

Itération 3 groupe 1

Exercice 7.28.1 :

création de la commande **test** acceptant un second mot qui représente un nom de fichier, et exécute toutes les commandes lues dans ce fichier texte, les classes à modifier sont « CommandWord » et « GameEngine » :

```
230     private void test(Command command){
231         if (!command.hasSecondWord()) {
232             // if there is no second word, we don't know where to go...
233             gui.println("Whate is the name of the file ?");
234             return;
235         }
236         try
237         {
238             String nameFile = "test.txt";
239             nameFile = command.getSecondWord();
240             Scanner sc = new Scanner(new File(nameFile));
241             while (sc.hasNext()) {
242                 String line = sc.nextLine();
243                 //gui.println(line);
244                 interpretCommand(line);
245             }
246             sc.close();
247         }
248         catch (FileNotFoundException ex)
249         {
250             gui.println("File not fond.");
251             ex.printStackTrace();
252         }
253     }
```

Apprentissage lecture de fichiers de text, hasNext(), next(), exceptions : Fait.

<https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>

<https://docs.oracle.com/javase/8/docs/api/java/io/File.html>

Exercice 7.28.2:

exercice fait par rapport au scénario développé dans ce jeu.

Exercice 7.29:

L'ajout de la classe **Player** qui contient un nom, le poids maximum qu'il supporte, le poids qu'il porte actuellement, un gain et les objets que le joueur a ramassé ou mangé.

```
1  public class Player {
2      private String name;
3      private int weightMax;
4      private int weight;
5      private int gain;
6      private ItemList items;
7
8      public Player(String name) {
9          this.name = name;
10         this.weightMax = 60;
11         this.weight = 0;
12         this.gain = 0;
13         this.items = new ItemList();
14     }
```

Exercice 7.30:

L'ajout de deux commandes take et drop qui s'occupe de la prise ou du dépôt d'un objet déjà prit respectivement, les classes modifiées sont « CommandWord » et « GameEngine » :

```
255 private void take(Command command) {
256     if (!command.hasSecondWord()) {
257         gui.println("vous voulez manger quoi?");
258         return;
259     }
260
261     String ob = command.getSecondWord();
262
263     // Try to leave current room.
264     Objet objet = currentRoom.getObjet(ob);
265
266     if (objet == null) {
267         gui.println("tu vas rien manger!");
268     } else if (currentPlayer.addObjets(objet)) {
269         currentRoom.deleteObjetRoom(ob);
270         gui.println("take :");
271         gui.println(objet.toString());
272     } else {
273         gui.println("weight Exceed! choose an other object.");
274     }
275 }
```

```
278 private void drop(Command command) {
279     if (!command.hasSecondWord()) {
280         gui.println("vous voulez manger quoi?");
281         return;
282     }
283
284     String ob = command.getSecondWord();
285
286     // Try to leave current room.
287     Objet objet = currentPlayer.getObjet(ob);
288
289     if (objet == null) {
290         gui.println("tu vas rien manger!");
291     } else {
292         currentRoom.addObjetsRoom(objet);
293         currentPlayer.deleteObjetPlayer(objet);
294         gui.println("drop :");
295         gui.println(objet.toString());
296     }
297 }
```

Exercice 7.31 & exercice 7.31.1::

porter items fait.

Création de la classe ItemList qui gère une liste d'items et mutualiser la gestion des items qui ont été dupliqué dans Room et dans Player.

```
1 import java.util.HashMap;
2 import java.util.Map;
3 import java.util.Set;
4
5 public class ItemList{
6     private HashMap<String, Objet> objets;
7     private int nombreObjets;
8     public ItemList(){
9         this.objets = new HashMap<String, Objet>();
10        this.nombreObjets = 0;
11    }
12
13    public Objet getObjet(String ob) {
14        return objets.get(ob);
15    }
16
17    public void createObjets()
18    {
19        int min = 2;
20        int max = 5;
21        nombreObjets = min + (int)(Math.random() * ((max - min) + 1)); // entre 2 à 5
22        for(int i = 0; i < nombreObjets ; i++){
23            Objet obj = new Objet();
24            objets.put(""+i, obj);
25        }
26    }
27
28    public String lookObjets(){
29        String strObjets = "Nombre de objets: " + nombreObjets + "\n";
30        for(Map.Entry<String, Objet> entry : objets.entrySet()){
31            String key = entry.getKey();
32            Objet obj = entry.getValue();
33            strObjets += key;
34            strObjets += obj;
35        }
36        return strObjets;
37    }
38
39    public void deleteObjet(String ob){
40        objets.remove(ob);
41        --nombreObjets;
42    }
43
44    public void addObjet(Objet obj){
45        objets.put(""+nombreObjets,obj);
46        nombreObjets++;
47    }
48
49    public boolean hasCles(){
50        int i = 0;
51        for(Map.Entry<String, Objet> entry : objets.entrySet()){
52            Objet obj = entry.getValue();
53            if(obj.getDescription().equals("Clé de la chambre à trésor"))
54                i++;
55        }
56        if (i == 5)
57            return true;
58        return false;
59    }
60    public boolean magic(Objet ob){
61        if(ob.getDescription().equals("Un cookie"))
62            return true;
63        return false;
64    }
65 }
```

Exercice 7.32 & exercice 7.33 :

Poid max et inventaire c'est fait dans dans la classe Player et GameEngine avec une nouvelle commande *items*.

```
299     private void items() {
300         gui.println(currentPlayer.lookObjetsPlayer());
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
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Et cette dernière appelle la méthode lookObjets dans la classe ItemsList.

Exercice 7.34:

l'ajout d'un objet magic cookie qui augmente le poid maximum d'un joueur, qui appelle la méthode magic de la classe ItemsList.

```
16     public boolean addObjets(Objet ob){
17         if(!weightExceed(ob)){
18             this.items.addObjet(ob);
19             if(items.magic(ob)){
20                 this.weightMax += 10;
21             }
22             this.weight += ob.getWeight();
23             this.gain += ob.getGain();
24             System.out.println(this.weight);
25             System.out.println(this.weightMax);
26             return true;
27         }
28         int w = this.weight + ob.getWeight();
29         System.out.println("ta dépasser"+w);
30         return false;
31     }
```

Apprentissage de enum et values : fait :

<https://docs.oracle.com/javase/8/docs/api/java/lang/Enum.html>

Exercice 7.35 & exercice 7.35.1:

Consultation du code source du projet zuul-with-enums-v1 pour voir comment le type CommandWord est utilisé. Les classes Command, CommandWords, Game, et Parser sont toutes adaptées dans cette version de notre projet, pour prendre en charge ce changement. le programme fonctionne toujours comme prévu.

Exercice 7.35.2:

En s'inspirant de *processCommand()* de la classe *Game* dans *zuul-withenums-v1*, on a remplacé dans *interpreteCommand()* de la classe *GameEngine* la suite de if else par un *switch* (autorisé sur un type énuméré !).

```
146         switch(commandWord){
147             case HELP:
148                 printHelp();
149                 break;
150             case LOOK:
151                 look();
152                 break;
153             case EAT:
154                 eat(command);
155                 break;
156             case BACK:
157                 back(command);
158                 break;
159             case GO:
160                 goRoom(command);
161                 break;
162             case TEST:
163                 test(command);
164                 break;
165             case TAKE:
166                 take(command);
167                 break;
168             case DROP:
169                 drop(command);
170                 break;
171             case ITEMS:
172                 items();
173                 break;
174             case QUIT:
175                 wantToQuit = quit(command);
176                 break;
177             default : {
178                 gui.println("I don't know what you mean...");
179                 return false;
180             }
181         }
```

Exercice 7.36, exercice 7.37, exercice 7.38, exercice 7.39, exercice 7.40 et exercice 7.41:

déjà fait.