
5030333321
5030333322
5030333323
5030333324
5030333325
5030333326
5030333327
5030333328
5030333329
5030333330
5030333331
5030333332
5030333333
5030333334
5030333335
5030333336
5030333337
5030333338
5030333339
50303333310
50303333311
50303333312
50303333313
50303333314
50303333315
50303333316
50303333317
50303333318
50303333319
50303333320
50303333321
50303333322
50303333323
50303333324
50303333325
50303333326
50303333327
50303333328
50303333329
50303333330
50303333331
50303333332
50303333333
50303333334
50303333335
50303333336
50303333337
50303333338
50303333339
503033333310
503033333311
503033333312
503033333313
503033333314
503033333315
503033333316
503033333317
503033333318
503033333319
503033333320
503033333321
503033333322
503033333323
503033333324
503033333325
503033333326
503033333327
503033333328
503033333329
503033333330
503033333331
503033333332
503033333333
503033333334
503033333335
503033333336
503033333337
503033333338
503033333339
5030333333310
5030333333311
5030333333312
5030333333313
5030333333314
5030333333315
5030333333316
5030333333317
5030333333318
5030333333319
5030333333320
5030333333321
5030333333322
5030333333323
5030333333324
5030333333325
5030333333326
5030333333327
5030333333328
5030333333329
5030333333330
5030333333331
5030333333332
5030333333333
5030333333334
5030333333335
5030333333336
5030333333337
5030333333338
5030333333339
50303333333310
50303333333311
50303333333312
50303333333313
50303333333314
50303333333315
50303333333316
50303333333317
50303333333318
50303333333319
50303333333320
50303333333321
50303333333322
50303333333323
50303333333324
50303333333325
50303333333326
50303333333327
50303333333328
50303333333329
50303333333330
50303333333331
50303333333332
50303333333333
50303333333334
50303333333335
50303333333336
50303333333337
50303333333338
50303333333339
503033333333310
503033333333311
503033333333312
503033333333313
503033333333314
503033333333315
503033333333316
503033333333317
503033333333318
503033333333319
503033333333320
503033333333321
503033333333322
503033333333323
503033333333324
503033333333325
503033333333326
503033333333327
503033333333328
503033333333329
503033333333330
503033333333331
503033333333332
503033333333333
503033333333334
503033333333335
503033333333336
503033333333337
503033333333338
503033333333339
5030333333333310
5030333333333311
5030333333333312
5030333333333313
5030333333333314
5030333333333315
5030333333333316
5030333333333317
5030333333333318
5030333333333319
5030333333333320
5030333333333321
5030333333333322
5030333333333323
5030333333333324
5030333333333325
5030333333333326
5030333333333327
5030333333333328
5030333333333329
5030333333333330
5030333333333331
5030333333333332
5030333333333333
5030333333333334
5030333333333335
5030333333333336
5030333333333337
5030333333333338
5030333333333339
50303333333333310
50303333333333311
50303333333333312
50303333333333313
50303333333333314
50303333333333315
50303333333333316
50303333333333317
50303333333333318
50303333333333319
50303333333333320
50303333333333321
50303333333333322
50303333333333323
50303333333333324
50303333333333325
50303333333333326
50303333333333327
50303333333333328
50303333333333329
50303333333333330
50303333333333331
50303333333333332
50303333333333333
50303333333333334
50303333333333335
50303333333333336
50303333333333337
50303333333333338
50303333333333339
503033333333333310
503033333333333311
503033333333333312
503033333333333313
503033333333333314
503033333333333315
503033333333333316
503033333333333317
503033333333333318
503033333333333319
503033333333333320
503033333333333321
503033333333333322
503033333333333323
503033333333333324
503033333333333325
503033333333333326
503033333333333327
503033333333333328
503033333333333329
503033333333333330
503033333333333331
503033333333333332
503033333333333333
503033333333333334
503033333333333335
503033333333333336
503033333333333337
503033333333333338
503033333333333339
5030333333333333310
5030333333333333311
5030333333333333312
5030333333333333313
5030333333333333314
5030333333333333315
5030333333333333316
5030333333333333317
5030333333333333318
5030333333333333319
5030333333333333320
5030333333333333321
5030333333333333322
5030333333333333323
5030333333333333324
5030333333333333325
5030333333333333326
5030333333333333327
5030333333333333328
5030333333333333329
5030333333333333330
5030333333333333331
5030333333333333332
5030333333333333333
5030333333333333334
5030333333333333335
5030333333333333336
5030333333333333337
5030333333333333338
5030333333333333339
50303333333333333310
50303333333333333311
50303333333333333312
50303333333333333313
50303333333333333314
50303333333333333315
50303333333333333316
50303333333333333317
50303333333333333318
50303333333333333319
50303333333333333320
50303333333333333321
50303333333333333322
50303333333333333323
50303333333333333324
50303333333333333325
50303333333333333326
50303333333333333327
50303333333333333328
50303333333333333329
50303333333333333330
50303333333333333331
50303333333333333332
50303333333333333333
50303333333333333334
50303333333333333335
50303333333333333336
50303333333333333337
50303333333333333338
50303333333333333339
503033333333333333310
503033333333333333311
503033333333333333312
503033333333333333313
503033333333333333314
503033333333333333315
503033333333333333316
503033333333333333317
503033333333333333318
503033333333333333319
503033333333333333320
503033333333333333321
503033333333333333322
503033333333333333323
503033333333333333324
503033333333333333325
503033333333333333326
503033333333333333327
503033333333333333328
503033333333333333329
503033333333333333330
503033333333333333331
503033333333333333332
503033333333333333333
503033333333333333334
503033333333333333335
503033333333333333336
503033333333333333337
503033333333333333338
503033333333333333339
5030333333333333333310
5030333333333333333311
5030333333333333333312
5030333333333333333313
5030333333333333333314
5030333333333333333315
5030333333333333333316
5030333333333333333317
5030333333333333333318
5030333333333333333319
5030333333333333333320
5030333333333333333321
5030333333333333333322
5030333333333333333323
5030333333333333333324
5030333333333333333325
5030333333333333333326
5030333333333333333327
5030333333333333333328
5030333333333333333329
5030333333333333333330
5030333333333333333331
5030333333333333333332
5030333333333333333333
5030333333333333333334
5030333333333333333335
5030333333333333333336
5030333333333333333337
5030333333333333333338
5030333333333333333339
50303333333333333333310
50303333333333333333311
50303333333333333333312
50303333333333333333313
50303333333333333333314
50303333333333333333315
50303333333333333333316
50303333333333333333317
50303333333333333333318
50303333333333333333319
50303333333333333333320
50303333333333333333321
50303333333333333333322
50303333333333333333323
50303333333333333333324
50303333333333333333325
50303333333333333333326
50303333333333333333327
50303333333333333333328
50303333333333333333329
50303333333333333333330
50303333333333333333331
50303333333333333333332
50303333333333333333333
50303333333333333333334
50303333333333333333335
50303333333333333333336
50303333333333333333337
50303333333333333333338
50303333333333333333339
503033333333333333333310
503033333333333333333311
503033
```

```
43                     if (y > 0.3 && y <= 0.5) {
44                         n = 3;
45                     }
46                     if (y > 0.5 && y <= 0.7) {
47                         n = 5;
48                     }
49                     if (y > 0.7 && y <= 0.85) {
50                         n = 2;
51                     }
52                     if (y > 0.85) {
53                         n = 6;
54                     }
55
56                 for (int l = 0; l < n; l++) {
57
58                     double x = MT.nextDouble();
59                     if (x <= 0.5) {
60
61                         Rabbits.add(new Rabbi());
62                         Rabbits.get(k).setPortee(x);
63                     } else if (x > 0.5) {
64
65                         Rabbits.add(new Rabbi());
66                         Rabbits.get(k).setPortee(x);
67                     }
68                 }
69             }
70         }
71         if (i % 12 == 0) {
72             Rabbits.get(k).reinitPortee();
73             Rabbits.get(k).setChanceSurvival();
74             Rabbits.get(k).kill();
75         }
76     }
77 }
78 return Rabbits.size();
79 }
80 static double mean(int m, int f, int a)
81 {
82     long[] Bin = new long[20];
83     double somme = 0.0;
```

```
85         for(int i = 0; i < 20; i++)
86         {
87             Bin[i] = lapin(m,f,a);
88             somme +=Bin[i];
89         }
90         return (somme/ 20);
91     }
92
93     static double relativeError(int m, int f, int a)
94     {
95         return((mean(m,f,a) - lapin(m,f,a))/lapin(m,
96     })
97
98     public static void main(String[] args) {
99
100        int[] init = {0x123, 0x234, 0x345, 0x456};
101        MT.setSeed(init);
102
103        int nbMales;
104        int nbFemales;
105        int nbAnnee;
106        Scanner input = new Scanner(System.in);
107
108        System.out.println("Nombre de lapins males ?");
109        nbMales = input.nextInt();
110
111
112        System.out.println("Nombre de lapins femelle");
113        nbFemales = input.nextInt();
114
115        System.out.println("Nombre d'année que va du");
116        nbAnnee = input.nextInt();
117        for(int i=0; i<10; i++) {
118            System.out.println("Sur une periode de ");
119            System.out.println(lapin(nbMales, nbFema
120            System.out.println("une moyenne de " + m
121            System.out.println("et un taux d'erreur
122        }
123
124    }
125 }
```

```
1 //Par Mohamed Damergi
2 package com.company;
3
4
5 public class Rabbit {
6     //Age
7     int ageMois;
8     int ageAnnee;
9     char sexe;
10    boolean isMature;
11    boolean isActiveSex = false;
12    double chanceSurvie = 0.5;
13    boolean isAlive = true;
14
15    //Les femelles
16    boolean isSterile = false;
17    int nbPortee = 0;
18    int nbPorteeInit;
19
20
21    MTRandom MT = new MTRandom();
22
23
24    /*****
25        Creation de nos lapins
26
27        cette fonction permet de creer nos lapins elle p
28        mois ainsi que ca maturité
29
30        elle nous permet aussi de definir si le lapin es
31        un nombre de portée aux femelles
32
33        elle ne retourne rien
34    *****
35    public Rabbit(char sexe, int ageMois,boolean isMa
36        int[] init ={0x123, 0x234, 0x345, 0x456};
37        MT.setSeed(init);
38
39        this.sex = sexe;
40        this.ageMois = ageMois;
41        this.isMature = isMature;
42        double x = MT.nextDouble();
```

```
43     double y = MT.nextDouble();
44     if(this.sex == 'f'){
45         if(x<0.1){
46             this.isSterile = true;
47         }
48         if(!this.isSterile){
49
50             if(y<=0.4){
51                 this.nbPorteeInit = 6;
52             }
53             if(y>0.4 && y<=0.7){
54                 boolean b = MT.nextBoolean();
55                 if(b){
56                     this.nbPorteeInit = 5;
57                 }
58                 if(!b){
59                     this.nbPorteeInit = 7;
60                 }
61             }
62             if(y>0.7 && y<=0.9){
63                 boolean b = MT.nextBoolean();
64                 if(b){
65                     this.nbPorteeInit = 4;
66                 }
67                 if(!b){
68                     this.nbPorteeInit = 8;
69                 }
70                 this.nbPorteeInit = 4;
71             }
72             if(y>0.9){
73                 boolean b = MT.nextBoolean();
74                 if(b){
75                     this.nbPorteeInit = 3;
76                 }
77                 if(!b){
78                     this.nbPorteeInit = 9;
79                 }
80             }
81             this.nbPortee = this.nbPorteeInit;
82     }
```

```
85         }
86     }
87
88     /*****
89      * modificateur d'age
90      * Cette fonction permet d'incrementer l'age de no
91      * de les activer sexuellement.
92
93      * elle ne prend rien en entrée
94      * et ne retourne rien
95     *****/
96     public void setAge(){ //en mois
97         double x = MT.nextDouble();
98
99         this.ageMois += 1;
100        this.ageAnnee = this.ageMois/12;
101
102        if(this.ageMois == 5) {
103            if (x <= 0.2) {
104                isMature = true;
105                if(!isSterile){
106                    isActiveSex = true;
107                }
108            }
109        }
110        else if(this.ageMois == 6){
111            if (x >0.2 && x <= 0.5){
112                isMature = true;
113                if(!isSterile){
114                    isActiveSex = true;}
115            }
116        }
117        else if(this.ageMois == 7){
118            if(x > 0.5 && x <= 0.8){
119                isMature = true;
120                if(!isSterile){
121                    isActiveSex = true;}
122            }
123        }
124        else if(this.ageMois > 8){
125            if(x > 0.8){
126                isMature = true;
```

```
127             isActiveSex = true;
128         }
129     }
130 }
131
132 /**
133     configuration des chance de survie
134     Cette fonction permet de modifier les chances d
135
136     elle ne prend rien en entrée
137     et ne retourne rien
138 */
139 public void setChanceSurvie(){
140
141     if(this.isMature){
142         this.chanceSurvie = 0.75;
143     }
144     switch (this.ageAnnee) {
145         case 8 -> this.chanceSurvie = 0.6;
146         case 9 -> this.chanceSurvie = 0.45;
147         case 10 -> this.chanceSurvie = 0.3;
148         case 11 -> this.chanceSurvie = 0.15;
149         case 12 -> this.chanceSurvie = 0;
150     }
151 }
152 /**
153     decrementation du nombre de portée
154     Cette fonction permet de decrementer le nombre
155 */
156 public void setPortee(){
157     this.nbPortee -=1;
158 }
159
160 /**
161     reinitialisation du nombre de porté
162     cette fonction reinitialise le nombre de portée
163 */
164 public void reinitPortee() {
165     this.nbPortee = this.nbPorteeInit;
166 }
167 /**
168     la faucheuse
```

```
169      cette fonction tue les lapins
170      ****
171
172  public void kill(){
173
174      if(MT.nextDouble() >this.chanceSurvie){
175          isAlive = false;
176      }
177
178  }
179 }
180
181
182
183
184
```

```
1 /*
2  * MTRandom : A Java implementation of the MT19937 (M
3  *             pseudo random number generator algorithm
4  *             original C code by Makoto Matsumoto and
5  * Author   : David Beaumont
6  * Email    : mersenne-at-www.goui.net
7  *
8  * For the original C code, see:
9  *     http://www.math.sci.hiroshima-u.ac.jp/~m-mat/M
10 *
11 * This version, Copyright (C) 2005, David Beaumont.
12 *
13 * This library is free software; you can redistribute
14 * modify it under the terms of the GNU Lesser Genera
15 * License as published by the Free Software Foundati
16 * version 2.1 of the License, or (at your option) an
17 *
18 * This library is distributed in the hope that it wi
19 * but WITHOUT ANY WARRANTY; without even the implied
20 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE
21 * Lesser General Public License for more details.
22 *
23 * You should have received a copy of the GNU Lesser
24 * License along with this library; if not, write to
25 * Foundation, Inc., 51 Franklin St, Fifth Floor, Bos
26 */
27
28 package com.company;
29
30 import java.util.Random;
31
32 /**
33 * @version 1.0
34 * @author David Beaumont, Copyright 2005
35 * <p>
36 * A Java implementation of the MT19937 (Mersenne Twi
37 * random number generator algorithm based upon the o
38 * by Makoto Matsumoto and Takuji Nishimura (see
39 * <a href="http://www.math.sci.hiroshima-u.ac.jp/~m-
40 * http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/em
41 * more information.
42 * <p>
```

```

43 * As a subclass of java.util.Random this class provi
44 * canonical method next() for generating bits in the
45 * number sequence. Anyone using this class should i
46 * inherited methods (nextInt(), nextFloat etc.) to o
47 * normal. This class should provide a drop-in repla
48 * standard implementation of java.util.Random with t
49 * advantage of having a far longer period and the ab
50 * far larger seed value.
51 * <p>
52 * This is not a cryptographically strong sour
53 * and should not be used for cryptographic sy
54 * other situation where true random numbers are requ
55 * <p>
56 * <!-- Creative Commons License -->
57 * <a href="http://creativecommons.org/licenses/LGPL/
58 * This software is licensed under the <a href="http:
59 * <!-- /Creative Commons License -->
60 *
61 * <!--
62 * <rdf:RDF xmlns="http://web.resource.org/cc/"
63 *     xmlns:dc="http://purl.org/dc/elements/1.1/"
64 *     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-sy
65 *
66 * <Work rdf:about="">
67 *   <license rdf:resource="http://creativecommons.o
68 *     <dc:type rdf:resource="http://purl.org/dc/dcmi/t
69 * </Work>
70 *
71 * <License rdf:about="http://creativecommons.org/lic
72 *   <permits rdf:resource="http://web.resource.org/
73 *   <permits rdf:resource="http://web.resource.org/
74 *   <requires rdf:resource="http://web.resource.org/
75 *   <permits rdf:resource="http://web.resource.org/
76 *   <requires rdf:resource="http://web.resource.org/
77 *   <requires rdf:resource="http://web.resource.org/
78 * </License>
79 *
80 * </rdf:RDF>
81 * -->
82 *
83 */
84 public class MTRandom extends Random {

```

```

85
86     /**
87      * Auto-generated serial version UID. Note that
88      * support serialisation of its internal state a
89      * necessary to implement read/write methods to
90      * This is only here to make Eclipse shut up abo
91      */
92     private static final long serialVersionUID = -51
93
94     // Constants used in the original C implementati
95     private final static int UPPER_MASK = 0x80000000
96     private final static int LOWER_MASK = 0xffffffff
97
98     private final static int N = 624;
99     private final static int M = 397;
100    private final static int MAGIC[] = { 0x0, 0x9908
101    private final static int MAGIC_FACTOR1 = 1812433
102    private final static int MAGIC_FACTOR2 = 1664525
103    private final static int MAGIC_FACTOR3 = 1566083
104    private final static int MAGIC_MASK1 = 0x9d2c5
105    private final static int MAGIC_MASK2 = 0xefc60
106    private final static int MAGIC_SEED = 1965021
107    private final static long DEFAULT_SEED = 5489L;
108
109    // Internal state
110    private transient int[] mt;
111    private transient int mti;
112    private transient boolean compat = false;
113
114    // Temporary buffer used during setSeed(long)
115    private transient int[] ibuf;
116
117    /**
118     * The default constructor for an instance of MT
119     * the no-argument constructor for java.util.Ran
120     * in the class being initialised with a seed va
121     * System.currentTimeMillis().
122     */
123    public MTRandom() { }
124
125    /**
126     * This version of the constructor can be used t

```

```

127     * behaviour to the original C code version of t
128     * exactly replicating the case where the seed v
129     * prior to calling genrand_int32.
130     * <p>
131     * If the compatibility flag is set to true, the
132     * seeded with the same default value as was use
133     * code. Furthermore the setSeed() method, whic
134     * long value, will be limited to using only the
135     * seed to facilitate seamless migration of exis
136     * where identical behaviour is required.
137     * <p>
138     * Whilst useful for ensuring backwards compatib
139     * that this feature not be used unless specific
140     * the reduction in strength of the seed value.
141     *
142     * @param compatible Compatibility flag for repl
143     * behaviour.
144     */
145     public MTRandom(boolean compatible) {
146         super(0L);
147         compat = compatible;
148         setSeed(compat?DEFAULT_SEED:System.currentTimeMillis());
149     }
150
151     /**
152     * This version of the constructor simply initia
153     * the given 64 bit seed value. For a better ra
154     * this seed value should contain as much entrop
155     *
156     * @param seed The seed value with which to init
157     */
158     public MTRandom(long seed) {
159         super(seed);
160     }
161
162     /**
163     * This version of the constructor initialises t
164     * given byte array. All the data will be used
165     * instance.
166     *
167     * @param buf The non-empty byte array of seed i
168     * @throws NullPointerException if the buffer is

```

```
169     * @throws IllegalArgumentException if the buffer
170     */
171     public MTRandom(byte[] buf) {
172         super(0L);
173         setSeed(buf);
174     }
175
176     /**
177      * This version of the constructor initialises the
178      * given integer array. All the data will be used
179      * by this instance.
180      *
181      * @param buf The non-empty integer array of seed
182      * @throws NullPointerException if the buffer is null
183      * @throws IllegalArgumentException if the buffer
184      */
185     public MTRandom(int[] buf) {
186         super(0L);
187         setSeed(buf);
188     }
189
190     // Initializes mt[N] with a simple integer seed.
191     // required as part of the Mersenne Twister algo
192     // not be made public.
193     private final void setSeed(int seed) {
194
195         // Annoying runtime check for initialisation
196         // caused by java.util.Random invoking setSeed
197         // This is unavoidable because no fields in
198         // have been initialised at this point, not
199         // were placed at the declaration of the memory
200         if (mt == null) mt = new int[N];
201
202         // ---- Begin Mersenne Twister Algorithm ---
203         mt[0] = seed;
204         for (mti = 1; mti < N; mti++) {
205             mt[mti] = (MAGIC_FACTOR1 * (mt[mti-1] ^
206             })
207             // ---- End Mersenne Twister Algorithm ---
208         }
209
210     /**

```

```
211     * This method resets the state of this instance  
212     * bits of seed data provided. Note that if the  
213     * is passed to two different instances of MTRan  
214     * which share the same compatibility state) the  
215     * of numbers generated by both instances will b  
216     * <p>  
217     * If this instance was initialised in 'compatib  
218     * this method will only use the lower 32 bits o  
219     * passed in and will match the behaviour of the  
220     * exactly with respect to state initialisation.  
221     *  
222     * @param seed The 64 bit value used to initiali  
223     * number generator state.  
224     */  
225     public final synchronized void setSeed(long seed  
226         if (compat) {  
227             setSeed((int)seed);  
228         } else {  
229             // Annoying runtime check for initialisa  
230             // caused by java.util.Random invoking s  
231             // This is unavoidable because no fields  
232             // have been initialised at this point,  
233             // were placed at the declaration of the  
234             // if (ibuf == null) ibuf = new int[2];  
235             if (ibuf == null) ibuf = new int[2];  
236             ibuf[0] = (int)seed;  
237             ibuf[1] = (int)(seed >>> 32);  
238             setSeed(ibuf);  
239         }  
240     }  
241  
242     /**  
243     * This method resets the state of this instance  
244     * array of seed data provided. Note that calli  
245     * is equivalent to calling "setSeed(pack(buf))"  
246     * will result in a new integer array being gene  
247     * call. If you wish to retain this seed data t  
248     * random sequence to be restarted then it would  
249     * to use the "pack()" method to convert it into  
250     * first and then use that to re-seed the instan  
251     * of the class will be the same in both cases b
```

```

253     * efficient.
254     *
255     * @param buf The non-empty byte array of seed i
256     * @throws NullPointerException if the buffer is
257     * @throws IllegalArgumentException if the buffer
258     */
259     public final void setSeed(byte[] buf) {
260         setSeed(pack(buf));
261     }
262
263     /**
264      * This method resets the state of this instance
265      * array of seed data provided. This is the can
266      * resetting the pseudo random number sequence.
267      *
268      * @param buf The non-empty integer array of seed
269      * @throws NullPointerException if the buffer is
270      * @throws IllegalArgumentException if the buffer
271      */
272     public final synchronized void setSeed(int[] buf
273         int length = buf.length;
274         if (length == 0) throw new IllegalArgumentExceptionEx
275         // ---- Begin Mersenne Twister Algorithm ---
276         int i = 1, j = 0, k = (N > length ? N : length);
277         setSeed(MAGIC_SEED);
278         for (; k > 0; k--) {
279             mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>
280                 i++; j++;
281                 if (i >= N) { mt[0] = mt[N-1]; i = 1; }
282                 if (j >= length) j = 0;
283             }
284             for (k = N-1; k > 0; k--) {
285                 mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>
286                     i++;
287                     if (i >= N) { mt[0] = mt[N-1]; i = 1; }
288                 }
289                 mt[0] = UPPER_MASK; // MSB is 1; assuring no
290                 // ---- End Mersenne Twister Algorithm ---
291             }
292
293             /**
294              * This method forms the basis for generating a

```

```

295     * sequence from this class. If given a value o
296     * behaves identically to the genrand_int32 func
297     * C code and ensures that using the standard ne
298     * (inherited from Random) we are able to replic
299     * <p>
300     * Note that where the number of bits requested
301     * then bits will simply be masked out from the
302     * integer value. That is to say that:
303     * <pre>
304     * mt.setSeed(12345);
305     * int foo = mt.nextInt(16) + (mt.nextInt(16) <<
306     * will not give the same result as
307     * <pre>
308     * mt.setSeed(12345);
309     * int foo = mt.nextInt(32);</pre>
310     *
311     * @param bits The number of significant bits de
312     * @return The next value in the pseudo random s
313     * specified number of bits in the lower part of
314     */
315     protected final synchronized int next(int bits)
316         // ---- Begin Mersenne Twister Algorithm ---
317         int y, kk;
318         if (mti >= N) {                      // generate N wo
319
320             // In the original C implementation, mti
321             // to determine if initialisation has oc
322             // it initialises this instance with DEF
323             // This is no longer necessary as initia
324             // Java instance must result in initiali
325             // Use the constructor MTRandom(true) to
326             // compatible behaviour.
327
328             for (kk = 0; kk < N-M; kk++) {
329                 y = (mt[kk] & UPPER_MASK) | (mt[kk+1]
330                     mt[kk] = mt[kk+M] ^ (y >>> 1) ^ MAGI
331             }
332             for (;kk < N-1; kk++) {
333                 y = (mt[kk] & UPPER_MASK) | (mt[kk+1]
334                     mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^
335             }
336             y = (mt[N-1] & UPPER_MASK) | (mt[0] & LO

```

```

337         mt[N-1] = mt[M-1] ^ (y >>> 1) ^ MAGIC[y
338
339         mti = 0;
340     }
341
342     y = mt[mti++];
343
344     // Tempering
345     y ^= (y >>> 11);
346     y ^= (y << 7) & MAGIC_MASK1;
347     y ^= (y << 15) & MAGIC_MASK2;
348     y ^= (y >>> 18);
349     // ---- End Mersenne Twister Algorithm -----
350     return (y >>> (32-bits));
351 }
352
353 // This is a fairly obscure little code section
354 // byte[] into an int[] in little endian orderin
355
356 /**
357 * This simply utility method can be used in cas
358 * array of seed data is to be used to repeatedl
359 * random number sequence. By packing the byte
360 * integer array first, using this method, and t
361 * setSeed() with that; it removes the need to r
362 * array each time setSeed() is called.
363 * <p>
364 * If the length of the byte array is not a mult
365 * it is implicitly padded with zeros as necessa
366 * <pre> byte[] { 0x01, 0x02, 0x03, 0x04, 0x0
367 * becomes
368 * <pre> int[] { 0x04030201, 0x00000605 }</p
369 * <p>
370 * Note that this method will not complain if th
371 * is empty and will produce an empty integer ar
372 * setSeed() method will throw an exception if t
373 * array is passed to it.
374 *
375 * @param buf The non-null byte array to be pack
376 * @return A non-null integer array of the packe
377 * @throws NullPointerException if the given byt
378 */

```

```
379     public static int[] pack(byte[] buf) {
380         int k, blen = buf.length, ilen = ((buf.length + 3) / 4) * 4;
381         int[] ibuf = new int[ilen];
382         for (int n = 0; n < ilen; n++) {
383             int m = (n+1) << 2;
384             if (m > blen) m = blen;
385             for (k = buf[--m]&0xff; (m & 0x3) != 0; m += 4)
386                 ibuf[n] = k;
387         }
388         return ibuf;
389     }
390 }
```

SIMULATEUR PSEUDO ALÉATOIRE DU JEU DE LA VIE

Introduction

Ce code est une implémentation du célèbre Game of Life de John Conway.

Il utilise la bibliothèque SDL2 pour créer une interface graphique et afficher l'évolution des cellules en temps réel.

Le Jeu de la Vie est une simulation basée sur des règles simples permettant de simuler des phénomènes complexes.

Chaque cellule peut être dans l'un des deux états possibles (vivante ou morte), et son état futur dépend des états de ses voisines selon des règles préétablies. Le code utilise une taille de grille de 40x60 et une taille de cellule de 10 pixels.

La configuration initiale de la cellule est générée aléatoirement. Le joueur peut interagir avec la grille en cliquant sur les cellules, en les faisant passer de vivantes à mortes et vice versa, et peut également appuyer sur la barre d'espace pour faire évoluer la grille selon les règles du jeu.



```

1 #include <SDL2/SDL.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5
6 #define ROWS 40
7 #define COLS 60
8 #define CELL_SIZE 10
9
10 void drawGrid(SDL_Renderer *renderer, int grid[][COLS]) {
11     SDL_Rect rect;
12     rect.w = CELL_SIZE;
13     rect.h = CELL_SIZE;
14     for (int i = 0; i < ROWS; i++) {
15         for (int j = 0; j < COLS; j++) {
16             if (grid[i][j] == 1) {
17                 SDL_SetRenderDrawColor(renderer,
18                     255, 255, 255, 255);
19             } else {
20                 SDL_SetRenderDrawColor(renderer, 0
21                     , 0, 0, 255);
22             }
23             rect.x = j * CELL_SIZE;
24             rect.y = i * CELL_SIZE;
25             SDL_RenderFillRect(renderer, &rect);
26         }
27     }
28     int countNeighbors(int row, int col, int grid[][COLS]) {
29         int count = 0;
30         for (int i = row - 1; i <= row + 1; i++) {
31             for (int j = col - 1; j <= col + 1; j++) {
32                 if (i >= 0 && i < ROWS && j >= 0 && j
33 < COLS) {
34                     if (grid[i][j] == 1) {
35                         count++;
36                     }
37                 }

```

```

38     }
39     count -= grid[row][col];
40     return count;
41 }
42
43 void update(int grid[][COLS]) {
44     int newGrid[ROWS][COLS];
45     for (int i = 0; i < ROWS; i++) {
46         for (int j = 0; j < COLS; j++) {
47             int neighbors = countNeighbors(i, j,
grid);
48             if (grid[i][j] == 1) {
49                 if (neighbors < 2 || neighbors > 3
) {
50                     newGrid[i][j] = 0;
51                 } else {
52                     newGrid[i][j] = 1;
53                 }
54             } else {
55                 if (neighbors == 3) {
56                     newGrid[i][j] = 1;
57                 } else {
58                     newGrid[i][j] = 0;
59                 }
60             }
61         }
62     }
63     for (int i = 0; i < ROWS; i++) {
64         for (int j = 0; j < COLS; j++) {
65             grid[i][j] = newGrid[i][j];
66         }
67     }
68 }
69
70 void handleInput(SDL_Event *event, int grid[][COLS
]) {
71     int mouseX, mouseY;
72     switch (event->type) {
73         case SDL_QUIT:
74             exit(0);
75             break;
76         case SDL_MOUSEBUTTONDOWN:

```

```

77             mouseX = event->button.x / CELL_SIZE;
78             mouseY = event->button.y / CELL_SIZE;
79             grid[mouseY][mouseX] = !grid[mouseY][
80                 mouseX];
81         break;
82     case SDL_KEYDOWN:
83         if (event->key.keysym.sym ==
84             SDLK_SPACE) {
85             update(grid);
86         } else if (event->key.keysym.sym ==
87             SDLK_c) {
88             for (int i = 0; i < ROWS; i++) {
89                 for (int j = 0; j < COLS; j
90                     ++
91                 ) {
92                     grid[i][j] = 0;
93                 }
94             }
95         }
96     }
97 }
98
99     int main(int argc, char *argv[]) {
100         srand(time(NULL));
101         SDL_Init(SDL_INIT_VIDEO);
102
103         SDL_Window *window =
104             SDL_CreateWindow("Game of Life",
105
106             SDL_WINDOWPOS_UNDEFINED,
107             SDL_WINDOWPOS_UNDEFINED,
108             COLS * CELL_SIZE,
109             ROWS * CELL_SIZE,
110             SDL_WINDOW_SHOWN);
111
112         SDL_Renderer *renderer =
113             SDL_CreateRenderer(window, -1,
114                 SDL_RENDERER_ACCELERATED);
115
116

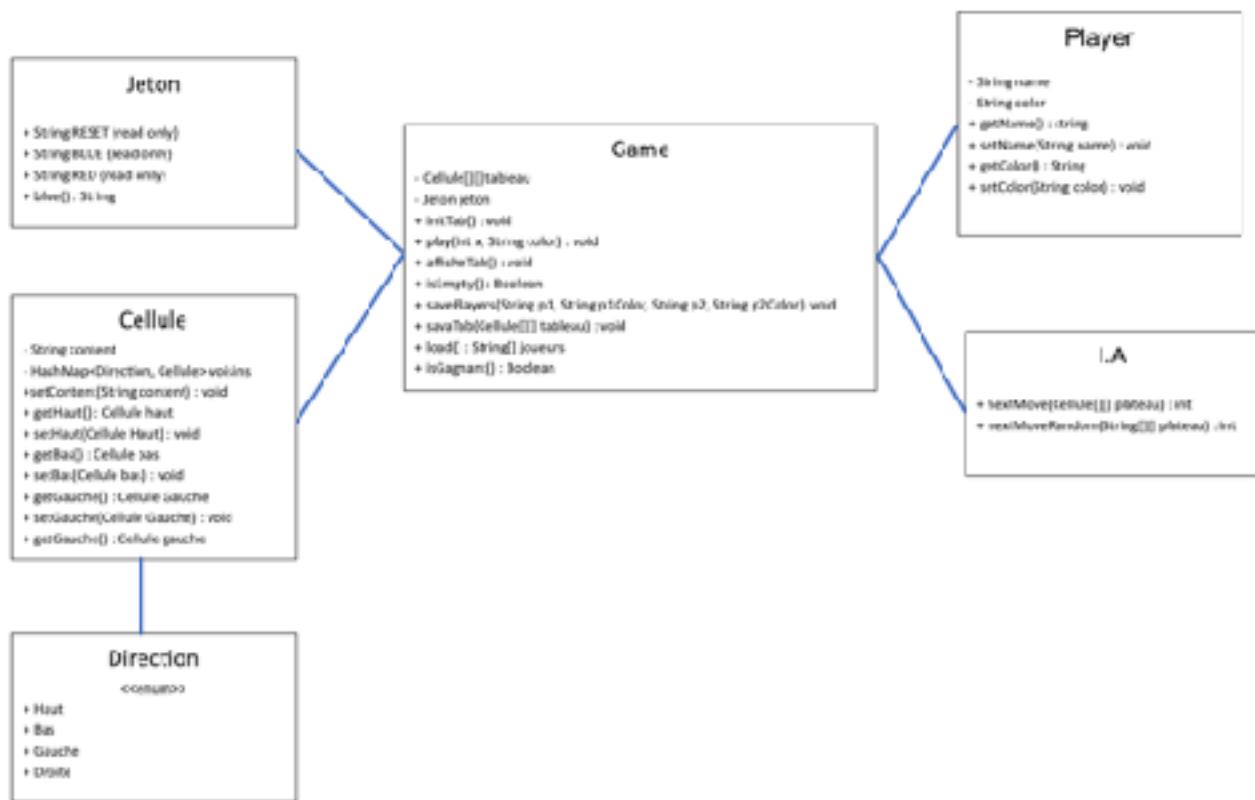
```

```
107             int grid[ROWS][COLS] = {0};  
108  
109 // Initial random configuration  
110         for (int i = 0; i < ROWS; i++) {  
111             for (int j = 0; j < COLS; j  
112                 ++ ) {  
113                 grid[i][j] = rand() % 2;  
114             }  
115         }  
116         while (1) {  
117             SDL_Event event;  
118             while (SDL_PollEvent(&event  
)) {  
119                 handleInput(&event, grid);  
120             }  
121  
122             SDL_SetRenderDrawColor(  
    renderer, 0, 0, 0, 255);  
123             SDL_RenderClear(renderer);  
124  
125             drawGrid(renderer, grid);  
126  
127             SDL_RenderPresent(renderer);  
128             SDL_Delay(10);  
129         }  
130  
131         SDL_DestroyRenderer(renderer);  
132         SDL_DestroyWindow(window);  
133         SDL_Quit();  
134  
135         return 0;  
136     }
```

JEU DE PUISSANCE 4

Ce programme en Java vous permet de jouer au célèbre jeu de stratégie où deux joueurs s'affrontent pour aligner quatre jetons de leur couleur horizontalement, verticalement ou en diagonale. Dans ce projet, vous pourrez jouer contre un autre joueur ou contre une intelligence artificielle qui utilisera des algorithmes pour essayer de vous battre. Nous avons utilisé une matrice de cellules pour représenter le plateau de jeu et avons implémenté différentes fonctions pour gérer la logique du jeu. Nous avons également ajouté une fonctionnalité de sauvegarde et de chargement pour que vous puissiez continuer vos parties ultérieurement. Que vous soyez un joueur expérimenté ou un débutant, notre programme vous offrira une expérience de jeu agréable et stimulante. Alors, êtes-vous prêt à jouer ?

DIAGRAMME UML :



J'ai choisi d'utiliser une matrice de cellules qui fera office de plateau de jeu, chaque cellule de la grille sera initialisée avec le symbole \circ pour dire qu'elle est vide, et grâce à la HashMap, chaque cellule contiendra une référence vers les cellules qui l'entourent.

Chaque joueur a un nom et une couleur. Grâce au code ANSI, mes cellules seront soit bleues soit rouges, affichées par le symbole \bullet .

Afin de savoir si le plateau est encore vide, j'ai ajouté une fonction qui parcourt la première ligne de notre plateau à la recherche d'une cellule vide (cellule contenant le symbole \circ).

Pour savoir qui a gagné, j'ai choisi d'implémenter une fonction `isGagnant` qui parcourt notre plateau à la recherche d'une suite de quatre jetons de la même couleur.

Pour sauvegarder, j'ai choisi de mettre le tableau dans un fichier texte et les informations sur les joueurs dans un autre.

Pour recharger, je recharge ligne par ligne le fichier texte dans notre plateau et je mets les informations joueurs dans un tableau afin de les réattribuer aux joueurs par la suite.

Pour l'IA, j'ai voulu faire un algorithme MinMax, mais faute de temps, je me suis plus orienté vers une approche plus simple. En premier lieu, j'ai fait un algorithme qui joue aléatoirement, puis j'ai fait un algorithme qui essaye de monter le plus possible.

1	2	3	4	5	6	7
\circ	\circ	\circ	\circ	\circ	\circ	\circ
\circ	\circ	\circ	\bullet	\circ	\circ	\circ
\circ	\circ	\circ	\bullet	\bullet	\circ	\circ
\circ	\circ	\bullet	\bullet	\bullet	\circ	\circ
\circ	\circ	\bullet	\bullet	\bullet	\circ	\circ
\bullet	\bullet	\bullet	\bullet	\bullet	\circ	\circ
Tour du joueur 1 ►						

```
1 public class IA {  
2     /*  
3     ****  
4         nextMove() : permet à l'IA de jouer un tour  
5             Prend comme paramètres le plateau  
6             Retourne le choix de l'IA  
7     ****  
8     */  
9     public int nextMove(Cellule[][] plateau) {  
10         float min = 800;  
11         int col = 0;  
12         int lignes = 6;  
13         int colonnes = 7;  
14         float midx = (float)colonnes / 2;  
15         float midy = (float)lignes / 2;  
16         for (int i = 0; i < lignes; i++) {  
17             for (int j = 0; j < colonnes; j++) {  
18                 if (plateau[i][j].getContent().equals  
19                     float valx = Math.abs(i-midx);  
20                     float valy = Math.abs(j-midy);  
21                     float finalVal= valx+ valy;  
22                     if (finalVal < min) {  
23                         min = finalVal;  
24                         col = j;  
25                     }  
26                 }  
27             }  
28         }  
29         return col;  
30     }  
31 }  
32 }
```

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.io.PrintWriter;
4 import java.util.Scanner;
5
6 import static java.lang.Thread.sleep;
7 public class Game {
8     Cellule[][] tableau = new Cellule[6][7];
9     Jeton jeton = new Jeton();
10
11    /*
12     *****
13     InitTab() : Initialisation du tableau de jeu
14     Ne prend rien en parametre
15     Ne retourne rien
16     *****
17    */
18    public void initTab(){
19        //plateau.initPlateau();
20        for (int i = 0; i < 6;i++){
21            for (int j = 0; j < 7;j++) {
22                tableau[i][j] = new Cellule();
23            }
24        }
25        for (int i = 0; i<6;i++){
26            for (int j = 0; j<7;j++){
27                tableau[i][j].setContent("◎");
28                try {
29                    tableau[i][j].setHaut(tableau[i - 1][j]);
30                }catch(Exception ignored){}
31                try{
32                    assert tableau != null;
33                    assert tableau[i] != null;
34                    assert tableau[i][j] != null;
35                    tableau[i][j].setBas(tableau[i + 1][j]);
36                }catch(Exception ignored){}
37                try {
38                    tableau[i][j].setGauche(tableau[i][j - 1]);
39                }catch(Exception ignored){}
40                try{
41                    assert tableau != null;
```

```

42                     assert tableau[i] != null;
43                     assert tableau[i][j] != null;
44                     tableau[i][j].setDroite(tableau[i]
45                     }catch(Exception ignored){}
46                 }
47             }
48         }
49
50     /*
51      *****
52      Play() : Deroulement d'un tour
53          Prend comme parametres le choix du joueur
54          Ne retourne rien
55      *****
56     */
57     public void play(int x,String color) throws FileNotFoundException
58
59         if(x == -1){ //save
60             saveTab(tableau);
61         }
62         else{
63             for(int i = 5; i>=0;i--){
64                 if(tableau[i][x].getContent().equals(
65                     if(color.equals("Rouge")){
66                         tableau[i][x].setContent(jetons);
67                         break;
68                     }
69                     if(color.equals("Bleu")){
70                         tableau[i][x].setContent(jetons);
71                         break;
72                     }
73                 }
74             }
75         }
76     }
77
78     /*
79      *****
80      afficheTab() : Affichage du plateau apres chaque tour
81          Ne prend rien en parametre
82          Ne retourne rien

```

```
83     ****
84     */
85     public void affichTab(){
86         System.out.println("1 2 3 4 5 6 7");
87         for(int i=0; i<6; i++){
88             for(int j=0; j<7; j++){System.out.print(
89                 System.out.println();
90             }
91         }
92     /*
93     ****
94     IsEmpty() : Verifie s'il y a encore de la pl
95         Ne prend rien en parametre
96         Retourne un Boolean
97     ****
98     */
99     public boolean isEmpty() {
100         for (int i = 0; i < 7; i++) {
101             if (tableau[0][i].getContent().equals("@"))
102                 return true;
103             }
104         }
105         return false;
106     }
107 */
108 /*
109     ****
110     savePlayer() : Permet de sauvegarder les joe
111         Prend comme parametres les noms des joue
112         Ne retourne rien
113     ****
114     */
115     public void savePlayers(String p1, String p1Colo
116         File joueurs = new File("Save_J.txt");
117         PrintWriter pr = new PrintWriter(joueurs);
118         pr.println(p1);
119         pr.println(p1Color);
120         pr.println(p2);
121         pr.println(p2Color);
122         pr.close();
```

```

124     }
125
126     /*
127      *****
128      saveTabr() : Permet de sauvegarder le Plateau
129      Prend comme parametres le plateau
130      Ne retourne rien
131      *****
132     */
133     public void saveTab(Cellule[][] tableau) throws
134         File tab = new File("Save_T.txt");
135         PrintWriter pr = new PrintWriter(tab);
136         for (int i = 0; i < 6; i++) {
137             for (int j = 0; j < 7; j++) {
138                 pr.println(tableau[i][j].getContent());
139             }
140         }
141         pr.close();
142     }
143
144     /*
145      *****
146      load() : Permet de recharger une partie
147      Ne prend rien comme parametre
148      Retourne les joueurs
149      *****
150     */
151     public String[] load() throws FileNotFoundException
152         String[] joueurs = new String[4];
153         Scanner loaderTab = new Scanner(new File("Save_T.txt"));
154         Scanner loaderP = new Scanner(new File("Save_P.txt"));
155         for (int i = 0; i < 6; i++) {
156             for (int j = 0; j < 7; j++) {
157                 if(loaderTab.hasNext()) {
158                     tableau[i][j].setContent(loaderTab.next());
159                 }
160             else{
161                 System.out.println("Il n'y a pas");
162             }
163             if(loaderP.hasNext()){
164                 for(int k = 0; k < 4; k++){

```

```
165                     joueurs[k]= loaderP.next();
166                 }
167             }
168         }
169     }
170     return joueurs;
171 }
172 /*
173 *****
174     isGagnant() : Permet de savoir si un joueur
175         Ne prend pas de parametre
176         Retourne un boolean
177 *****
178 */
179 public boolean isGagnant() {
180     for(int i = 0; i <6;i++){
181         for(int j = 0; j <7;j++){
182             if(! tableau[i][j].getContent().equa
183                 if(tableau[i][j].getContent().eq
184                     && tableau[i][j+1].getCo
185                     && tableau[i][j+2].getCo
186                 ){
187                     return true;
188                 }
189             }
190         }
191     }
192 }
193 for(int i = 0; i <3;i++){
194     for(int j = 0; j <7;j++){
195         if(! tableau[i][j].getContent().equa
196             if(tableau[i][j].getContent().eq
197                 && tableau[i+1][j].getCo
198                 && tableau[i+2][j].getCo
199             ){
200                 return true;
201             }
202         }
203     }
204 }
205 for(int i = 0; i <3;i++){
```

```
206         for(int j = 0; j < 7;j++){
207             if(! tableau[i][j].getContent().equa
208                 if(tableau[i][j].equals(tableau[
209                     && tableau[i+1][j+1].get
210                     && tableau[i+2][j+2].get
211                 ){
212                     return true;
213                 }
214             }
215         }
216     }
217 }
218 for(int i = 0; i < 3;i++){
219     for(int j = 3; j < 7;j++){
220         if(! tableau[i][j].getContent().equa
221             if(tableau[i][j].equals(tableau[
222                 && tableau[i+1][j-1].get
223                 && tableau[i+2][j-2].get
224             ){
225                 return true;
226             }
227         }
228     }
229 }
230 return(false);
231 }
232
233 public void easterEgg() throws InterruptedException
234
235     String[][] easter = new String[6][7];
236     for (int i = 0; i < 6;i++){
237         for (int j = 0; j < 7;j++) {
238             easter[i][j]=jeton.rouge();
239         }
240     }
241     easter[1][1] = jeton.bleu();
242     easter[2][1] = jeton.bleu();
243     easter[1][2] = jeton.bleu();
244     easter[2][2] = jeton.bleu();
245     easter[1][4] = jeton.bleu();
246     easter[2][4] = jeton.bleu();
```

```
247         easter[1][5] = jeton.bleu();
248         easter[2][5] = jeton.bleu();
249         easter[4][1] = jeton.bleu();
250         easter[4][2] = jeton.bleu();
251         easter[4][3] = jeton.bleu();
252         easter[4][4] = jeton.bleu();
253         easter[4][5] = jeton.bleu();
254         System.out.println("1 2 3 4 5 6 7");
255         for(int i=0; i<6; i++){
256             for(int j=0; j<7; j++){System.out.print(
257                 System.out.println();
258             }
259             sleep(2000);
260         }
261     }
262 }
```

```
1 import java.io.FileNotFoundException;
2 import java.util.Scanner;
3 public class Main {
4     public static void main(String[] args) throws Fil
5         String load,adversaire; // toujours le p1 qui
6         String[] T;
7         int choix;
8         Scanner input = new Scanner(System.in);
9         Player p1 = new Player();
10        Player p2 = new Player();
11        IA Machine = new IA();
12        Game jeu = new Game();
13        System.out.println();
14        System.out.println("PUISANCE 4-2000");
15        System.out.println("*****");
16        jeu.initTab();
17        do {
18            System.out.print("Voulez vous recharger l
19            load = input.next();
20        } while (!(load.equals("Oui") || load.equals(
21
22        if (load.equals("Oui")) { // PAS DE GOTO LES
23            T = jeu.load();
24            p1.setName(T[0]);
25            p1.setColor(T[1]);
26            p2.setName(T[2]);
27            if (p2.getName().equals("Machine")){
28                adversaire = "Machine";
29            }
30            else{
31                adversaire = "Humain";
32            }
33
34            p2.setColor(T[3]);
35        } else {
36            System.out.println();
37            System.out.print("Player 1 ► ");
38            p1.setName(input.next());
39
40            do {
41                System.out.print("Color (Rouge/Bleu)
```

```

42                     p1.setColor(input.nextInt());
43     } while !(p1.getColor().equals("Rouge"))
44
45     do {
46         System.out.print("Voulez vous joueur
47             adversaire = input.nextInt();
48     } while !(adversaire.equals("Humain") ||
49
50     if (adversaire.equals("Humain")) {
51         System.out.print("Player 2 ►");
52         p2.setName(input.nextInt());
53     } else {
54         p2.setName("Machine");
55     }
56     if (p1.getColor().equals("Bleu")) {
57         p2.setColor("Rouge");
58     }
59     if (p1.getColor().equals("Rouge")) {
60         p2.setColor("Bleu");
61     }
62 }
63 System.out.println(p1.getName() + " joue " +
64 System.out.println(p2.getName() + " joue " +
65 System.out.println("Vous pouvez sauvegarder a
66 jeu.affichTab();
67 do {
68     do {
69         System.out.print("Tour du joueur 1 ►";
70         choix = input.nextInt();
71         if (choix == 42) { //EasterEgg
72             jeu.easterEgg();
73             jeu.affichTab();
74         }
75         if (choix == 0) {
76             jeu.savePlayers(p1.getName(), p1.
77         }
78     } while !(choix >= 1 && choix <= 7 || ch
79     jeu.play(choix - 1, p1.getColor());
80     jeu.affichTab();
81     if(jeu.isGagnant()){
82         System.out.println(p1.getName() + " g

```

```
83             break;
84         }
85         if (adversaire.equals("Humain")) {
86             do {
87                 System.out.print("Tour du joueur");
88                 choix = input.nextInt();
89                 if (choix == 42) { //EasterEgg
90                     jeu.easterEgg();
91                     jeu.affichTab();
92                 }
93                 if (choix == 0) {
94                     jeu.savePlayers(p1.getName())
95                 }
96                 } while (!(choix >= 1 && choix <= 7));
97                 jeu.play(choix - 1, p2.getColor());
98                 jeu.affichTab();
99                 if(jeu.isGagnant()){
100                     System.out.println(p2.getName())
101                     break;
102                 }
103             } else {
104                 jeu.play(Machine.nextMove(jeu.tableau));
105                 jeu.affichTab();
106                 if(jeu.isGagnant()){
107                     System.out.println("l'Ordinateur");
108                     break;
109                 }
110             }
111         }while (jeu.isEmpty());
112         System.out.println("Fin Partie");
113     }
114 }
```

```
1 public class Jeton {  
2  
3     public static final String RESET = "\u001B[0m";  
4     public static final String BLUE = "\u001B[36m";  
5     public static final String RED = "\u001B[0;31m";  
6  
7     public String bleu(){  
8         return(BLUE+"@"+RESET);  
9     }  
10    public String rouge(){  
11        return(RED+"@"+RESET);  
12    }  
13 }  
14
```

```
1 public class Player {  
2     String name;  
3     String color;  
4  
5     public String getName() {  
6         return name;  
7     }  
8     public void setName(String name) {  
9         this.name = name;  
10    }  
11    public String getColor() {  
12        return color;  
13    }  
14    public void setColor(String color) {  
15        this.color = color;  
16    }  
17}  
18
```

```
1 import java.util.HashMap;
2
3 public class Cellule {
4     String content;
5     HashMap<Direction, Cellule> voisins = new HashMap<>;
6     public String getContent() {
7         return content;
8     }
9     public void setContent(String content) {
10        this.content = content;
11    }
12    public void setHaut(Cellule haut) {
13        voisins.put(Direction.Haut, haut);
14    }
15    public void setBas(Cellule bas) {
16        voisins.put(Direction.Bas, bas);
17    }
18    public void setGauche(Cellule gauche) {
19        voisins.put(Direction.Gauche, gauche);
20    }
21    public void setDroite(Cellule droite) {
22        voisins.put(Direction.Droite, droite);
23    }
24 }
25
26 enum Direction {
27     Haut,
28     Bas,
29     Gauche,
30     Droite
31 }
32
33
34
```

SITE WEB

Dans le cadre de mon projet de création d'un site web dynamique, j'ai réalisé une revue scientifique en utilisant les langages HTML et CSS. Ce site web comprend cinq rubriques : santé, espace, culture, nature et technologie. L'objectif de ce projet est de fournir des informations pertinentes et précises dans chacune de ces rubriques, tout en offrant une expérience utilisateur agréable et interactive.

Lors de la réalisation de ce projet, j'ai dû mettre en pratique mes connaissances en HTML et CSS pour concevoir une interface utilisateur intuitive et esthétique. J'ai également travaillé sur l'optimisation du code pour améliorer les performances du site web.

La rubrique santé contient des articles sur les dernières avancées dans le domaine de la médecine et de la santé, tandis que la rubrique espace présente des informations sur l'exploration spatiale et les découvertes scientifiques récentes. La rubrique culture explore les relations entre la science et la culture, tandis que la rubrique nature se concentre sur les découvertes scientifiques en lien avec la biodiversité et l'environnement. Enfin, la rubrique technologie aborde les avancées technologiques les plus récentes.

Dans l'ensemble, ce projet m'a permis de mettre en pratique mes compétences en HTML et CSS tout en me permettant de découvrir de nouveaux domaines scientifiques.

Vous trouverez le code source du site sur mon compte Github:

<https://github.com/D4L1IR3Tr0/D4L1IR3Tr0.github.io>

Ainsi que le site en question sur ce lien:

<https://dalidamergi.me/main>

Distributeur Automatique de nourriture pour chat

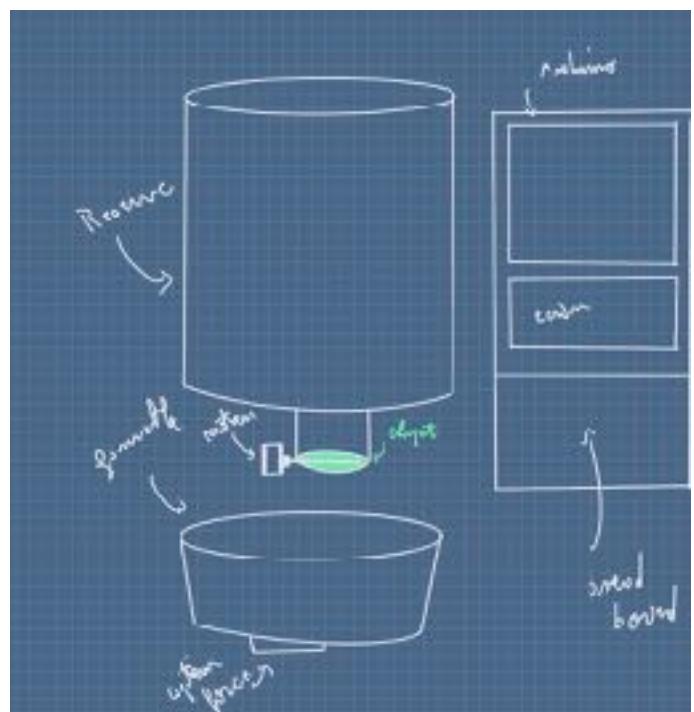
La robotique est un domaine que j'ai toujours trouvé fascinant, aujourd'hui cette même fascination s'est transformée petit à petit en violon d'Ingres. Ainsi dans le cadre d'un projet personnel j'ai décidé de créer un distributeur automatique de nourriture pour mon chat.

L'idée est que le distributeur sert deux portions de 30g par jour, une le matin et l'autre le soir. Le dit distributeur doit aussi mesurer la quantité de nourriture déjà présente dans la gamelle. Ainsi si la quantité de nourriture déjà présente dans la gamelle est inférieure à 30g à 8h et à 22h une trappe s'ouvre à l'aide d'un servo moteur afin de permettre la distribution. Cette trappe reste ouverte pendant 5 secondes.

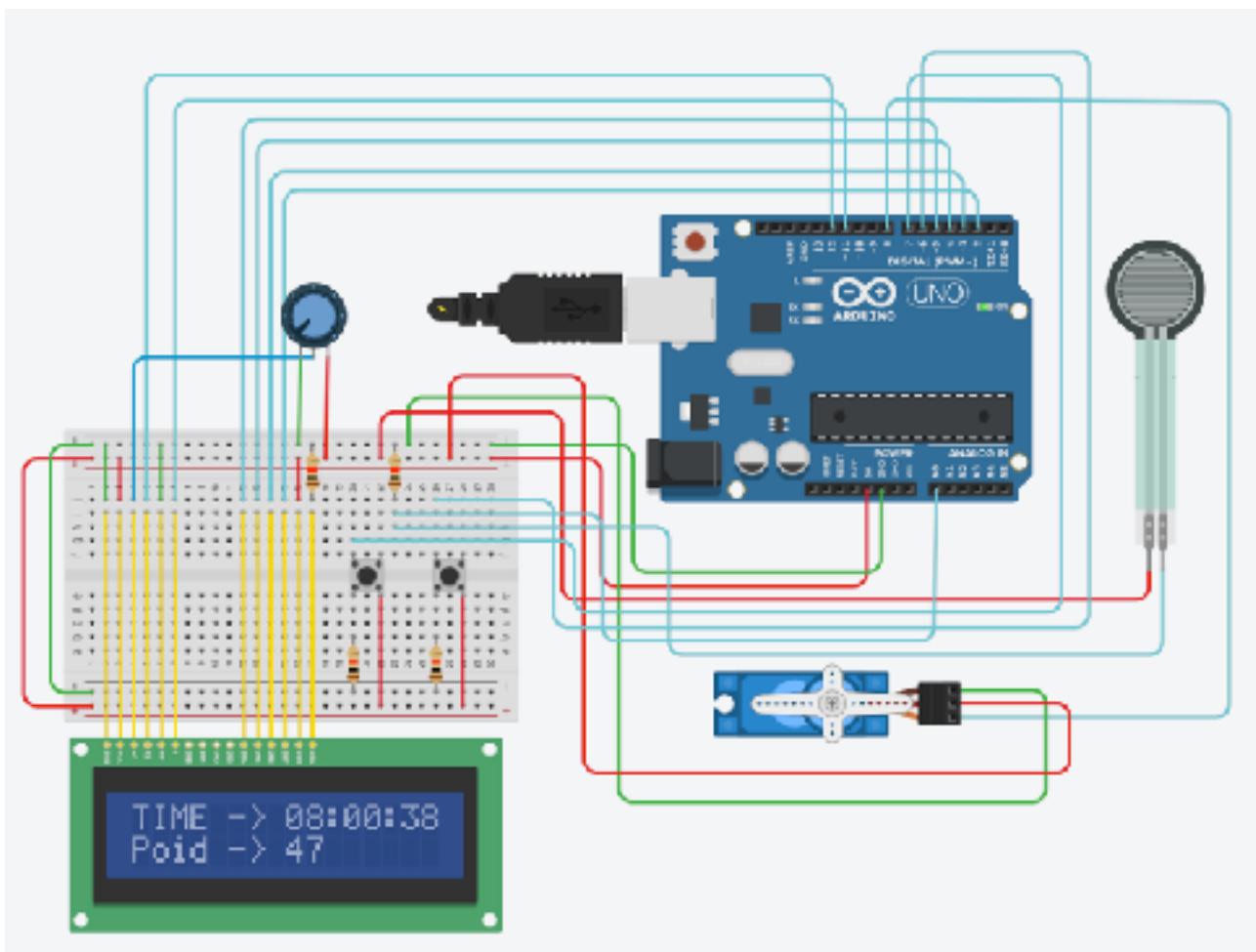
Le dispositif comprend un servo moteur, un écran LCD 16x2, deux boutons poussoir, un potentiomètre et a été réalisé grâce à une carte Arduino.

Afin de vérifier le prototype une BreadBoard (Platine d'essais) a été utilisée mais pour la version finale l'idéal serait de créer une carte électronique avec les composants soudés dessus.

En guise de réservoir une bonbonne d'eau de 5 lettres a été utilisée



Montage



L'afficheur LCD permet d'afficher la quantité de nourriture déjà présente dans la gamelle ainsi que l'heure.

Le temps étant géré par une simple incrémentation des secondes chaque 1000 millisecondes j'ai rajouté deux boutons poussoirs servant à changer les heures et les minutes.

Le potentiomètre sert à augmenter ou diminuer la luminosité de l'écran.

Code en C++

```
1
2 #include <LiquidCrystal.h>
3 #include <Servo.h>
4
5 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6
7 int Seconde = 0;
8 int Minute = 0;
9 int Heure = 0;
10 int Button1 = 0;
11 int Button2 = 0;
12
13
14 Servo servo_8;
15
16 void setup() {
17
18     pinMode(A0, INPUT);
19     pinMode(6, INPUT);
20     pinMode(13, INPUT);
21     servo_8.attach(8, 500, 2500);
22     pinMode(LED_BUILTIN, OUTPUT);
23
24
25     Seconde = 55;
26     Minute = 59;
27     Heure = 7;
28     lcd.begin(16, 2);
29     analogWrite(6, 128);
30
31
32 }
33
34 void loop() {
35
36
37
38     char timeStr[16];
39     char poidStr[3];
40     char Aff[16];
41
42     sprintf(timeStr, "TIME -> %02d:%02d:%02d", Heure
```

```
42 , Minute, Seconde);
43   sprintf(Aff, "Poid -> %02d ", analogRead(A0));
44
45
46 if(digitalRead(7) == HIGH) {
47   Heure += 1;
48 }
49 if(digitalRead(6) == HIGH) {
50   Minute += 1;
51 }
52
53
54
55 delay(1000);
56 if (Seconde >= 59) {
57   Seconde = 0;
58   Minute++;
59   if (Minute >= 59) {
60     Minute = 0;
61     Heure++;
62     if((Heure == 24)) Heure = 0;
63   }
64 }
65 else {
66   Seconde++;
67 }
68
69
70 lcd.setCursor(0, 1);
71 lcd.print(Aff);
72 lcd.setCursor(0, 0);
73 lcd.print(timeStr);
74
75 if (((Heure == 8 && Minute == 0 && Seconde < 05
76 ) || (Heure == 22 && Minute == 0 && Seconde < 05
77 )) && analogRead(A0)> 250) {
78   servo_8.write(50);
79 } else {
80   servo_8.write(0);
81 }
```

Véhicule a deux roues motrices

Ce projet consiste a créer un vehicule télécommandé a l'aide d'une carte programmable Arduino.

Le véhicule est équipé de deux roues motrice ainsi que d'une roulette pivotante.

Le véhicule est aussi équipé d'un capteur a ultrasons afin de mesurer la distance entre le véhicule et les objets en face.

Le véhicule sera controlé par une télécommande doté de quatre boutons pour avancer, reculer et tourner.

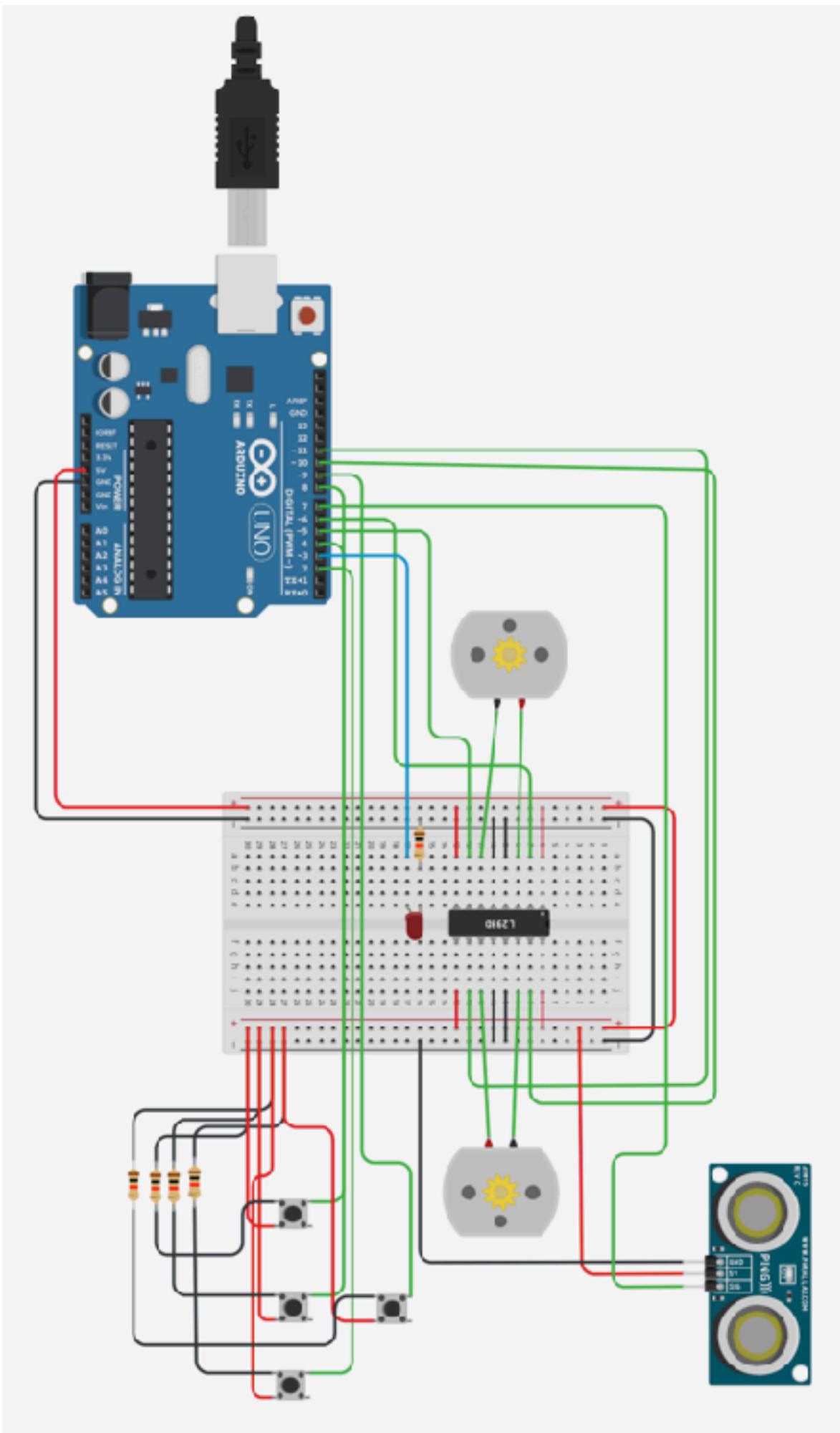
Le véhicule peut bien évidemment être aussi programmé par avance afin d'effectuer des mouvement sans être controlé par un humain.

Si le véhicule est a une distance inférieur a 30cm il s'arrête et ne peu plus avancer, un voyant rouge s'allume alors afin de signaler cela a l'utilisateur.

Sinon le véhicule peut avancer, reculer et tourner sur lui même.

Le montage comprend deux moteur, une puce L293D servant d'entraineur de moteur, un capteur de distance par ultrasons, 4 boutons ainsi qu'une diode LED servant de voyant lumineux.

Si la vitesse des moteur est trop importante il faudrait penser a rajouter un potentiomètre afin de la contrôler.



Code en C++

```
1 void setup() {
2
3     Serial.begin(9600);
4
5     pinMode(5,OUTPUT);
6     pinMode(6,OUTPUT);
7     pinMode(11,OUTPUT);
8     pinMode(10,OUTPUT);
9     pinMode(7,INPUT);
10    pinMode(4, INPUT);
11    pinMode(9, INPUT);
12    pinMode(2, INPUT);
13    pinMode(8, INPUT);
14
15 }
16 }
17 long duration, cm;
18
19 void stop()
20 {
21     digitalWrite(5,LOW);
22     digitalWrite(6,LOW);
23     digitalWrite(11,LOW);
24     digitalWrite(10,LOW);
25 }
26
27 void ar()
28 {
29     digitalWrite(5,HIGH);
30     digitalWrite(6,LOW);
31     digitalWrite(11,HIGH);
32     digitalWrite(10,LOW);
33 }
34 void av()
35 {
36     cm = microsecondsToCm(duration);
37     if(cm > 30){
38         digitalWrite(5,LOW);
39         digitalWrite(6,HIGH);
40         digitalWrite(11,LOW);
41         digitalWrite(10,HIGH);
42 }
```

```

43 }else{
44     digitalWrite(3,HIGH);
45     stop();
46 }
47 }
48 void g()
49 {
50     digitalWrite(5,LOW);
51     digitalWrite(6,HIGH);
52     digitalWrite(11,HIGH);
53     digitalWrite(10,LOW);
54 }
55 void d()
56 {
57     digitalWrite(5,HIGH);
58     digitalWrite(6,LOW);
59     digitalWrite(11,LOW);
60     digitalWrite(10,HIGH);
61 }
62
63 long microsecondsToCm(long microseconds)
64 {
65     return microseconds / 29 / 2;
66 }
67
68 void loop()
69 {
70     digitalWrite(3,LOW);
71     pinMode(7, OUTPUT);
72     digitalWrite(7, LOW);
73     delayMicroseconds(2);
74     digitalWrite(7, HIGH);
75     delayMicroseconds(5);
76     digitalWrite(7, LOW);
77     pinMode(7, INPUT);
78     duration = pulseIn(7, HIGH);
79
80     if(digitalRead(2)==HIGH){
81         Serial.print("D");
82         d();
83     }
84     else if(digitalRead(9)==HIGH){

```

```
85     Serial.print("AV");
86     av();
87 }
88 else if(digitalRead(4)==HIGH){
89     Serial.print("G");
90     g();
91 }
92 else if(digitalRead(8)==HIGH){
93     Serial.print("AR");
94     ar();
95 }
96 else{
97     stop();
98 }
99
100 }
```

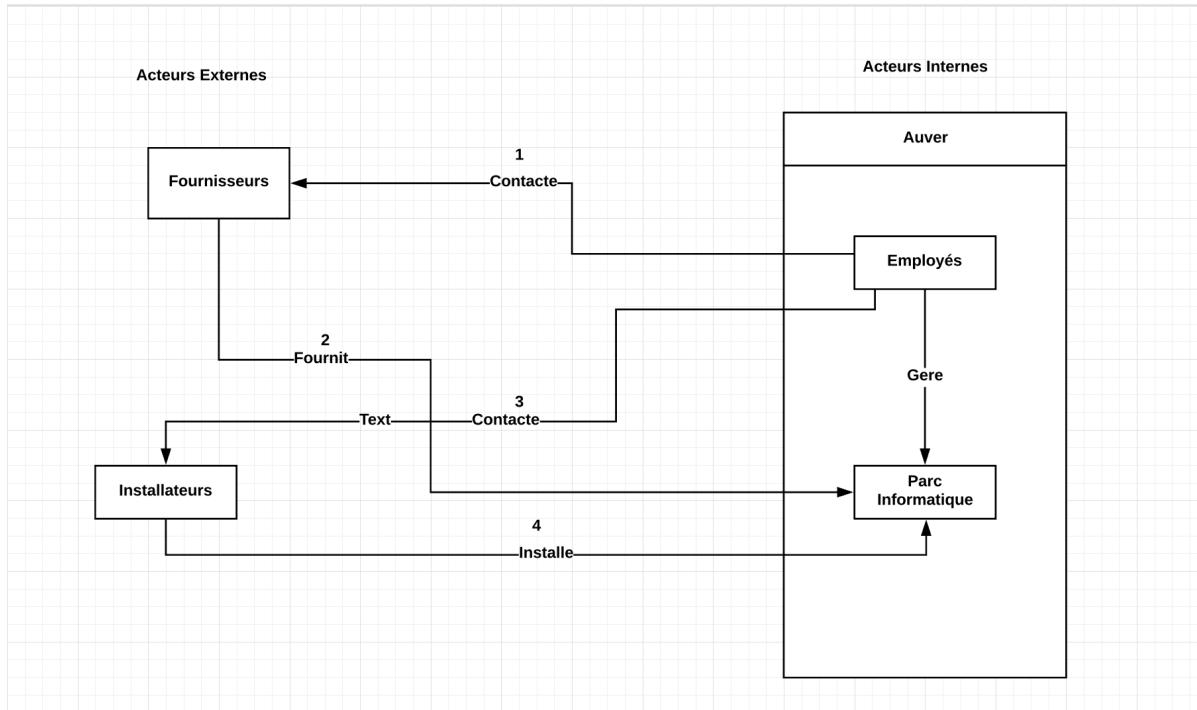
Etude de la base de donnée de la société fictive AUVER

Auver est une entreprise basée en Auvergne qui gère plusieurs parcs informatique et qui aujourd’hui a fait appel à une entreprise extérieure afin de venir leur installer les dernières versions des logiciels, logiciels qui ont préalablement été commandés à une autre entreprise externe spécialiste dans ce domaine.

On se propose donc de faire l’étude de la base de donnée de cette transaction entre Auver et les deux autres entreprises extérieures.

Tp-SI

I) Diagramme de Flux :



II) Dictionnaire de données :

Code Rubrique	Libelé	Type	Nature	Calcul	Intégrité
Num-Emp	Numero de l'employer	N	E		
Nom-Emp	Nom de l'employer	A	E		
Prenom-Em	Prenom de l'employer	A	E		
Grade-Emp	Grade de l'employer	N	E		
Dip-Emp	Diplôme de l'employer	A	E		
Dte-Emb-Emp	Date d'embauche de l'employer	D	E		xx xx xxxx
Serv-Emp	Service de l'employer	A	E		

Code Rubrique	Libelé	Type	Nature	Calcul	Intégrité
Form-Emp	Formation suivie par l'employeur	A	E		
Tps-Util-Log-Emp	Temps d'utilisation d'un logiciel par employeur	D	E		xx xx xx
Nom-Serv	Nom du service	A	E		
Num-Ordi	Numero de l'ordinateur	N	E		
Nom-Ordi	Nom de l'ordinateur	N	E		
Disq-Ordi	Taille du disque de l'ordinateur	N	E		
Mem-Ordi	Taille de la mémoire de l'ordinateur	N	E		
Dte-Acha-Ordi	Date d'achat de l'ordinateur	D	E		xx xx xxxx
Serv-Ordi	Service de l'ordinateur	A	E		
Nom-Log	Nom du logiciel	A	E		
Type-Log	Type du logiciel	A	E		
Edit-Log	Editeur du logiciel	A	E		
Num-Serie	Numero de serie du logiciel	N	E		
Dte-Instl-Log	Date d'installation du logiciel	D	E		xx xx xxxx
Tps-Util-Log-Emp-Glo	Temps Global d'utilisation d'un logiciel par un employeur	N	E		xx xx xx
Cod-Instlr	Code de l'installateur	AN	E		
Nom-Instlr	Nom de l'installateur	A	E		

Code Rubrique	Libelé	Type	Nature	Calcul	Intégrité
Sct-Instlr	Société de l'installateur	A	E		
Fct-Instlr	Fonction de l'installateur	A	E		
Nom-Fourn	Nom du fournisseur	A	E		
Adr-Fourn	Adresse du fournisseur	AN	E		
Remise-AI	Remise les as de l'info	N	C	10% Si CA >1000 20% Si CA > 2000 30% Si CA > 3500	
Remise-IP	Remise Infoparc	N	C	15% Si CA > 2000 25% Si CA > 3000	
Remise-IF	Remise infofutur	N	C		
Taux-Remise	Taux de remise du fournisseur	N	C	En fonction de l'entreprise et du nombre d'ordi	
Cfre-Affr	Chiffre d'affaire	N	E		

III) Dépendances fonctionnelles élémentaires directes :

Num-Emp → Nom-Emp

Num-Emp → Prénom-Emp

Num-Emp → Dte-Emb-Emp

Num-Emp → Grade-Emp

Num-Emp → DipEmp

Num-Emp → Form-Emp

Num-Emp → Serv-Emp

Num-Emp → Tps-Util-Log-Emp

Num-Ordi → Nom-Ordi

Num-Ordi → Mem-Ordi

Num-Ordi → Disq-Ordi

Num-Ordi → Dte-Acha-Ordi

Num-Ordi → Serv-Ordi

Num-Serie → Nom-Log

Num-Serie → Type-Log

Num-Serie → Edit-Log

Num-Serie → Dte-Instl-Log

Cod-Instlir → Nom-Instlir

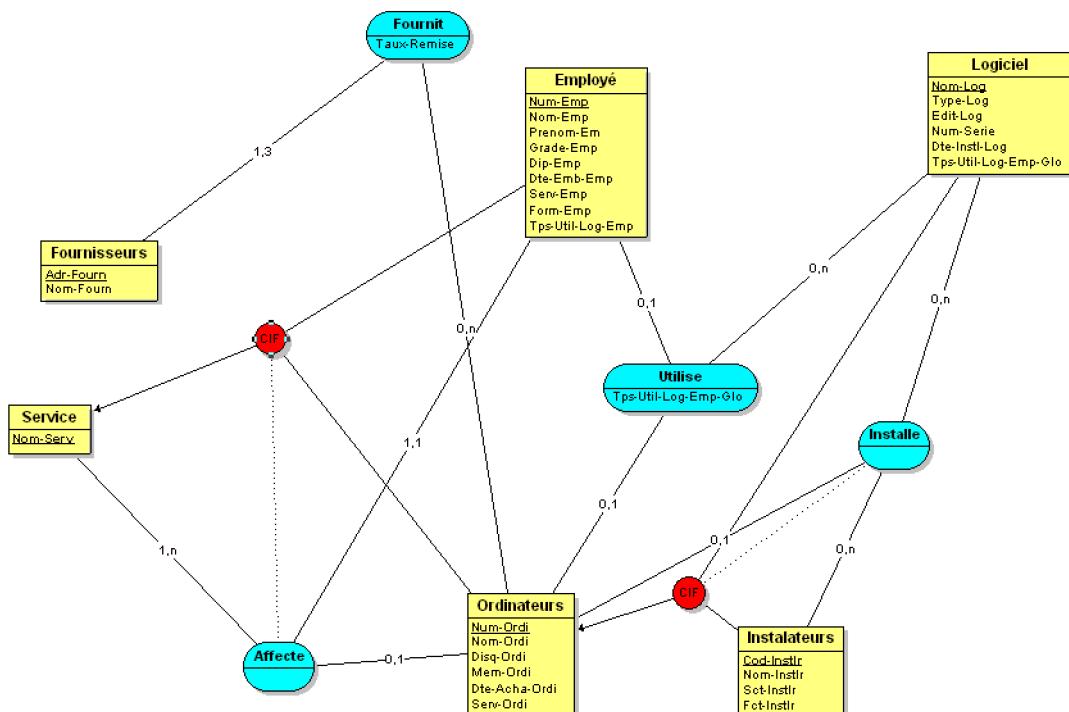
Cod-Instlir → Sct-Instlir

Cod-Instlir → Fct-Instlir

Adr-Fourn → Nom-Fourn

Adr-Fourn → Cfre-Affr

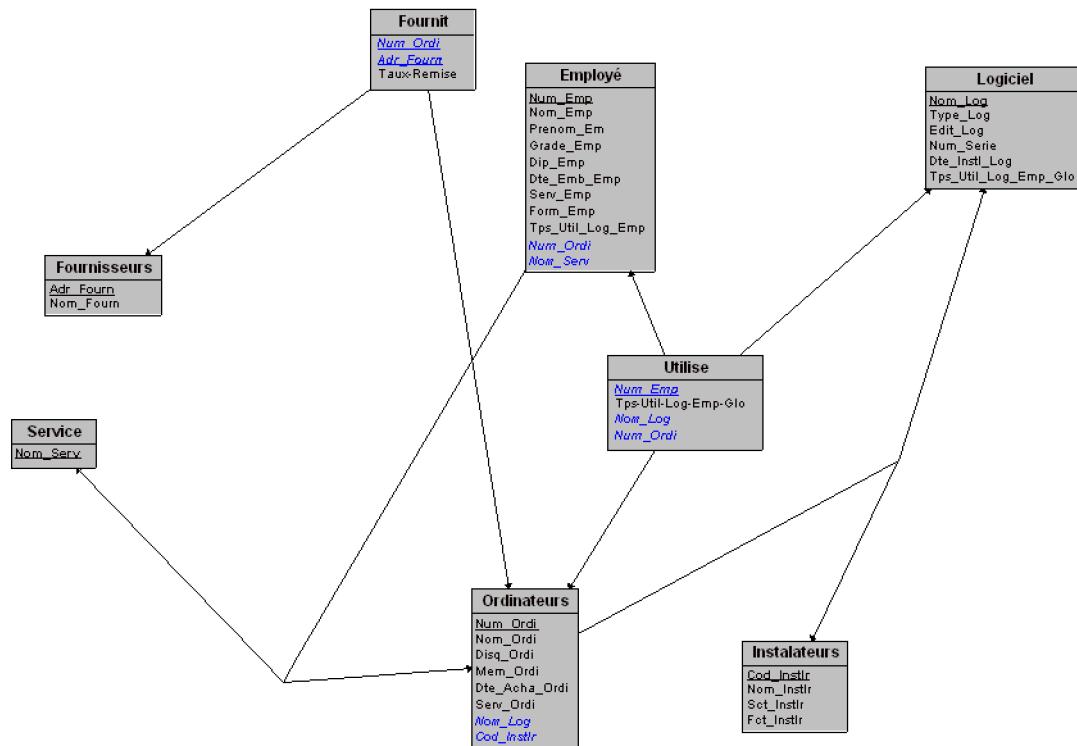
IV) Construction MCD simple :



V) Construction MLD textuel :

Logiciel = (Nom_Log VARCHAR(50), Type_Log VARCHAR(50), Edit_Log VARCHAR(50), Num_Serie INT, Dte_Instl_Log DATE, Tps_Util_Log_Emp_Glo TIME);
Instalateurs = (Cod_Instlr VARCHAR(50), Nom_Instlr VARCHAR(50), Sct_Instlr VARCHAR(50), Fct_Instlr VARCHAR(50));
Ordinateurs = (Num_Ordi INT, Nom_Ordi VARCHAR(50), Disq_Ordi INT, Mem_Ordi INT, Dte_Acha_Ordi DATE, Serv_Ordi VARCHAR(50), #Nom_Log*, #Cod_Instlr*);
Service = (Nom_Serv VARCHAR(50));
Fournisseurs = (Adr_Fourn VARCHAR(50), Nom_Fourn VARCHAR(50));
Employé = (Num_Emp INT, Nom_Emp VARCHAR(50), Prenom_Em VARCHAR(50), Grade_Emp VARCHAR(50), Dip_Emp VARCHAR(50), Dte_Emb_Emp DATE, Serv_Emp VARCHAR(50), Form_Emp VARCHAR(50), Tps_Util_Log_Emp TIME, #Num_Ordi, #Nom_Serv);
Utilise = (#Num_Emp, Tps_Util_Log_Emp_Glo TIME, #Nom_Log, #Num_Ordi);
Fournit = (#Num_Ordi, #Adr_Fourn, Taux_Remise INT);

MLD :

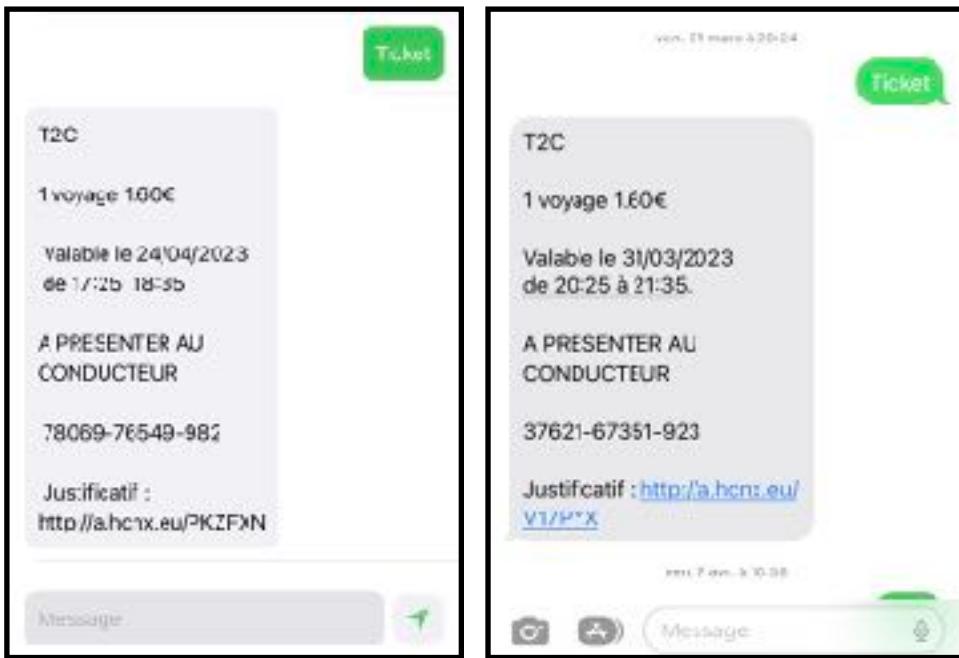


Application IOS (transport de Clermont)

Un ami à moi se faisait toujours contrôler par les contrôleurs de la ville de Clermont Ferrand et m'avais demandé de lui trouver une solution.

Trouvant l'idée amusante je me suis pris au jeu et j'ai donc codé en swiftUI une application IOS qui génère des ticket de tram reçus par SMS.

La fraude étant illégale l'application n'a été utilisée qu'une seule et unique fois en guise de test (qui a d'ailleurs fonctionné) mais jamais depuis.



Ma version

Version originale

```

import SwiftUI

struct ContentView: View {
    @Environment(\.colorScheme) var colorScheme

    struct Message: Codable {
        let text: String
        let isResponse: Bool
    }

    @State private var messageText = ""
    @State private var messages: [Message] = []
    @State private var responseText = ""

    let heureFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "HH:mm"
        return formatter
    }()

    let date_formatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateStyle = .short
        return formatter
    }()

    func randomString(length: Int) -> String {
        let letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
        return String((0..

```

```

NavigationView {
    VStack {
        List(messages.reversed(), id: \.text) { message in
            HStack {
                if message.isResponse {
                    Spacer()
                    Text(message.text)
                        .foregroundColor(.white)
                        .padding(10)
                        .background(Color.green)
                        .cornerRadius(10)
                } else {
                    Text(message.text)

                        .foregroundColor(colorScheme == .dark ?
                            .white : .black)
                        .padding(10)
                        .background(Color(.systemGray6))
                        .cornerRadius(10)
                }
            }
        }
    }
    .listStyle(PlainListStyle())
    .navigationBarTitle("Ticket T2C")
    .onAppear {
        UITableView.appearance().separatorStyle = .none

        // Récupérer les messages enregistrés dans UserDefaults
        let defaults = UserDefaults.standard
        if let messagesData = defaults.data(forKey: "messages"),
           let decodedMessages = try?
               JSONDecoder().decode([Message].self, from:
                   messagesData) {
            messages = decodedMessages
        }
    }
    .onTapGesture {
        // Masquer le clavier lorsque l'utilisateur touche
        // l'écran
        UIApplication.shared.sendAction(#selector(UIResponder
            .resignFirstResponder), to: nil, from: nil, for: nil)
    }

    HStack {
        TextField("Message", text: $messageText)
            .padding(10)
            .background(Color(.systemGray5))
    }
}

```

```

        .cornerRadius(10)

    Button(action: {
        if messageText.lowercased() == "clear" {
            // Effacer tous les
            messages
            messages.removeAll()

        } else if messageText.lowercased() == "ticket" {
            // Générer la réponse
        responseText = ("T2C\n\n1 voyage 1.60€ \n\n Valable le
        \n(date_actuelle_str) \n de \n(heure_actuelle_str)
        \n(heure_plus_une_str)\n\nA PRESENTER AU \nCONDUCTEUR \n\n \n(random_number)
        \n\n Justificatif :\n") + link
            messages.insert(Message(text: messageText,
                isResponse: true), at: 0)
            messages.insert(Message(text: responseText,
                isResponse: false), at: 0)

        }else if messageText.lowercased() == "ticket " {
            // Générer la réponse
            responseText = "T2C\n\n1 voyage 1.60€ \n\n Valable le \n(date_actuelle_str) \n de
            \n(heure_moins_str) à
            \n(heure_plus_une1_str)\n\nA PRESENTER AU
            \nCONDUCTEUR \n\n \n(random_number) \n\n Justificatif :\n" + link
            messages.insert(Message(text: messageText,
                isResponse: true), at: 0)
            messages.insert(Message(text: responseText,
                isResponse: false), at: 0)

        }
        else {
            messages.insert(Message(text: messageText,
                isResponse: true), at: 0)
        }

        // Sauvegarder les messages dans UserDefaults
        let defaults = UserDefaults.standard
        let messagesData = try?
            JSONEncoder().encode(messages)
        defaults.set(messagesData, forKey: "messages")

        messageText = ""
    })
{
    Image(systemName: "paperplane.fill")
        .foregroundColor(.green)
        .padding(10)
        .background(Color(.systemGray6))
        .cornerRadius(10)
}
.padding()

```

```
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Ce portfolio contient certains des projets sur lesquels j'ai eu à travailler, qu'ils soient personnels ou dans le cadre de mes études.

J'ai essayé de mettre quelques projets de chaque type de discipline ainsi il y a de l'étude de base de donnée, des projets de simulation pseudo aléatoire, de la cryptographie, de la création de jeu en java, de la conception de site web en HTML et CSS, de la programmation de systèmes en C++...

D'autres projets sont présent sur ma page Github:

<https://github.com/D4L1IR3Tr0>

**Ce PORTFOLIO est
maintenant
disponible en
siteWeb**

<https://d4l1ir3tr0.github.io/D4L1IR3Tr0-pf.github.io/Index>