



# MANDATORY OVERTIME

Team 3

# TEAM INTRODUCTION



**FELICIA**

- Retired US Air Force
- UMD Student, MS, Software Engineering
- AWS Marketplace



**AYRAT**

- US Army Medic
- Registered Nurse
- Amazon Devices & Demand Planning



**GABRIEL**

- Retired US Army Medic
- Electrical Engineer
- AWS Insights

# GAME OVERVIEW

Mandatory Overtime is a game where you play as a new Software Developer Engineer at Amazon. It is Halloween night and you are planning to attend a party later that evening.

Explore the office building and find the missing items before your manager finds you and assigns you mandatory overtime!



# TECHNOLOGIES



**GIT**



**JAVA SWING**



**AZURE DEVOPS**



**GITHUB**

# STAKEHOLDER INTERACTION



Daily Check-in



Sprint Planning



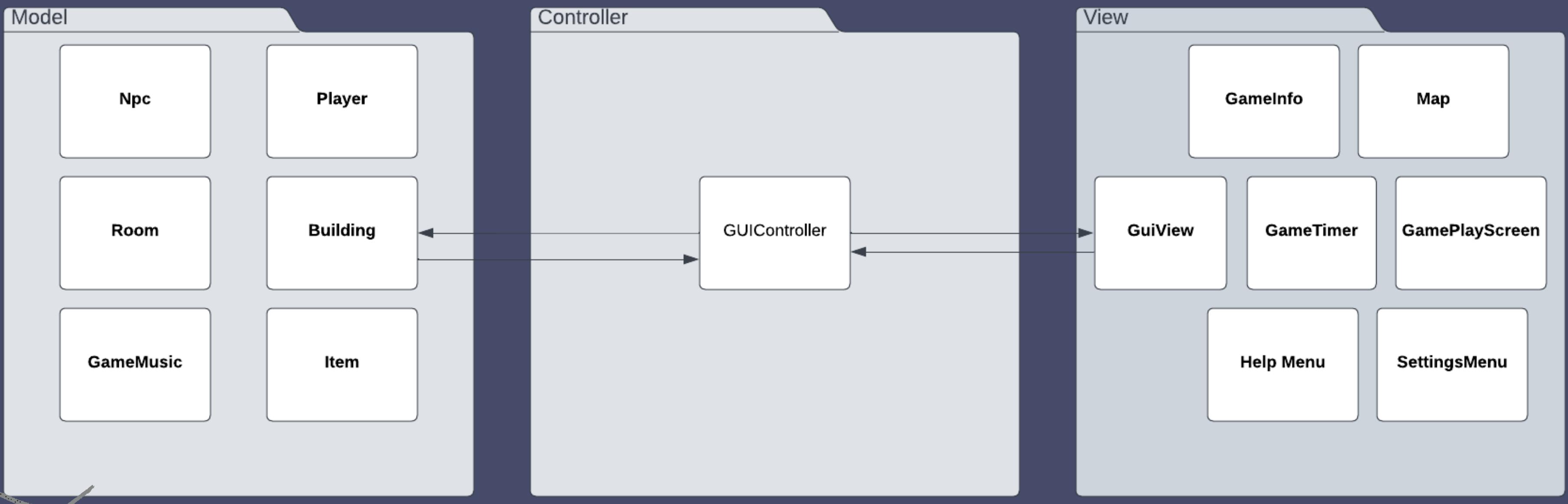
Release and Feedback

# PROJECT MANAGEMENT

- Stand-ups
- Ticket Management /Backlog Review(Azure Devops)
- 3 Iterations
- Associate Commits with User Stories



# </> DESIGN

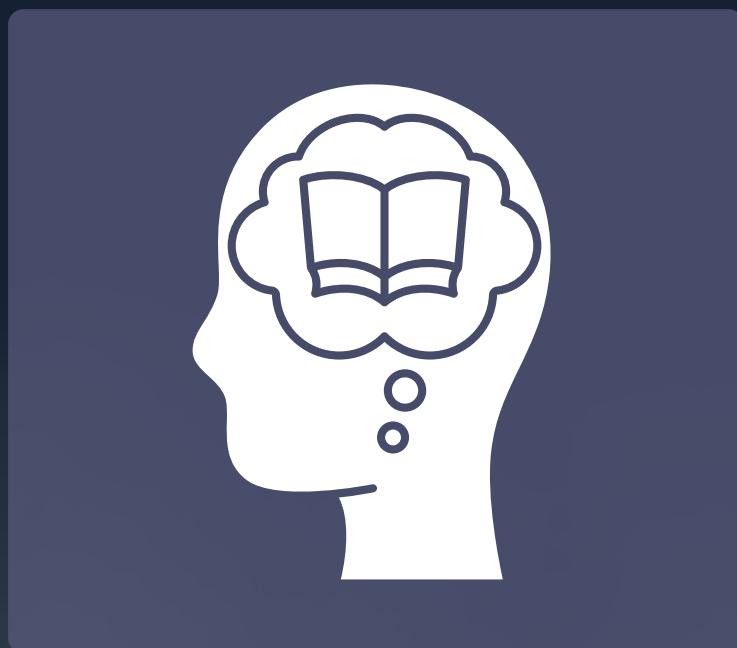




# CHALLENGES



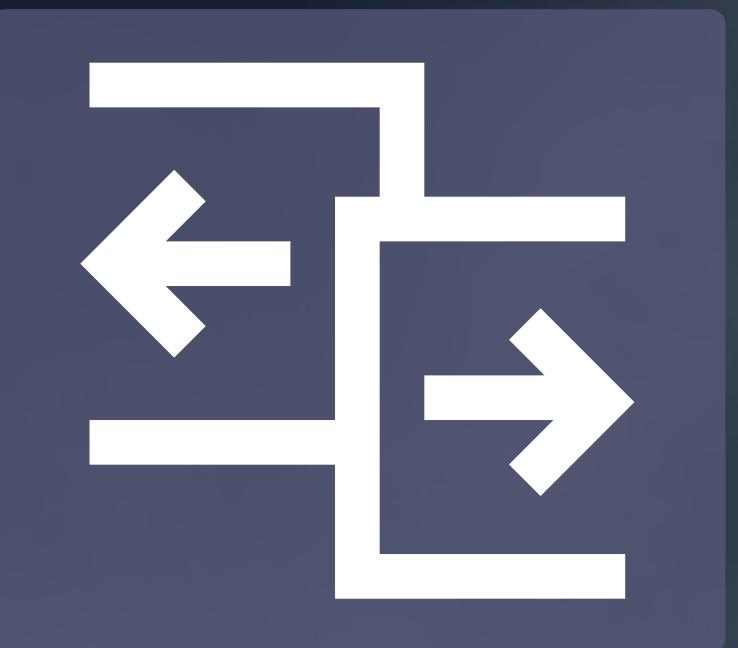
**REFACTORING**



**LEARNING JAVA  
SWING**



**CONSUMERS/  
EVENT LISTENERS**



**MVC  
SEPARATION OF  
CONCERNS**





# BEFORE & AFTER



GamePlay.java

```
86     public void startGameFromNew() throws IOException {
87         building = new Building();
88         building.createGameStructureFromNew();
89         try {
90             //Ask if user wants to start new game, continue from previous game
91
92             //Asks name and formats
93             System.out.println("\nWhat is your name");
94             String name = inputParser.readLine().trim();
95             String nameTemp;
96             while (name.length() < 1) {
97                 nameTemp = inputParser.readLine().trim();
98                 name = nameTemp;
99             }
100
101            //Updates Player instance name
102            building.setName(name);
103
104            //prints startup Info
105            System.out.println(userView.startUpInfo(name));
106
107            //run printRoomDescription() here to show current room.
108            building.startingRoomDescription();
109
110            //start game loop
111            gamePlayCommands();
112        } catch (IOException e) {
113            throw new RuntimeException(e);
114        }
115    }
116
117    public void gamePlayCommands() {
118        Scanner scanner = new Scanner(System.in);
119        String command;
120        while (true) {
121            command = scanner.nextLine();
122            if (command.equals("quit")) {
123                break;
124            }
125            else if (command.equals("help")) {
126                System.out.println("Available commands: quit, help, go [direction], look, inventory, take [item], drop [item]");
127            }
128            else if (command.equals("go north") || command.equals("go south") || command.equals("go east") || command.equals("go west")) {
129                building.movePlayer(command);
130            }
131            else if (command.equals("look")) {
132                building.printRoomDescription();
133            }
134            else if (command.equals("inventory")) {
135                building.printInventory();
136            }
137            else if (command.equals("take [item]")) {
138                building.takeItem(command);
139            }
140            else if (command.equals("drop [item]")) {
141                building.dropItem(command);
142            }
143            else {
144                System.out.println("Unknown command: " + command);
145            }
146        }
147    }
148
149    public void printRoomDescription() {
150        building.printRoomDescription();
151    }
152
153    public void startingRoomDescription() {
154        building.printStartingRoomDescription();
155    }
156
157    public void movePlayer(String direction) {
158        building.movePlayer(direction);
159    }
160
161    public void printInventory() {
162        building.printInventory();
163    }
164
165    public void takeItem(String item) {
166        building.takeItem(item);
167    }
168
169    public void dropItem(String item) {
170        building.dropItem(item);
171    }
172
173    public void presentGameInfo() {
174        view.presentGameInfo();
175    }
176
177    public void setUpGamePlayHandlers() {
178        view.setGamePlayHandlers();
179    }
180
181    public void getSettingsDialog() {
182        view.getSettingsDialog();
183    }
184
185    public void toggleCheatCheckbox() {
186        view.toggleCheatCheckbox();
187    }
188
189    public void getNameTextField() {
190        return view.getNameTextField();
191    }
192
193    public void setName(String name) {
194        building.setName(name);
195    }
196
197    public void setGameTime(int gameTime) {
198        building.setGameTime(gameTime);
199    }
200
201    public void setMinutes(int minutes) {
202        building.setMinutes(minutes);
203    }
204
205    public void setSeconds(int seconds) {
206        building.setSeconds(seconds);
207    }
208
209    public void setDifficulty(String difficulty) {
210        building.setDifficulty(difficulty);
211    }
212
213    public void setStartingRoomDescription() {
214        building.setStartingRoomDescription();
215    }
216
217    public void printInventory() {
218        building.printInventory();
219    }
220
221    public void printRoomDescription() {
222        building.printRoomDescription();
223    }
224
225    public void printStartingRoomDescription() {
226        building.printStartingRoomDescription();
227    }
228
229    public void movePlayer(String direction) {
230        building.movePlayer(direction);
231    }
232
233    public void presentGameInfo() {
234        view.presentGameInfo();
235    }
236
237    public void setGameTime(int gameTime) {
238        building.setGameTime(gameTime);
239    }
240
241    public void setMinutes(int minutes) {
242        building.setMinutes(minutes);
243    }
244
245    public void setSeconds(int seconds) {
246        building.setSeconds(seconds);
247    }
248
249    public void setDifficulty(String difficulty) {
250        building.setDifficulty(difficulty);
251    }
252
253    public void setStartingRoomDescription() {
254        building.setStartingRoomDescription();
255    }
256
257    public void printInventory() {
258        building.printInventory();
259    }
260
261    public void printRoomDescription() {
262        building.printRoomDescription();
263    }
264
265    public void printStartingRoomDescription() {
266        building.printStartingRoomDescription();
267    }
268
269    public void movePlayer(String direction) {
270        building.movePlayer(direction);
271    }
272
273    public void presentGameInfo() {
274        view.presentGameInfo();
275    }
276
277    public void setGameTime(int gameTime) {
278        building.setGameTime(gameTime);
279    }
280
281    public void setMinutes(int minutes) {
282        building.setMinutes(minutes);
283    }
284
285    public void setSeconds(int seconds) {
286        building.setSeconds(seconds);
287    }
288
289    public void setDifficulty(String difficulty) {
290        building.setDifficulty(difficulty);
291    }
292
293    public void setStartingRoomDescription() {
294        building.setStartingRoomDescription();
295    }
296
297    public void printInventory() {
298        building.printInventory();
299    }
300
301    public void printRoomDescription() {
302        building.printRoomDescription();
303    }
304
305    public void printStartingRoomDescription() {
306        building.printStartingRoomDescription();
307    }
308
309    public void movePlayer(String direction) {
310        building.movePlayer(direction);
311    }
312
313    public void presentGameInfo() {
314        view.presentGameInfo();
315    }
316
317    public void setGameTime(int gameTime) {
318        building.setGameTime(gameTime);
319    }
320
321    public void setMinutes(int minutes) {
322        building.setMinutes(minutes);
323    }
324
325    public void setSeconds(int seconds) {
326        building.setSeconds(seconds);
327    }
328
329    public void setDifficulty(String difficulty) {
330        building.setDifficulty(difficulty);
331    }
332
333    public void setStartingRoomDescription() {
334        building.setStartingRoomDescription();
335    }
336
337    public void printInventory() {
338        building.printInventory();
339    }
340
341    public void printRoomDescription() {
342        building.printRoomDescription();
343    }
344
345    public void printStartingRoomDescription() {
346        building.printStartingRoomDescription();
347    }
348
349    public void movePlayer(String direction) {
350        building.movePlayer(direction);
351    }
352
353    public void presentGameInfo() {
354        view.presentGameInfo();
355    }
356
357    public void setGameTime(int gameTime) {
358        building.setGameTime(gameTime);
359    }
360
361    public void setMinutes(int minutes) {
362        building.setMinutes(minutes);
363    }
364
365    public void setSeconds(int seconds) {
366        building.setSeconds(seconds);
367    }
368
369    public void setDifficulty(String difficulty) {
370        building.setDifficulty(difficulty);
371    }
372
373    public void setStartingRoomDescription() {
374        building.setStartingRoomDescription();
375    }
376
377    public void printInventory() {
378        building.printInventory();
379    }
380
381    public void printRoomDescription() {
382        building.printRoomDescription();
383    }
384
385    public void printStartingRoomDescription() {
386        building.printStartingRoomDescription();
387    }
388
389    public void movePlayer(String direction) {
390        building.movePlayer(direction);
391    }
392
393    public void presentGameInfo() {
394        view.presentGameInfo();
395    }
396
397    public void setGameTime(int gameTime) {
398        building.setGameTime(gameTime);
399    }
399
400    public void setMinutes(int minutes) {
401        building.setMinutes(minutes);
402    }
403
404    public void setSeconds(int seconds) {
405        building.setSeconds(seconds);
406    }
407
408    public void setDifficulty(String difficulty) {
409        building.setDifficulty(difficulty);
410    }
411
412    public void setStartingRoomDescription() {
413        building.setStartingRoomDescription();
414    }
415
416    public void printInventory() {
417        building.printInventory();
418    }
419
420    public void printRoomDescription() {
421        building.printRoomDescription();
422    }
423
424    public void printStartingRoomDescription() {
425        building.printStartingRoomDescription();
426    }
427
428    public void movePlayer(String direction) {
429        building.movePlayer(direction);
430    }
431
432    public void presentGameInfo() {
433        view.presentGameInfo();
434    }
435
436    public void setGameTime(int gameTime) {
437        building.setGameTime(gameTime);
438    }
439
440    public void setMinutes(int minutes) {
441        building.setMinutes(minutes);
442    }
443
444    public void setSeconds(int seconds) {
445        building.setSeconds(seconds);
446    }
447
448    public void setDifficulty(String difficulty) {
449        building.setDifficulty(difficulty);
450    }
451
452    public void setStartingRoomDescription() {
453        building.setStartingRoomDescription();
454    }
455
456    public void printInventory() {
457        building.printInventory();
458    }
459
460    public void printRoomDescription() {
461        building.printRoomDescription();
462    }
463
464    public void printStartingRoomDescription() {
465        building.printStartingRoomDescription();
466    }
467
468    public void movePlayer(String direction) {
469        building.movePlayer(direction);
470    }
471
472    public void presentGameInfo() {
473        view.presentGameInfo();
474    }
475
476    public void setGameTime(int gameTime) {
477        building.setGameTime(gameTime);
478    }
479
480    public void setMinutes(int minutes) {
481        building.setMinutes(minutes);
482    }
483
484    public void setSeconds(int seconds) {
485        building.setSeconds(seconds);
486    }
487
488    public void setDifficulty(String difficulty) {
489        building.setDifficulty(difficulty);
490    }
491
492    public void setStartingRoomDescription() {
493        building.setStartingRoomDescription();
494    }
495
496    public void printInventory() {
497        building.printInventory();
498    }
499
500    public void printRoomDescription() {
501        building.printRoomDescription();
502    }
503
504    public void printStartingRoomDescription() {
505        building.printStartingRoomDescription();
506    }
507
508    public void movePlayer(String direction) {
509        building.movePlayer(direction);
510    }
511
512    public void presentGameInfo() {
513        view.presentGameInfo();
514    }
515
516    public void setGameTime(int gameTime) {
517        building.setGameTime(gameTime);
518    }
519
520    public void setMinutes(int minutes) {
521        building.setMinutes(minutes);
522    }
523
524    public void setSeconds(int seconds) {
525        building.setSeconds(seconds);
526    }
527
528    public void setDifficulty(String difficulty) {
529        building.setDifficulty(difficulty);
530    }
531
532    public void setStartingRoomDescription() {
533        building.setStartingRoomDescription();
534    }
535
536    public void printInventory() {
537        building.printInventory();
538    }
539
540    public void printRoomDescription() {
541        building.printRoomDescription();
542    }
543
544    public void printStartingRoomDescription() {
545        building.printStartingRoomDescription();
546    }
547
548    public void movePlayer(String direction) {
549        building.movePlayer(direction);
550    }
551
552    public void presentGameInfo() {
553        view.presentGameInfo();
554    }
555
556    public void setGameTime(int gameTime) {
557        building.setGameTime(gameTime);
558    }
559
560    public void setMinutes(int minutes) {
561        building.setMinutes(minutes);
562    }
563
564    public void setSeconds(int seconds) {
565        building.setSeconds(seconds);
566    }
567
568    public void setDifficulty(String difficulty) {
569        building.setDifficulty(difficulty);
570    }
571
572    public void setStartingRoomDescription() {
573        building.setStartingRoomDescription();
574    }
575
576    public void printInventory() {
577        building.printInventory();
578    }
579
580    public void printRoomDescription() {
581        building.printRoomDescription();
582    }
583
584    public void printStartingRoomDescription() {
585        building.printStartingRoomDescription();
586    }
587
588    public void movePlayer(String direction) {
589        building.movePlayer(direction);
590    }
591
592    public void presentGameInfo() {
593        view.presentGameInfo();
594    }
595
596    public void setGameTime(int gameTime) {
597        building.setGameTime(gameTime);
598    }
599
599
600    public void setMinutes(int minutes) {
601        building.setMinutes(minutes);
602    }
603
604    public void setSeconds(int seconds) {
605        building.setSeconds(seconds);
606    }
607
608    public void setDifficulty(String difficulty) {
609        building.setDifficulty(difficulty);
610    }
611
612    public void setStartingRoomDescription() {
613        building.setStartingRoomDescription();
614    }
615
616    public void printInventory() {
617        building.printInventory();
618    }
619
620    public void printRoomDescription() {
621        building.printRoomDescription();
622    }
623
624    public void printStartingRoomDescription() {
625        building.printStartingRoomDescription();
626    }
627
628    public void movePlayer(String direction) {
629        building.movePlayer(direction);
630    }
631
632    public void presentGameInfo() {
633        view.presentGameInfo();
634    }
635
636    public void setGameTime(int gameTime) {
637        building.setGameTime(gameTime);
638    }
639
640    public void setMinutes(int minutes) {
641        building.setMinutes(minutes);
642    }
643
644    public void setSeconds(int seconds) {
645        building.setSeconds(seconds);
646    }
647
648    public void setDifficulty(String difficulty) {
649        building.setDifficulty(difficulty);
650    }
651
652    public void setStartingRoomDescription() {
653        building.setStartingRoomDescription();
654    }
655
656    public void printInventory() {
657        building.printInventory();
658    }
659
660    public void printRoomDescription() {
661        building.printRoomDescription();
662    }
663
664    public void printStartingRoomDescription() {
665        building.printStartingRoomDescription();
666    }
667
668    public void movePlayer(String direction) {
669        building.movePlayer(direction);
670    }
671
672    public void presentGameInfo() {
673        view.presentGameInfo();
674    }
675
676    public void setGameTime(int gameTime) {
677        building.setGameTime(gameTime);
678    }
679
680    public void setMinutes(int minutes) {
681        building.setMinutes(minutes);
682    }
683
684    public void setSeconds(int seconds) {
685        building.setSeconds(seconds);
686    }
687
688    public void setDifficulty(String difficulty) {
689        building.setDifficulty(difficulty);
690    }
691
692    public void setStartingRoomDescription() {
693        building.setStartingRoomDescription();
694    }
695
696    public void printInventory() {
697        building.printInventory();
698    }
699
700    public void printRoomDescription() {
701        building.printRoomDescription();
702    }
703
704    public void printStartingRoomDescription() {
705        building.printStartingRoomDescription();
706    }
707
708    public void movePlayer(String direction) {
709        building.movePlayer(direction);
710    }
711
712    public void presentGameInfo() {
713        view.presentGameInfo();
714    }
715
716    public void setGameTime(int gameTime) {
717        building.setGameTime(gameTime);
718    }
719
720    public void setMinutes(int minutes) {
721        building.setMinutes(minutes);
722    }
723
724    public void setSeconds(int seconds) {
725        building.setSeconds(seconds);
726    }
727
728    public void setDifficulty(String difficulty) {
729        building.setDifficulty(difficulty);
730    }
731
732    public void setStartingRoomDescription() {
733        building.setStartingRoomDescription();
734    }
735
736    public void printInventory() {
737        building.printInventory();
738    }
739
740    public void printRoomDescription() {
741        building.printRoomDescription();
742    }
743
744    public void printStartingRoomDescription() {
745        building.printStartingRoomDescription();
746    }
747
748    public void movePlayer(String direction) {
749        building.movePlayer(direction);
750    }
751
752    public void presentGameInfo() {
753        view.presentGameInfo();
754    }
755
756    public void setGameTime(int gameTime) {
757        building.setGameTime(gameTime);
758    }
759
760    public void setMinutes(int minutes) {
761        building.setMinutes(minutes);
762    }
763
764    public void setSeconds(int seconds) {
765        building.setSeconds(seconds);
766    }
767
768    public void setDifficulty(String difficulty) {
769        building.setDifficulty(difficulty);
770    }
771
772    public void setStartingRoomDescription() {
773        building.setStartingRoomDescription();
774    }
775
776    public void printInventory() {
777        building.printInventory();
778    }
779
780    public void printRoomDescription() {
781        building.printRoomDescription();
782    }
783
784    public void printStartingRoomDescription() {
785        building.printStartingRoomDescription();
786    }
787
788    public void movePlayer(String direction) {
789        building.movePlayer(direction);
790    }
791
792    public void presentGameInfo() {
793        view.presentGameInfo();
794    }
795
796    public void setGameTime(int gameTime) {
797        building.setGameTime(gameTime);
798    }
799
799
800    public void setMinutes(int minutes) {
801        building.setMinutes(minutes);
802    }
803
804    public void setSeconds(int seconds) {
805        building.setSeconds(seconds);
806    }
807
808    public void setDifficulty(String difficulty) {
809        building.setDifficulty(difficulty);
810    }
811
812    public void setStartingRoomDescription() {
813        building.setStartingRoomDescription();
814    }
815
816    public void printInventory() {
817        building.printInventory();
818    }
819
820    public void printRoomDescription() {
821        building.printRoomDescription();
822    }
823
824    public void printStartingRoomDescription() {
825        building.printStartingRoomDescription();
826    }
827
828    public void movePlayer(String direction) {
829        building.movePlayer(direction);
830    }
831
832    public void presentGameInfo() {
833        view.presentGameInfo();
834    }
835
836    public void setGameTime(int gameTime) {
837        building.setGameTime(gameTime);
838    }
839
840    public void setMinutes(int minutes) {
841        building.setMinutes(minutes);
842    }
843
844    public void setSeconds(int seconds) {
845        building.setSeconds(seconds);
846    }
847
848    public void setDifficulty(String difficulty) {
849        building.setDifficulty(difficulty);
850    }
851
852    public void setStartingRoomDescription() {
853        building.setStartingRoomDescription();
854    }
855
856    public void printInventory() {
857        building.printInventory();
858    }
859
860    public void printRoomDescription() {
861        building.printRoomDescription();
862    }
863
864    public void printStartingRoomDescription() {
865        building.printStartingRoomDescription();
866    }
867
868    public void movePlayer(String direction) {
869        building.movePlayer(direction);
870    }
871
872    public void presentGameInfo() {
873        view.presentGameInfo();
874    }
875
876    public void setGameTime(int gameTime) {
877        building.setGameTime(gameTime);
878    }
879
880    public void setMinutes(int minutes) {
881        building.setMinutes(minutes);
882    }
883
884    public void setSeconds(int seconds) {
885        building.setSeconds(seconds);
886    }
887
888    public void setDifficulty(String difficulty) {
889        building.setDifficulty(difficulty);
890    }
891
892    public void setStartingRoomDescription() {
893        building.setStartingRoomDescription();
894    }
895
896    public void printInventory() {
897        building.printInventory();
898    }
899
900    public void printRoomDescription() {
901        building.printRoomDescription();
902    }
903
904    public void printStartingRoomDescription() {
905        building.printStartingRoomDescription();
906    }
907
908    public void movePlayer(String direction) {
909        building.movePlayer(direction);
910    }
911
912    public void presentGameInfo() {
913        view.presentGameInfo();
914    }
915
916    public void setGameTime(int gameTime) {
917        building.setGameTime(gameTime);
918    }
919
920    public void setMinutes(int minutes) {
921        building.setMinutes(minutes);
922    }
923
924    public void setSeconds(int seconds) {
925        building.setSeconds(seconds);
926    }
927
928    public void setDifficulty(String difficulty) {
929        building.setDifficulty(difficulty);
930    }
931
932    public void setStartingRoomDescription() {
933        building.setStartingRoomDescription();
934    }
935
936    public void printInventory() {
937        building.printInventory();
938    }
939
940    public void printRoomDescription() {
941        building.printRoomDescription();
942    }
943
944    public void printStartingRoomDescription() {
945        building.printStartingRoomDescription();
946    }
947
948    public void movePlayer(String direction) {
949        building.movePlayer(direction);
950    }
951
952    public void presentGameInfo() {
953        view.presentGameInfo();
954    }
955
956    public void setGameTime(int gameTime) {
957        building.setGameTime(gameTime);
958    }
959
960    public void setMinutes(int minutes) {
961        building.setMinutes(minutes);
962    }
963
964    public void setSeconds(int seconds) {
965        building.setSeconds(seconds);
966    }
967
968    public void setDifficulty(String difficulty) {
969        building.setDifficulty(difficulty);
970    }
971
972    public void setStartingRoomDescription() {
973        building.setStartingRoomDescription();
974    }
975
976    public void printInventory() {
977        building.printInventory();
978    }
979
980    public void printRoomDescription() {
981        building.printRoomDescription();
982    }
983
984    public void printStartingRoomDescription() {
985        building.printStartingRoomDescription();
986    }
987
988    public void movePlayer(String direction) {
989        building.movePlayer(direction);
990    }
991
992    public void presentGameInfo() {
993        view.presentGameInfo();
994    }
995
996    public void setGameTime(int gameTime) {
997        building.setGameTime(gameTime);
998    }
999
999
1000    public void setMinutes(int minutes) {
1001        building.setMinutes(minutes);
1002    }
1003
1004    public void setSeconds(int seconds) {
1005        building.setSeconds(seconds);
1006    }
1007
1008    public void setDifficulty(String difficulty) {
1009        building.setDifficulty(difficulty);
1010    }
1011
1012    public void setStartingRoomDescription() {
1013        building.setStartingRoomDescription();
1014    }
1015
1016    public void printInventory() {
1017        building.printInventory();
1018    }
1019
1020    public void printRoomDescription() {
1021        building.printRoomDescription();
1022    }
1023
1024    public void printStartingRoomDescription() {
1025        building.printStartingRoomDescription();
1026    }
1027
1028    public void movePlayer(String direction) {
1029        building.movePlayer(direction);
1030    }
1031
1032    public void presentGameInfo() {
1033        view.presentGameInfo();
1034    }
1035
1036    public void setGameTime(int gameTime) {
1037        building.setGameTime(gameTime);
1038    }
1039
1040    public void setMinutes(int minutes) {
1041        building.setMinutes(minutes);
1042    }
1043
1044    public void setSeconds(int seconds) {
1045        building.setSeconds(seconds);
1046    }
1047
1048    public void setDifficulty(String difficulty) {
1049        building.setDifficulty(difficulty);
1050    }
1051
1052    public void setStartingRoomDescription() {
1053        building.setStartingRoomDescription();
1054    }
1055
1056    public void printInventory() {
1057        building.printInventory();
1058    }
1059
1060    public void printRoomDescription() {
1061        building.printRoomDescription();
1062    }
1063
1064    public void printStartingRoomDescription() {
1065        building.printStartingRoomDescription();
1066    }
1067
1068    public void movePlayer(String direction) {
1069        building.movePlayer(direction);
1070    }
10
```

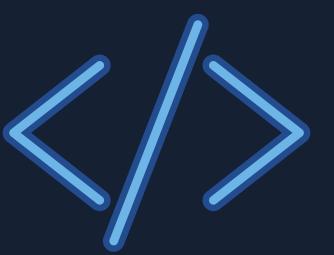


# BEFORE & AFTER



```
public static void playAudioFX(String soundFile) {  
  
    try {  
        URL audio = GameMusic.class.getResource("/" + soundFile);  
  
        AudioInputStream audioInput = AudioSystem.getAudioInputStream(audio);  
        clip = AudioSystem.getClip();  
        clip.open(audioInput);  
        startAudio();  
  
    } catch (UnsupportedAudioFileException e) {}  
    catch (Exception e) {}  
}  
  
blworth +1  
  
public static void playAudioMusic(String soundFile) {  
  
    try {  
        URL audio = GameMusic.class.getResource("/" + soundFile);  
        AudioInputStream audioInput = AudioSystem.getAudioInputStream(audio);  
        clip = AudioSystem.getClip();  
        clip.open(audioInput);  
        startBackgroundAudio();  
    } catch (UnsupportedAudioFileException e) {}  
    catch (Exception e) {}  
}
```

```
public static void playAudioFX(String soundFile) {  
  
    try {  
        URL audio = GameMusic.class.getResource( name: "/" + soundFile);  
        AudioInputStream audioInput = AudioSystem.getAudioInputStream( url: audio);  
        soundClip = AudioSystem.getClip();  
        soundClip.open( stream: audioInput);  
  
        // Set default sound FX volume  
        gainControlSoundFX = (FloatControl) soundClip.getControl( control: FloatControl.Type.MASTER_GAIN);  
        gainControlSoundFX.setValue(soundFxVolume);  
        soundClip.start();  
    } catch (UnsupportedAudioFileException e) {}  
    catch (Exception e) {}  
}  
  
1 usage  blworth +2  
  
public static void playAudioMusic(String soundFile) {  
  
    try {  
        URL audio = GameMusic.class.getResource( name: "/" + soundFile);  
        AudioInputStream audioInput = AudioSystem.getAudioInputStream( url: audio);  
        clip = AudioSystem.getClip();  
        clip.open( stream: audioInput);  
        startBackgroundAudio();  
    } catch (UnsupportedAudioFileException e) {}  
}
```



# BEFORE & AFTER



Building.java

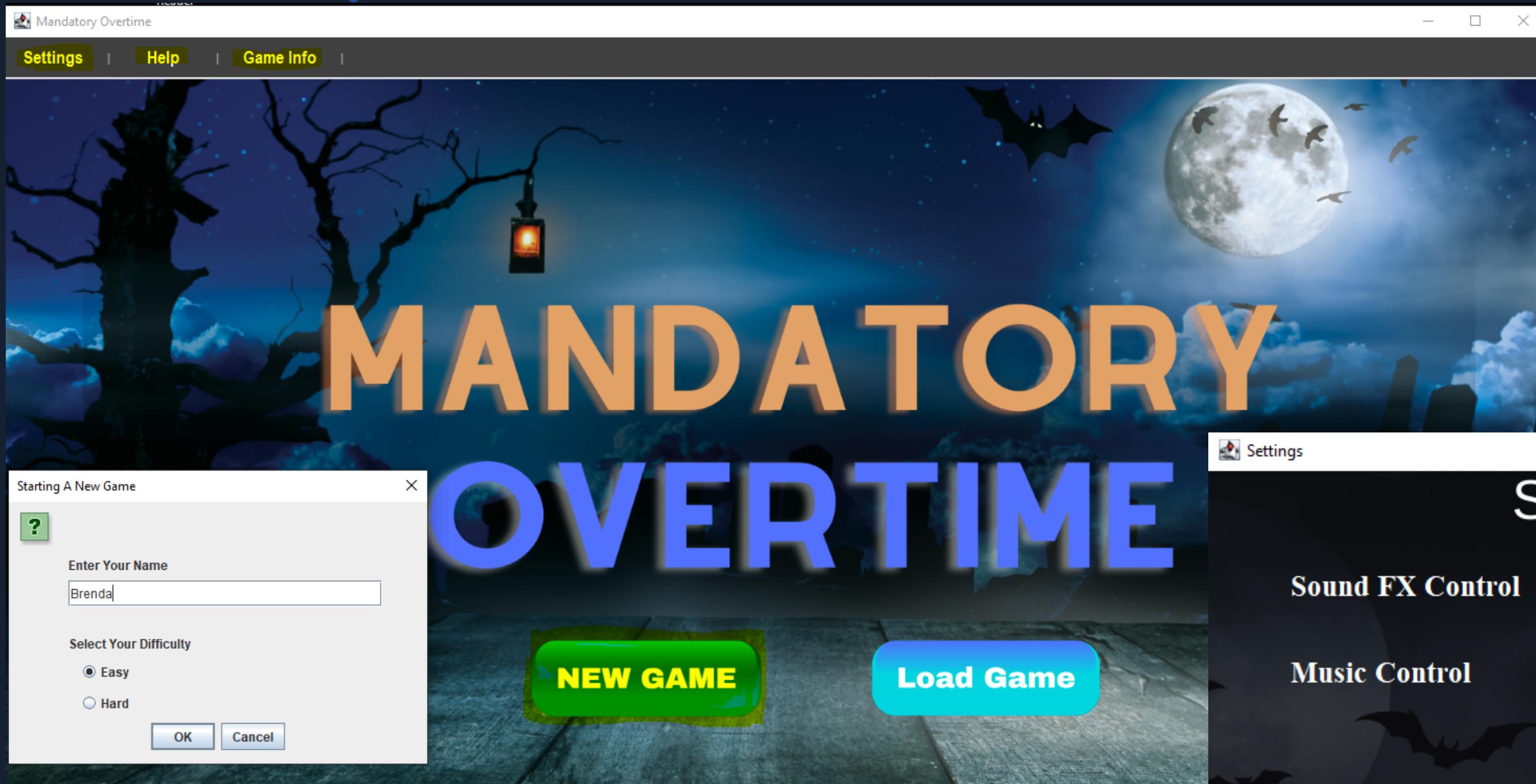
```
1 usage
378     public void getItem(String item) throws IOException, InterruptedException {
379         getRoomDescriptionInfo();
380         String playerCurrentLocation = player.getCurrentLocation();
381
382         //conditional to check if item is in array //check if location correct // check if npc do
383
384         if (items.containsKey(item) && items.get(item).getAcquired() == false && items.get(item)
385             .getLocation().equals(playerCurrentLocation)
386             && items.get(item).isNpc() == false) {
387
388             //conditionals to check it item has prerequisite
389             if (items.get(item).getPreReq() == null) {
390
391                 //conditional to check for challenge
392                 if (items.get(item).getChallenge() == true) {
393                     runItemChallenge(item);
394                 } else {
395                     player.addToInventory(item);
396                     GameMusic.playItemSound();
397                     items.get(item).setAcquired(true);
398                     //remove item from list in rooms method
399                     System.out.println(items.get(item).getPurpose());
400                     System.out.println(player.getInventory().toString());
401
402                 }
403                 checkItemPreReqIsFulfilled(item);
404             }
405             } else {
406                 System.out.printf("You cannot get %s.\n", item);
407                 System.out.print(">");
408             }
409         }
```

Building.java

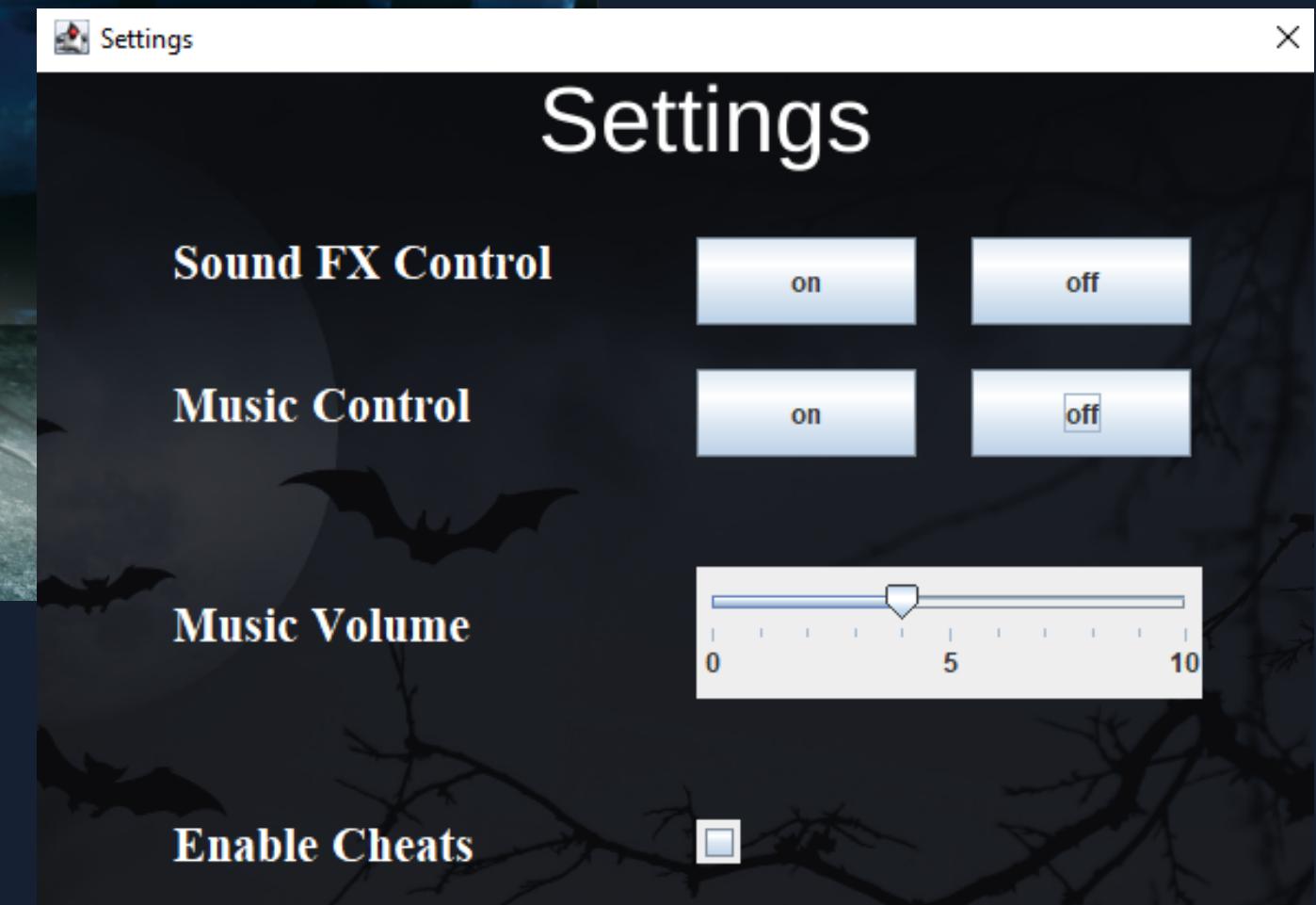
```
270
271     1 usage
272     public boolean getItem(String item)
273         throws IOException, InterruptedException, CantGetItemException {
274         String playerCurrentLocation = player.getCurrentLocation();
275
276         boolean itemAlreadyAcquired = items.get(item).getAcquired();
277         boolean itemIsAtTheLocation = items.get(item).getLocation()
278             .equals(playerCurrentLocation);
279         boolean npcHasItem = items.get(item).isNpc();
280
281         if (!itemAlreadyAcquired && itemIsAtTheLocation && !npcHasItem) {
282             //conditionals to check it item has prerequisite
283             if (items.get(item).getPreReq() == null) {
284                 //conditional to check for challenge
285                 if (items.get(item).getChallenge()) {
286                     runItemChallenge(item);
287                 } else {
288                     player.addToInventory(item);
289                     GameMusic.playItemSound();
290                     items.get(item).setAcquired(true);
291                     building.get(playerCurrentLocation).setItem(null);
292                 }
293             } else {
294                 checkItemPreReqIsFulfilled(item);
295             }
296         } else {
297             throw new CantGetItemException();
298         }
299     }
300 }
```



# HIGHLIGHTS/FEATURES



- Menu Bar
- Option to continue a saved game
- Personalized experience
- Game options-Difficulty level
- Sound control





# HIGHLIGHTS/FEATURES

Mandatory Overtime

Settings | Help | Game Info | **Map** | Save

Timer - 01 : 59

Quit To Main

You're at the Office on the 5th floor and feel the darkness encroaching on you, or just the darkmode of your laptop.

Are Right, A Lot

Bias For Action

Insist on Highest Standards

Frugality

Learn and Be Curious

Customer Obsession

Ownership

Think Big

Dive Deep

Earn Trust

EXITS

elevator conferen... desk

INVENTORY

Java For Gamers

amazon

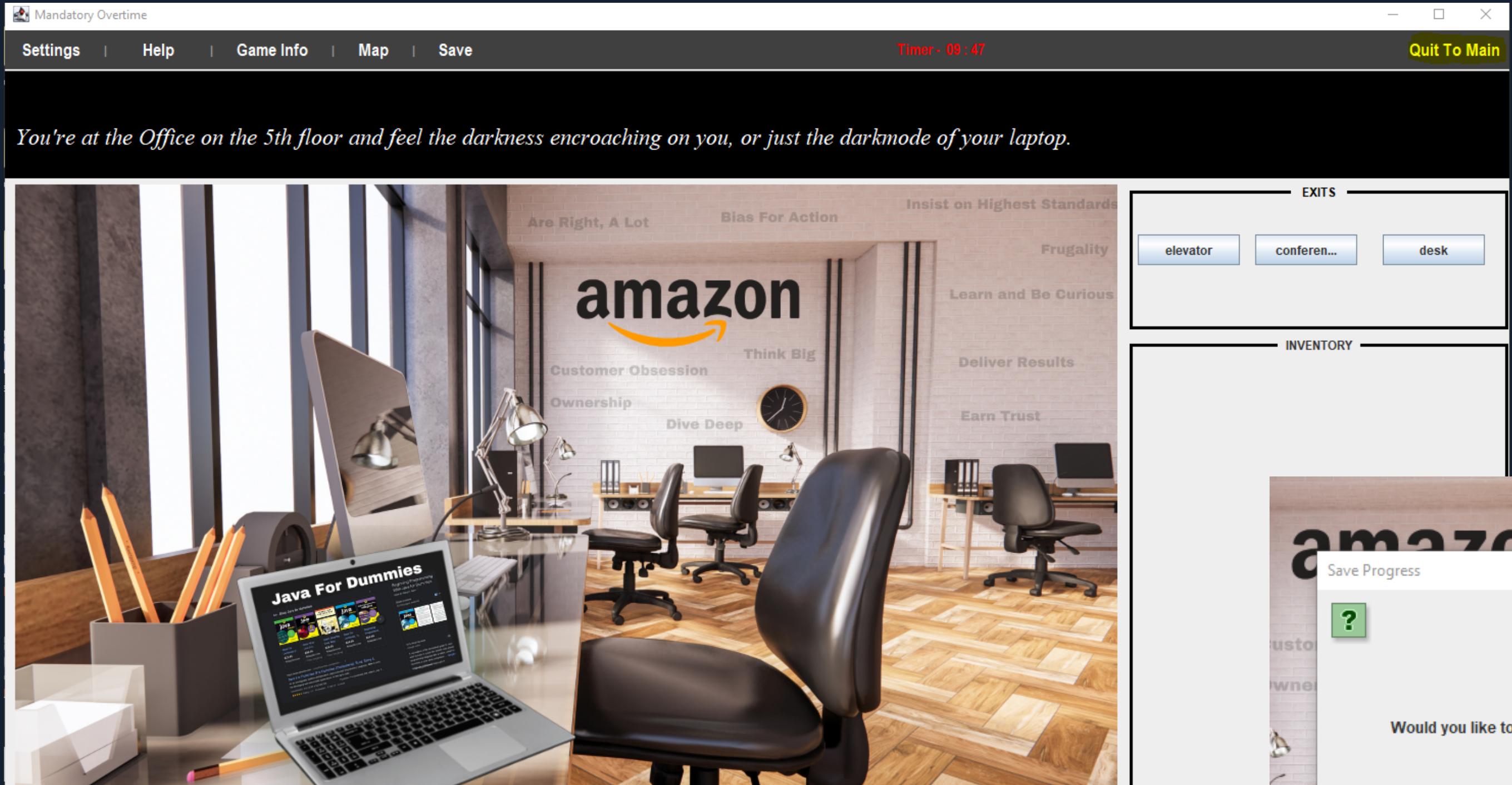
Conference Room	Desk	Office
IT Closet	Dev Floor	
Kitchen	Break Room	
Table	Party Floor	
Vending	Lobby	Exit

- Map
- Timer

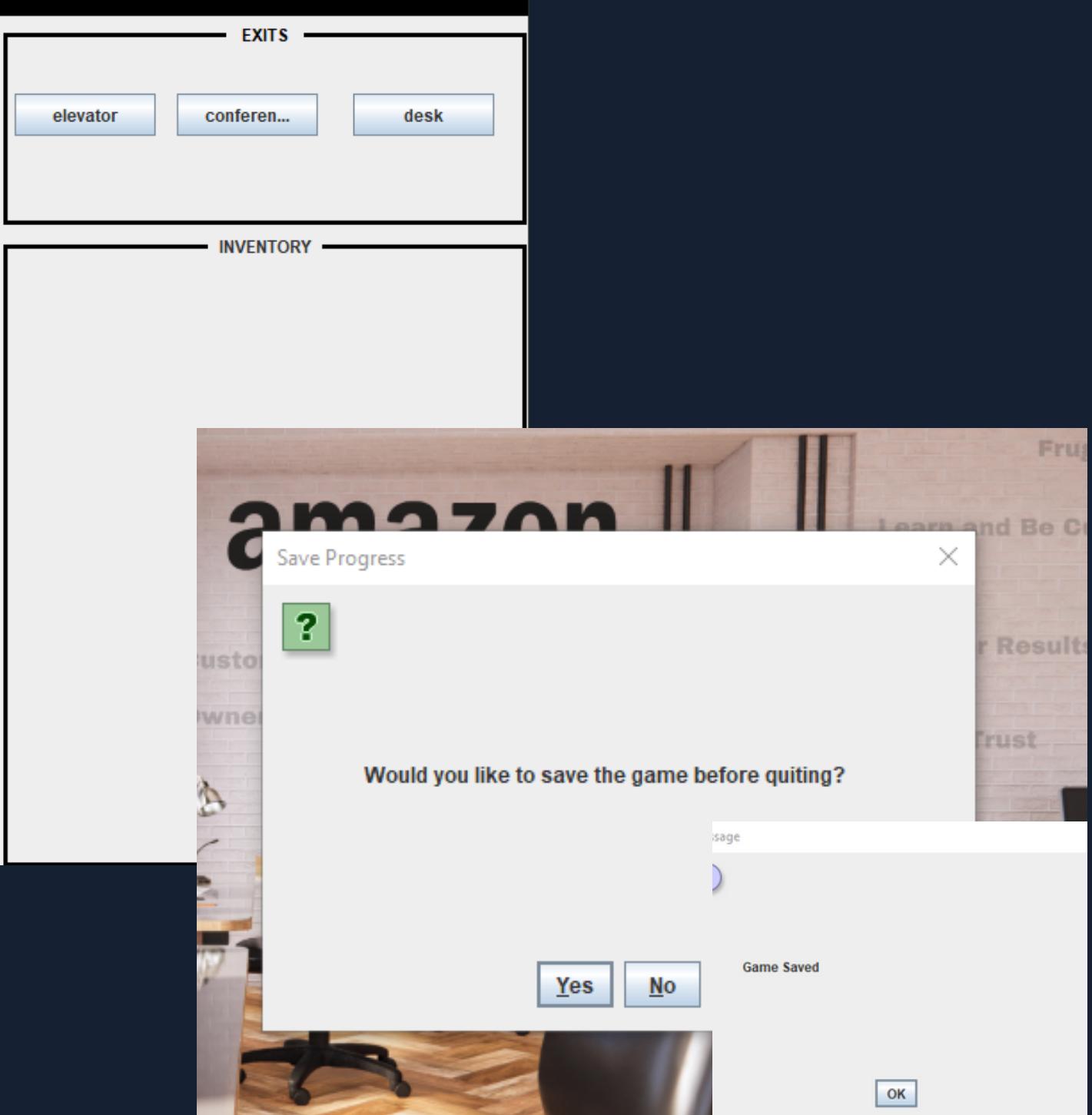




# HIGHLIGHTS/FEATURES



- Quit to main menu at anytime and start a new game





# HIGHLIGHTS

GuiView.java

```
1 usage
public GuiView(){
    JFrame frame = new JFrame( title: "Mandatory Overtime");
    frame.setPreferredSize(new Dimension( width: 1500, height: 800));
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setBackground(Color.BLACK);
    frame.setVisible(true);
    frame.pack();
    container = frame.getContentPane();
}

1 usage
public void presentGameScreen(){
    JLAYEREDPANE screen = gamePlayScreen.getGameScreen();
    container.removeAll();
    container.add(screen);
    container.repaint();
    container.revalidate();
}

3 usages
public void presentMainMenu(){
    JLAYEREDPANE menu = MainMenu.getHomeScreen();
    presentLoadingScreen();
    container.removeAll();
    container.add(menu);
    container.repaint();
    container.revalidate();
}
```

MainMenu.java

```
1 usage
private void buildMainMenu(){
    homeScreen = new JLAYEREDPANE();
    MenuBar.hideGameBtns();

    // BackGroundImage
    JLabel imageSection = new JLabel(mainMenu);
    imageSection.setSize( width: 1500, height: 800);

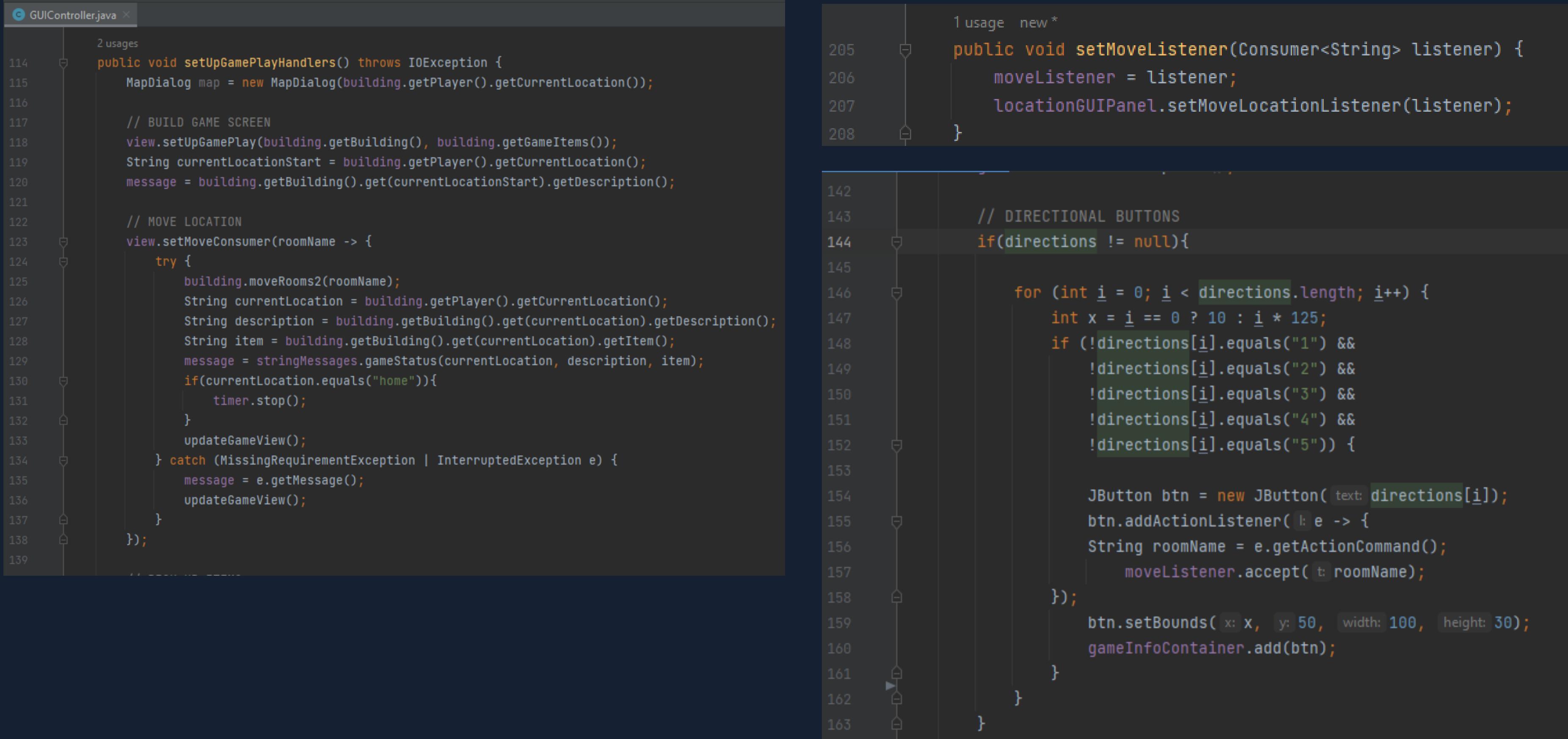
    // Start Button
    startBtn = new JButton(startButtonImage);
    startBtn.setBounds( x: 500, y: 575, width: 225, height: 75);
    startBtn.setFocusPainted(false);
    startBtn.setContentAreaFilled(false);
    startBtn.setBorderPainted(false);

    // Load Game Button
    loadBtn = new JButton(loadButtonImage);
    loadBtn.setBounds( x: 825, y: 575, width: 225, height: 75);
    loadBtn.setFocusPainted(false);
    loadBtn.setContentAreaFilled(false);
    loadBtn.setBorderPainted(false);

    homeScreen.add(MenuBar.getMenuBar(), Integer.valueOf( i: 1));
    homeScreen.add(imageSection, Integer.valueOf( i: 0));
    homeScreen.add(startBtn, Integer.valueOf( i: 2));
    homeScreen.add(loadBtn, Integer.valueOf( i: 2));
}
```



# </> HIGHLIGHTS



The image shows a Java code editor with two files displayed side-by-side.

**File 1: GUIController.java**

```
114     2 usages
115     public void setUpGamePlayHandlers() throws IOException {
116         MapDialog map = new MapDialog(building.getPlayer().getCurrentLocation());
117
118         // BUILD GAME SCREEN
119         view.setUpGamePlay(building.getBuilding(), building.getGameItems());
120         String currentLocationStart = building.getPlayer().getCurrentLocation();
121         message = building.getBuilding().get(currentLocationStart).getDescription();
122
123         // MOVE LOCATION
124         view.setMoveConsumer(roomName -> {
125             try {
126                 building.moveRooms2(roomName);
127                 string currentLocation = building.getPlayer().getCurrentLocation();
128                 string description = building.getBuilding().get(currentLocation).getDescription();
129                 String item = building.getBuilding().get(currentLocation).getItem();
130                 message = stringMessages.gameStatus(currentLocation, description, item);
131                 if(currentLocation.equals("home")){
132                     timer.stop();
133                 }
134                 updateGameView();
135             } catch (MissingRequirementException | InterruptedException e) {
136                 message = e.getMessage();
137                 updateGameView();
138             }
139         });
140     }
```

**File 2: Another File (lines 142-163)**

```
142
143     // DIRECTIONAL BUTTONS
144     if(directions != null){
145
146         for (int i = 0; i < directions.length; i++) {
147             int x = i == 0 ? 10 : i * 125;
148             if (!directions[i].equals("1") &&
149                 !directions[i].equals("2") &&
150                 !directions[i].equals("3") &&
151                 !directions[i].equals("4") &&
152                 !directions[i].equals("5")) {
153
154                 JButton btn = new JButton( text: directions[i]);
155                 btn.addActionListener( l: e -> {
156                     String roomName = e.getActionCommand();
157                     moveListener.accept( t: roomName);
158                 });
159
160                 btn.setBounds( x: x, y: 50, width: 100, height: 30);
161                 gameInfoContainer.add(btn);
162             }
163         }
164     }
```





# HIGHLIGHTS

Building.java

```
1 usage
public String interactWithNpc(String npcName) {
    if (GameMusic.getNpcAudioClip() != null) {
        GameMusic.getNpcAudioClip().stop();
    }
    String response = null;
    for (String npc : npcs.keySet()) {
        if (npc.equals(npcName) && player.getCurrentLocation()
            .equals(npcs.get(npc).getLocation())) {
            if (!player.getInventory().contains(npcs.get(npc).getPrereq())
                && npcs.get(npc).getNpcCount() == 0) {
                response = String.format(npcs.get(npc).getInitialDialogue(), player.getName());
                // Play initial sound
                GameMusic.playNPCSound(npcs.get(npc).getInitialAudio());
                npcs.get(npc).getNpcCount();
                break;
            } else if (player.getInventory().contains((npcs.get(npc).getPrereq()))) {
                response = String.format((npcs.get(npc).getDialogueWithItem()),
                    player.getName());
                player.removeFromInventory((npcs.get(npc).getPrereq()));
                player.addToInventory((npcs.get(npc).getItems()));
                // Play audio with item
                GameMusic.playNPCSound(npcs.get(npc).getAudioWithItem());
                npcs.get(npc).setItems(null);
                break;
            } else if ((npcs.get(npc).getItems()) == null) {
                response = String.format((npcs.get(npc).getDialogueQuestDone()),
                    player.getName());
                // Play audio with quest completed
                GameMusic.playNPCSound(npcs.get(npc).getAudioQuestDone());
                break;
            } else if (!player.getInventory().contains((npcs.get(npc).getPrereq()))
                && npcs.get(npc).getNpcCount() >= 1) {
                // Play audio no item
                response = String.format(npcs.get(npc).getDialogueNoItem(), player.getName());
                GameMusic.playNPCSound(npcs.get(npc).getAudioNoItem());
                break;
            }
        }
    }
}
```

GUIController.java

```
161
162 // INTERACT WITH NPC
163 view.getGamePlayScreen().getLocationGUIPanel().setNpcListener(npcName -> {
164     message = building.interactWithNpc(npcName);
165     updateGameView();
166 }
167 );

248 1 usage
private JButton createNPCButton(Room room){
249     JButton btn;
250     int[] coords = room.getNpcCoord();
251     InputStream img = getClass().getResourceAsStream( name: "/images/" + room.getNpcImage());
252     try {
253         ImageIcon npcImg = new ImageIcon(ImageIO.read(img));
254         // ITEM BUTTON CREATED
255         btn = new JButton(npcImg);
256         btn.setActionCommand(room.getNPC());
257         btn.setFocusPainted(false);
258         btn.setContentAreaFilled(false);
259         btn.setBounds(coords[0], coords[1], coords[2], coords[3]);
260         btn.setToolTipText(room.getNPC());
261         btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
262         btn.addActionListener(e -> {
263             npcListener.accept(room.getNPC());
264         });
265     }
266     catch (IOException e) {
267         throw new RuntimeException(e);
268     }
269     btn.setBorderPainted(false);
270     return btn;
271 }
```

# AGILE PRINCIPLES

01

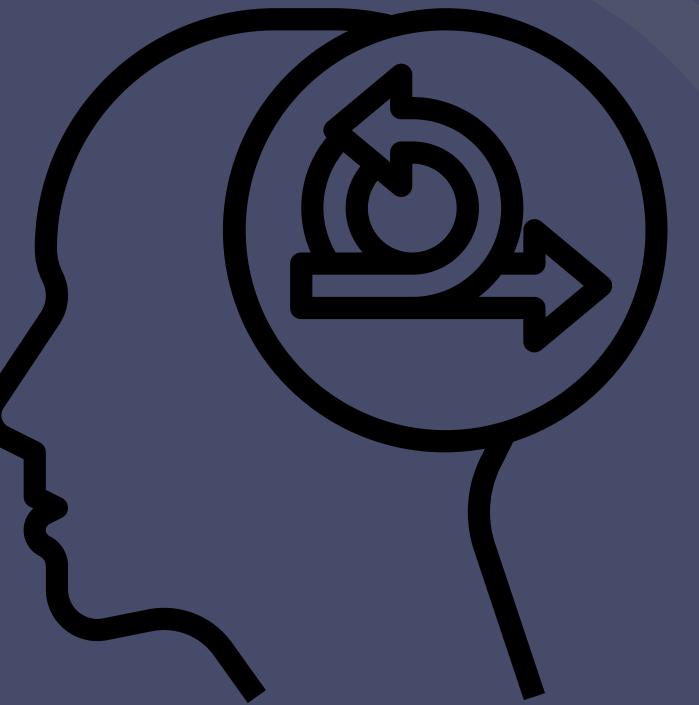
**Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**

02

**Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.**

04

**Business people and developers must work together daily throughout the project.**



**DEMO**

# WHAT'S NEXT



**IMPROVE  
PERFORMANCE**



**HOST ON WEBSITE**



**PLAYER SCORE**

# QUESTIONS

