
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

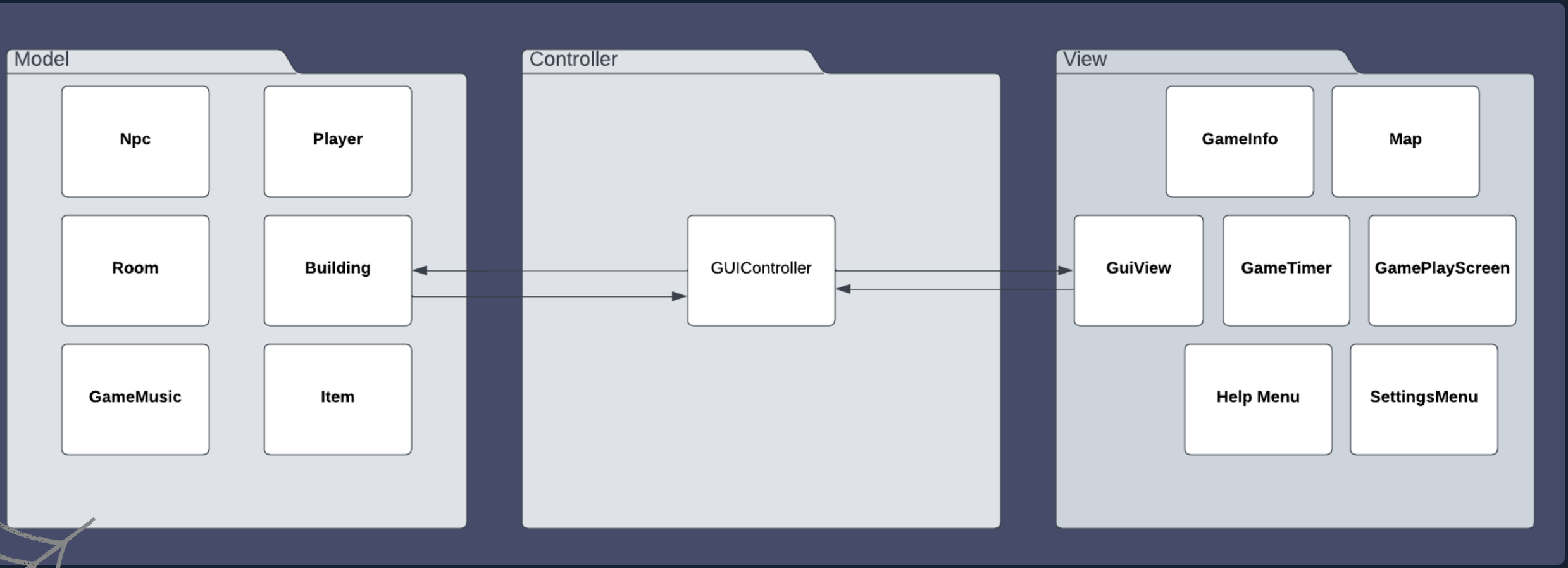
Sprint Planning

Release and Feedback

PROJECT MANAGEMENT

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

DESIGN

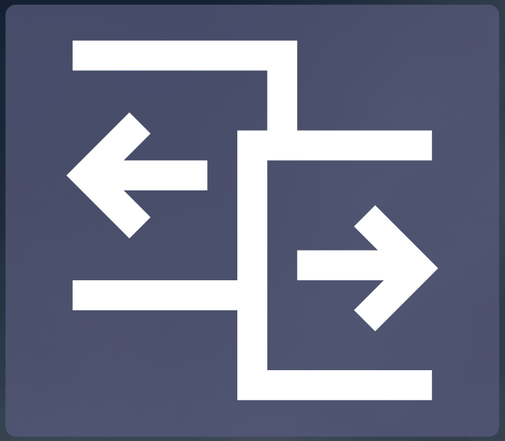

Model
Npc
Player
Room
Building
GameMusic
Item
Controller
GUIController
View
GameInfo
Map
GuiView
GameTimer
GamePlayScreen
Help Menu
SettingsMenu

CHALLENGES

REFACTORING

LEARNING JAVA SWING

CONSUMERS / EVENT LISTENERS



MVC SERPARATION 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);
```

GamePlay.java | GUIController.java

```
}

2 usages  Felicia Kenney +1 *
public void startNewGame() throws IOException {
    // Create New Building
    building = new Building();
    int result = JOptionPane.showConfirmDialog( parentComponent: null,  message: view.getGameStartPanel(),
        title: "Starting A New Game",  optionType: JOptionPane.OK_CANCEL_OPTION);
    if (result == JOptionPane.OK_OPTION) {
        String name = view.getNameTextField().getText();
        String gameLevel = view.getSelectedButton();
        if (name.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Please enter a valid name to start game")
            return;
        }
        int gameTime = gameLevel.equals("easy") ? EASY_GAME_TIME : HARD_GAME_TIME;
        timer.setMinutes(gameTime);
        building.createGameStructureFromNew( difficulty: gameLevel);
        building.setName(name);
        view.presentGameInfo();
        setUpGamePlayHandlers();
        view.getMenuBar().getSettingsDialog().getToggleCheatCheckbox().setEnabled(true);
        view.getNameTextField().setText(null);
    }
}
```

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

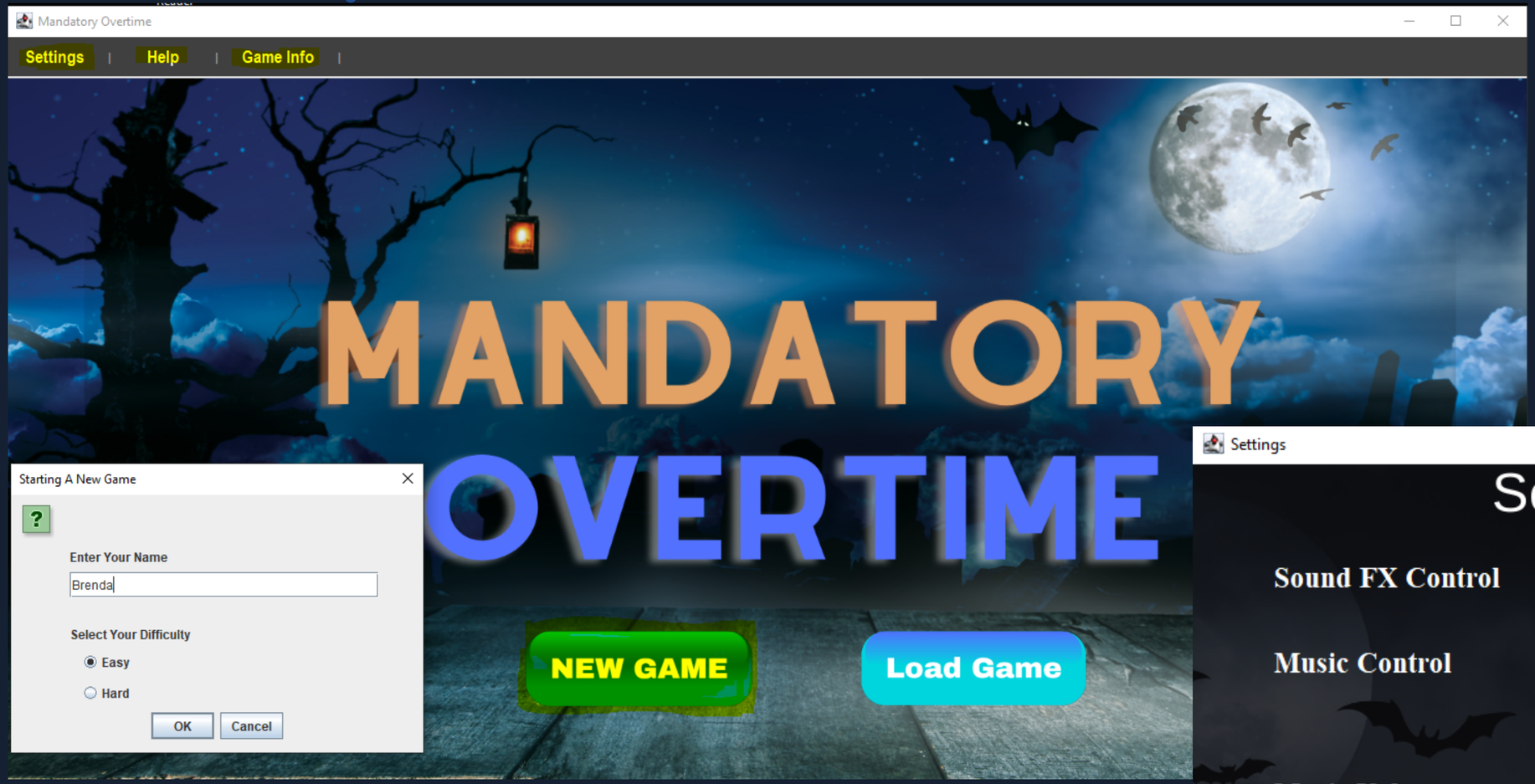
Building.java

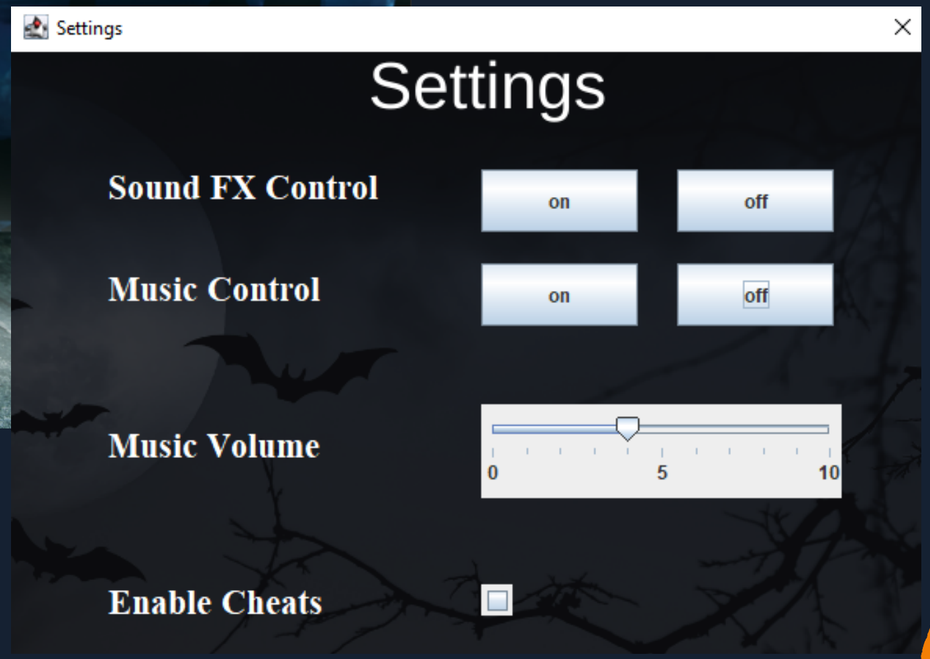
```
1 usage
public void getItem(String item) throws IOException, InterruptedException {
    getRoomDescriptionInfo();
    String playerCurrentLocation = player.getCurrentLocation();

    //conditional to check if item is in array //check if location correct // check if npc do

    if (items.containsKey(item) && items.get(item).getAcquired() == false && items.get(item)
        .getLocation().equals(playerCurrentLocation)
        && items.get(item).isNpc() == false) {

        //conditionals to check it item has prerequisite
        if (items.get(item).getPreReq() == null) {

            //conditional to check for challenge
            if (items.get(item).getChallenge() == true) {
                runItemChallenge(item);
            } else {
                player.addToInventory(item);
                GameMusic.playItemSound();
                items.get(item).setAcquired(true);
                //remove item from list in rooms method
                System.out.println(items.get(item).getPurpose());
                System.out.println(player.getInventory().toString());
            }
        } else {
            checkItemPreReqIsFulfilled(item);
        }
    } else {
        System.out.printf("You cannot get %s.\n", item);
        System.out.print(">");
    }
}
```

Building.java

```
1 usage
public boolean getItem(String item)
    throws IOException, InterruptedException, CantGetItemException {
    String playerCurrentLocation = player.getCurrentLocation();

    boolean itemAlreadyAcquired = items.get(item).getAcquired();
    boolean itemIsAtTheLocation = items.get(item).getLocation()
        .equals(playerCurrentLocation);
    boolean npcHasItem = items.get(item).isNpc();
    if (!itemAlreadyAcquired && itemIsAtTheLocation && !npcHasItem) {
        //conditionals to check it item has prerequisite
        if (items.get(item).getPreReq() == null) {
            //conditional to check for challenge
            if (items.get(item).getChallenge()) {
                runItemChallenge(item);
            } else {
                player.addToInventory(item);
                GameMusic.playItemSound();
                items.get(item).setAcquired(true);
                building.get(playerCurrentLocation).setItem(null);
            }
        } else {
            checkItemPreReqIsFulfilled(item);
        }
    } else {
        throw new CantGetItemException();
    }
    return player.getInventory().contains(item);
}
```

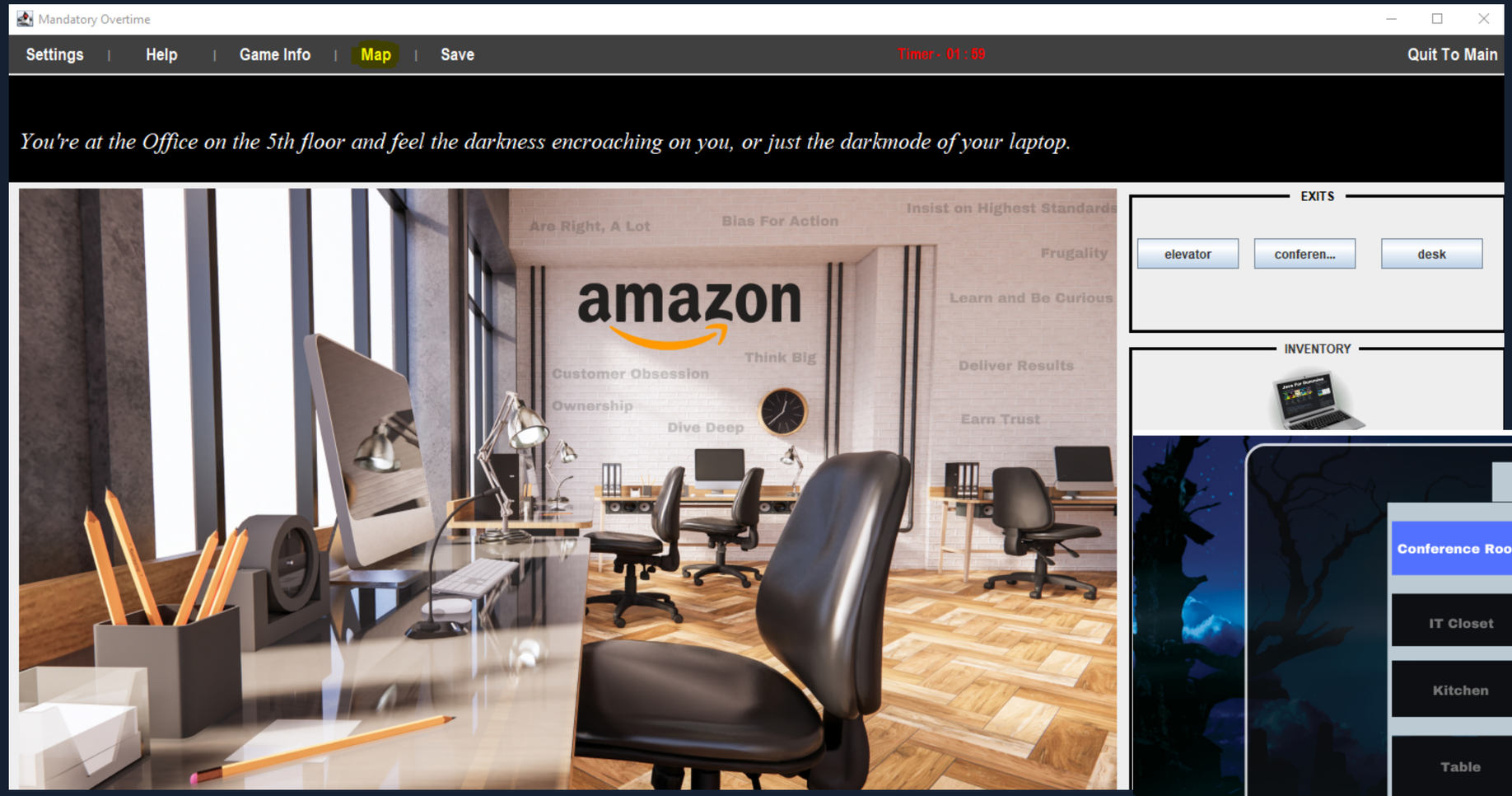
HIGHLIGHTS/FEATURES

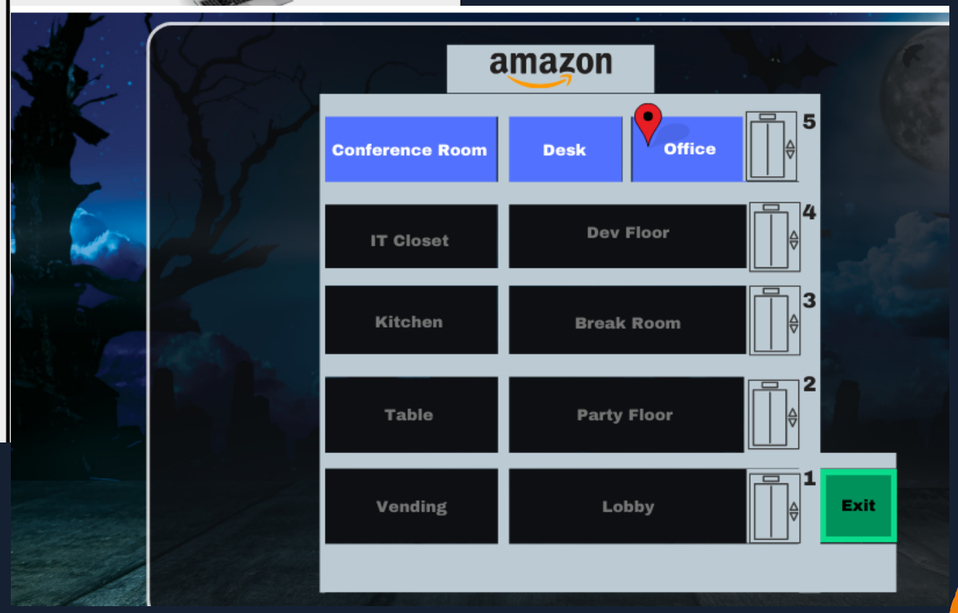




- Menu Bar
- Option to continue a saved game
- Personalized experience
- Game options-Difficulty level
- Sound control

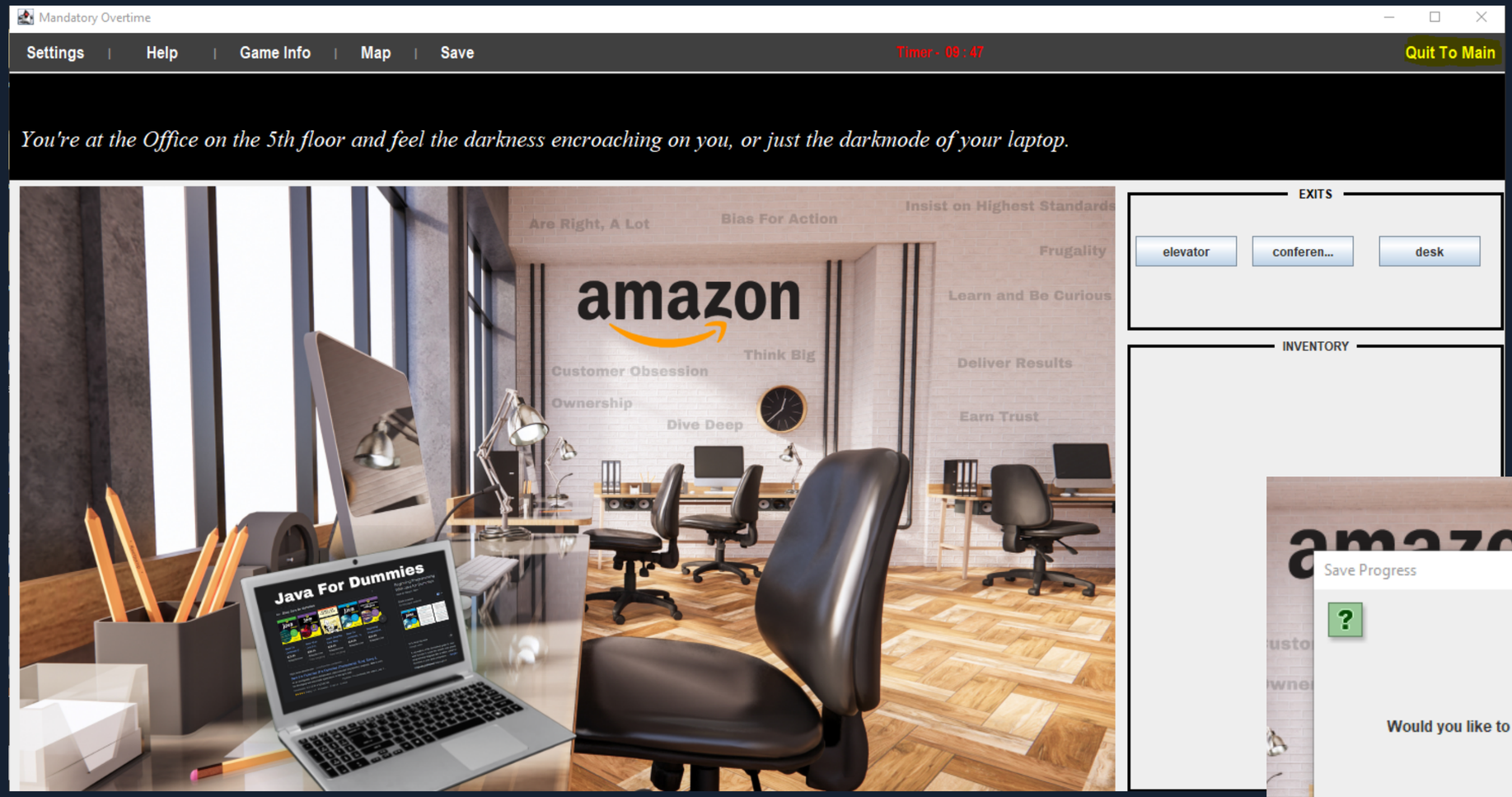
HIGHLIGHTS/FEATURES

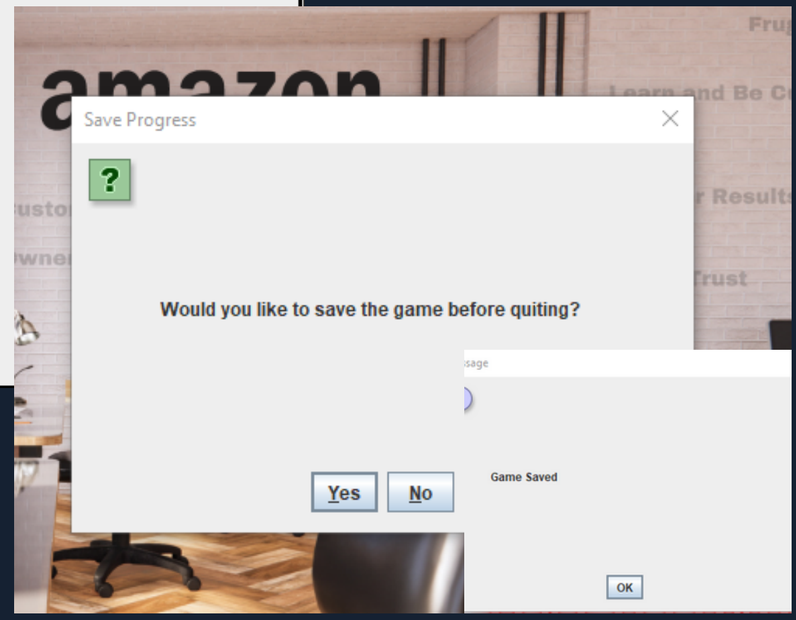




- Map
- Timer

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

GUIController.java

```
2 usages
public void setUpGamePlayHandlers() throws IOException {
    MapDialog map = new MapDialog(building.getPlayer().getCurrentLocation());

    // BUILD GAME SCREEN
    view.setUpGamePlay(building.getBuilding(), building.getGameItems());
    String currentLocationStart = building.getPlayer().getCurrentLocation();
    message = building.getBuilding().get(currentLocationStart).getDescription();

    // MOVE LOCATION
    view.setMoveConsumer(roomName -> {
        try {
            building.moveRooms2(roomName);
            String currentLocation = building.getPlayer().getCurrentLocation();
            String description = building.getBuilding().get(currentLocation).getDescription();
            String item = building.getBuilding().get(currentLocation).getItem();
            message = stringMessages.gameStatus(currentLocation, description, item);
            if(currentLocation.equals("home")){
                timer.stop();
            }
            updateGameView();
        } catch (MissingRequirementException | InterruptedException e) {
            message = e.getMessage();
            updateGameView();
        }
    });
```

```
1 usage  new *
public void setMoveListener(Consumer<String> listener) {
    moveListener = listener;
    locationGUIPanel.setMoveLocationListener(listener);
}
```

```
// DIRECTIONAL BUTTONS
if(directions != null){

    for (int i = 0; i < directions.length; i++) {
        int x = i == 0 ? 10 : i * 125;
        if (!directions[i].equals("1") &&
            !directions[i].equals("2") &&
            !directions[i].equals("3") &&
            !directions[i].equals("4") &&
            !directions[i].equals("5")) {

            JButton btn = new JButton( text: directions[i]);
            btn.addActionListener( l: e -> {
            String roomName = e.getActionCommand();
                moveListener.accept( t: roomName);
            });
            btn.setBounds( x: x,  y: 50,  width: 100,  height: 30);
            gameInfoContainer.add(btn);
        }
    }
}
```

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         );
```

```
1 usage
248 @   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
267
268         } catch (IOException e) {
269             throw new RuntimeException(e);
270         }
271         btn.setBorderPainted(false);
272         return btn;
```

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