

CMPT 276 Assignment 3  
Group 14 Refactoring

Hanxi Chen (301417307)  
Boyu Zhang (301414438)

## 1.Unused or Useless Variables.

```
src/main/java/Entity/Player.java
...
27 27 private Rectangle hitBox;
28 28
29 29 private int prevX, prevY;
30 30
31 31 public long lastDamageTime = 0;
32 32 private long durationBeforeDamage = 3000;
33 33 private BufferedImage sprites;
34 34 private int currentFrame;
35 35 public TileManager tileManager;
...
313 313 g2d.drawImage(sprites[currentFrame], x, y, null);
314 314
315 315
316 316
317 317
318 318 //w
319 319 // whether play can damage
320 320 // returns can or can't damage
321 321 w
322 322 public boolean canDamage() {
323 323     return System.currentTimeMillis() - this.lastDamageTime > this.durationBeforeDamage;
324 324 }
325 325
326 326 //w
327 327 // Player take damage
328 328 // decrease damage take damage
329 329 w
330 330 public void takeDamage(int damage) {
331 331     if(!canDamage())
332 332         return;
333 333     lives -= damage;
334 334     lastDamageTime = System.currentTimeMillis();
335 335     if (lives == 0) {
336 336         // TODO: player has died, handle game over condition
337 337         // change to death frame
338 338     }
339 339 }
...

```

## 2.Code Duplication

```
public class rewards extends Items
3 usages
protected int value;
3 usages
private int x,y;

src/main/java/rewards/bonus.java
...
18 18 public class bonus extends rewards {
19 19     private BufferedImage sprites;
20 20     private boolean pickedUp;
21 21     private int x,y;
22 22     private int currentFrame;
23 23     private TileManager tileManager;
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55
56 56
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72
73 73
74 74
75 75
76 76
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90
91 91
92 92
93 93
94 94
95 95
96 96
97 97
98 98
99 99
100 100
101 101
102 102
103 103
104 104
105 105
106 106
107 107
108 108
109 109
110 110
111 111
112 112
113 113
114 114
115 115
116 116
117 117
118 118
119 119
120 120
121 121
122 122
123 123
124 124
125 125
126 126
127 127
128 128
129 129
130 130
131 131
132 132
133 133
134 134
135 135
136 136
137 137
138 138
139 139
140 140
141 141
142 142
143 143
144 144
145 145
146 146
147 147
148 148
149 149
150 150
151 151
152 152
153 153
154 154
155 155
156 156
157 157
158 158
159 159
160 160
161 161
162 162
163 163
164 164
165 165
166 166
167 167
168 168
169 169
170 170
171 171
172 172
173 173
174 174
175 175
176 176
177 177
178 178
179 179
180 180
181 181
182 182
183 183
184 184
185 185
186 186
187 187
188 188
189 189
190 190
191 191
192 192
193 193
194 194
195 195
196 196
197 197
198 198
199 199
200 200
201 201
202 202
203 203
204 204
205 205
206 206
207 207
208 208
209 209
210 210
211 211
212 212
213 213
214 214
215 215
216 216
217 217
218 218
219 219
220 220
221 221
222 222
223 223
224 224
225 225
226 226
227 227
228 228
229 229
230 230
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
569 569
570 570
571 571
572 572
573 573
574 574
575 575
576 576
577 577
578 578
579 579
580 580
581 581
582 582
583 583
584 584
585 585
586 586
587 587
588 588
589 589
590 590
591 591
592 592
593 593
594 594
595 595
596 596
597 597
598 598
599 599
600 600
601 601
602 602
603 603
604 604
605 605
606 606
607 607
608 608
609 609
610 610
611 611
612 612
613 613
614 614
615 615
616 616
617 617
618 618
619 619
620 620
621 621
622 622
623 623
624 624
625 625
626 626
627 627
628 628
629 629
630 630
631 631
632 632
633 633
634 634
635 635
636 636
637 637
638 638
639 639
640 640
641 641
642 642
643 643
644 644
645 645
646 646
647 647
648 648
649 649
650 650
651 651
652 652
653 653
654 654
655 655
656 656
657 657
658 658
659 659
660 660
661 661
662 662
663 663
664 664
665 665
666 666
667 667
668 668
669 669
670 670
671 671
672 672
673 673
674 674
675 675
676 676
677 677
678 678
679 679
680 680
681 681
682 682
683 683
684 684
685 685
686 686
687 687
688 688
689 689
690 690
691 691
692 692
693 693
694 694
695 695
696 696
697 697
698 698
699 699
700 700
701 701
702 702
703 703
704 704
705 705
706 706
707 707
708 708
709 709
710 710
711 711
712 712
713 713
714 714
715 715
716 716
717 717
718 718
719 719
720 720
721 721
722 722
723 723
724 724
725 725
726 726
727 727
728 728
729 729
730 730
731 731
732 732
733 733
734 734
735 735
736 736
737 737
738 738
739 739
740 740
741 741
742 742
743 743
744 744
745 745
746 746
747 747
748 748
749 749
750 750
751 751
752 752
753 753
754 754
755 755
756 756
757 757
758 758
759 759
760 760
761 761
762 762
763 763
764 764
765 765
766 766
767 767
768 768
769 769
770 770
771 771
772 772
773 773
774 774
775 775
776 776
777 777
778 778
779 779
780 780
781 781
782 782
783 783
784 784
785 785
786 786
787 787
788 788
789 789
790 790
791 791
792 792
793 793
794 794
795 795
796 796
797 797
798 798
799 799
800 800
801 801
802 802
803 803
804 804
805 805
806 806
807 807
808 808
809 809
810 810
811 811
812 812
813 813
814 814
815 815
816 816
817 817
818 818
819 819
820 820
821 821
822 822
823 823
824 824
825 825
826 826
827 827
828 828
829 829
830 830
831 831
832 832
833 833
834 834
835 835
836 836
837 837
838 838
839 839
840 840
841 841
842 842
843 843
844 844
845 845
846 846
847 847
848 848
849 849
850 850
851 851
852 852
853 853
854 854
855 855
856 856
857 857
858 858
859 859
860 860
861 861
862 862
863 863
864 864
865 865
866 866
867 867
868 868
869 869
870 870
871 871
872 872
873 873
874 874
875 875
876 876
877 877
878 878
879 879
880 880
881 881
882 882
883 883
884 884
885 885
886 886
887 887
888 888
889 889
890 890
891 891
892 892
893 893
894 894
895 895
896 896
897 897
898 898
899 899
900 900
901 901
902 902
903 903
904 904
905 905
906 906
907 907
908 908
909 909
910 910
911 911
912 912
913 913
914 914
915 915
916 916
917 917
918 918
919 919
920 920
921 921
922 922
923 923
924 924
925 925
926 926
927 927
928 928
929 929
930 930
931 931
932 932
933 933
934 934
935 935
936 936
937 937
938 938
939 939
940 940
941 941
942 942
943 943
944 944
945 945
946 946
947 947
948 948
949 949
950 950
951 951
952 952
953 953
954 954
955 955
956 956
957 957
958 958
959 959
960 960
961 961
962 962
963 963
964 964
965 965
966 966
967 967
968 968
969 969
970 970
971 971
972 972
973 973
974 974
975 975
976 976
977 977
978 978
979 979
980 980
981 981
982 982
983 983
984 984
985 985
986 986
987 987
988 988
989 989
990 990
991 991
992 992
993 993
994 994
995 995
996 996
997 997
998 998
999 999
1000 1000

```

## 3.Poorly Structure Code.

```
while (running) {
    for (Enemy enemy : enemies) {
        if (enemy instanceof MovingEnemy) {
            if (enemy.getHitBox().intersects(player.getHitBox())) {
                // Player damage enemy test
                if (player.canDamage()) {
                    continue;
                }
                player.lastDamageTime = System.currentTimeMillis();
                System.out.println("Player collided with moving enemy");
            }
            try {
                AudioClip damageSound = AudioClipSystem.getAudioClipFromResource(
                    "File/assets/audio/damage.wav");
                Clip damageSoundClip = AudioClipSystem.getClip();
                damageSoundClip.setSource(damageSound);
                damageSoundClip.start();
            } catch (Exception e) {
                System.out.println("Error playing sound: " + e.getMessage());
            }
        }
        player.decreaseHealth(enemy.getDamage());
        healthBar.decreaseHealth();
        if (healthBar.isDead()) {
            // Player is dead, end game
            this.gameMusicClip.stop();
            try {
                AudioClip deathSound = AudioClipSystem.getAudioClipFromResource(
                    "File/assets/audio/gameover.wav");
                Clip deathSoundClip = AudioClipSystem.getClip();
                deathSoundClip.setSource(deathSound);
                deathSoundClip.start();
            } catch (Exception e) {
                System.out.println("Error playing sound: " + e.getMessage());
            }
            try {
                AudioClip gameOverSound = AudioClipSystem.getAudioClipFromResource(
                    "File/assets/audio/gameover.wav");
                Clip gameOverSoundClip = AudioClipSystem.getClip();
                gameOverSoundClip.setSource(gameOverSound);
                gameOverSoundClip.start();
            } catch (Exception e) {
                System.out.println("Error playing sound: " + e.getMessage());
            }
            stateManager.setState(new DeathScreenState());
            frame.dispose();
            running = false;
        }
    }
}

```

```
private void hitTrapEnemy(Enemy enemy) {...}
private void hitChestBox(Items item) {...}
private void hitBonus(Items item) {...}
// game running state runner
//
// no usages > handle +2
public void run() {
    // Render the running state.
    while (running) {
        for (Enemy enemy : enemies) {
            if (enemy instanceof MovingEnemy) {
                hitMovingEnemy(enemy);
            }
            if (enemy instanceof TrapEnemy) {
                hitTrapEnemy(enemy);
            }
            if (item instanceof regular) {
                hitChestBox(item);
            }
            if (item instanceof bonus) {
                hitBonus(item);
            }
        }
    }
}

```

Left: Previous

Right: After

## 5.Poorly Structure Code

```
56 85 // Set up player
57 86 player = new Player(304, 320, 32, 32, 3, tileManager.getTileNum());
58 87 // Set up enemies
59 88 enemies = new ArrayList<>();
60 89 enemies.add(new MovingEnemy(608, 512, 32, 32, 1, 10000));
61 90 enemies.add(new MovingEnemy(100, 80, 30, 17, 75));
62 91 enemies.add(new TrapEnemy(400, 60, 30, 17, 75));
63 92 enemies.add(new TrapEnemy(400, 850, 30, 17, 75));
64 93 enemies.add(new TrapEnemy(150, 180, 30, 17, 75));
65 94 enemies.add(new TrapEnemy(700, 400, 30, 17, 75));
66 95 items = new ArrayList<>();
67 96 items.add(new regular(730, 544, 10, 15, 75));
68 97 items.add(new regular(96, 64, 10, 15, 75));
69 98 items.add(new bonus(120, 352, 32, 32, 200, 100, 500, tileManager));
70 99 items.add(new bonus(704, 128, 32, 32, 200, 100, 100, tileManager));
71 100 items.add(new bonus(128, 480, 32, 32, 200, 100, 700, tileManager));
72 101
73 102
74 103
75 104
76 105
77 106
78 107
79 108
80 109
81 110
82 111
83 112
84 113
85 114
86 115
87 116
88 117
89 118
90 119
91 120
92 121
93 122
94 123
95 124
96 125
97 126
98 127
99 128
100 129
101 130
102 131
103 132
104 133
105 134
106 135
107 136
108 137
109 138
110 139
111 140
112 141
113 142
114 143
115 144
116 145
117 146
118 147
119 148
120 149
121 150
122 151
123 152
124 153
125 154
126 155
127 156
128 157
129 158
130 159
131 160
132 161
133 162
134 163
135 164
136 165
137 166
138 167
139 168
140 169
141 170
142 171
143 172
144 173
145 174
146 175
147 176
148 177
149 178
150 179
151 180
152 181
153 182
154 183
155 184
156 185
157 186
158 187
159 188
160 189
161 190
162 191
163 192
164 193
165 194
166 195
167 196
168 197
169 198
170 199
171 200
172 201
173 202
174 203
175 204
176 205
177 206
178 207
179 208
180 209
181 210
182 211
183 212
184 213
185 214
186 215
187 216
188 217
189 218
190 219
191 220
192 221
193 222
194 223
195 224
196 225
197 226
198 227
199 228
200 229
201 230
202 231
203 232
204 233
205 234
206 235
207 236
208 237
209 238
210 239
211 240
212 241
213 242
214 243
215 244
216 245
217 246
218 247
219 248
220 249
221 250
222 251
223 252
224 253
225 254
226 255
227 256
228 257
229 258
230 259
231 260
232 261
233 262
234 263
235 264
236 265
237 266
238 267
239 268
240 269
241 270
242 271
243 272
244 273
245 274
246 275
247 276
248 277
249 278
250 279
251 280
252 281
253 282
254 283
255 284
256 285
257 286
258 287
259 288
260 289
261 290
262 291
263 292
264 293
265 294
266 295
267 296
268 297
269 298
270 299
271 300
272 301
273 302
274 303
275 304
276 305
277 306
278 307
279 308
280 309
281 310
282 311
283 312
284 313
285 314
286 315
287 316
288 317
289 318
290 319
291 320
292 321
293 322
294 323
295 324
296 325
297 326
298 327
299 328
300 329
301 330
302 331
303 332
304 333
305 334
306 335
307 336
308 337
309 338
310 339
311 340
312 341
313 342
314 343
315 344
316 345
317 346
318 347
319 348
320 349
321 350
322 351
323 352
324 353
325 354
326 355
327 356
328 357
329 358
330 359
331 360
332 361
333 362
334 363
335 364
336 365
337 366
338 367
339 368
340 369
341 370
342 371
343 372
344 373
345 374
346 375
347 376
348 377
349 378
350 379
351 380
352 381
353 382
354 383
355 384
356 385
357 386
358 387
359 388
360 389
361 390
362 391
363 392
364 393
365 394
366 395
367 396
368 397
369 398
370 399
371 400
372 401
373 402
374 403
375 404
376 405
377 406
378 407
379 408
380 409
381 410
382 411
383 412
384 413
385 414
386 415
387 416
388 417
389 418
390 419
391 420
392 421
393 422
394 423
395 424
396 425
397 426
398 427
399 428
400 429
401 430
402 431
403 432
404 433
405 434
406 435
407 436
408
```

1. Unused or Useless Variables

We set up a Player Class for the movable character (Player controlled) in the game. We originally planned to set fatal moving enemies and ordinary moving enemies, but we did not implement a variety of moving enemies during development. The deleted useless variables are designed to record whether the player can continue to receive damage after receiving damage. Currently in the game moving enemies that give damage as critical hits so these variables are no longer needed. So I delete the useless variables during refactoring.

2. Code Duplication

The reward system of the game has an abstract "reward" class, and "bonus" rewards should inherit the characteristics of the "reward" class. But we additionally overwrite the x and y coordinates of "bonus" reward class that the "reward" class should have. (The coordinates are used to set the position of the item on the map). During refactoring I delete the variables we overwrite.

3. Poorly Structure Code

When the game is running to detect the interaction between the player and the map elements, we originally stacked all the codes and all the conditions (cases) together in the running state class. After refactoring, the run method became concise and easy to understand, and all collisions Conditions are written as additional methods that will be called when necessary.

4. Long List of method parameters

Our original trap enemy had too many parameters, which resulted in the need to enter many lines of repetitive code when creating it. Since we only have one kind of fixed trap in the game, I set the length, width and height variables for it before the trap's constructor. In this way, when the game is running, we only need to input the x and y axis coordinates as parameters when we want to generate the trap.

5. Poorly Structure Code

In the running state class of our generated game, the original version has no structure and piles up all the code used to generate map elements. After the refactor, I created a separate method for the generator of each element. When it is necessary to change the generated element, I only need to go to the generator to change without adjusting the running class. At the same time the code becomes more understandable and manageable.

6. Bad/Confusing Variable Name

In our initial map design, the game had a variety of terrain and barriers, but during the development process only walls were kept as obstacles. The map is composed of a circle of walls and has a door as an exit. The method originally designed to detect whether the tile where the obstacle is located can be collided has not been changed in time, resulting in the code review process to think that "IsTileSolid" includes both collision monster and trap conditions. "isTileWall" becomes more understandable after refactoring.