# Software Design Description

## An Endless Night

## Authors:

*Jory Alexander, Estephanie Gonzalez, Aaron Knobloch,*

*Caleb Sears*

# 1.0 Introduction

### *1.1. Purpose*

*The purpose of this document is to outline nuances on how our game will be built. An Endless night is standalone software created for a user to interact and traverse through for the purpose of making it to the last room to beat the game. This document will give details on the UI, data storage, and subsystems.*

### *1.2. Scope*

*An Endless Night is a text-based game, the application is free to download for any operating system that has latest java jre installed. Players are able to traverse the game, fight monsters, and solve puzzles through the use of keyboard commands. Administrators have access to back-end functionality, and can change the contents and aspects of the game.*

### *1.3. Glossary*

- ***UI**: User Interface, refers to the window that the User will interact with and enter in information to.*
- ***Hero**: Is the player object within the game that will be traversing through rooms and getting health adjustments throughout.*
- ***NPC**: Non-player character*

### *1.4. References*

- *An endless night Analysis document*
- *An endless night System Decomposition document*

### *1.5. Overview of Document*

2.0. –Deployment Diagram: *This section will contain the deployment diagram showing the hardware components of the system.*

3.0. –System Design: *This section will briefly describe the Subsystems contained in our game. This section will also outline design goals for the system.*

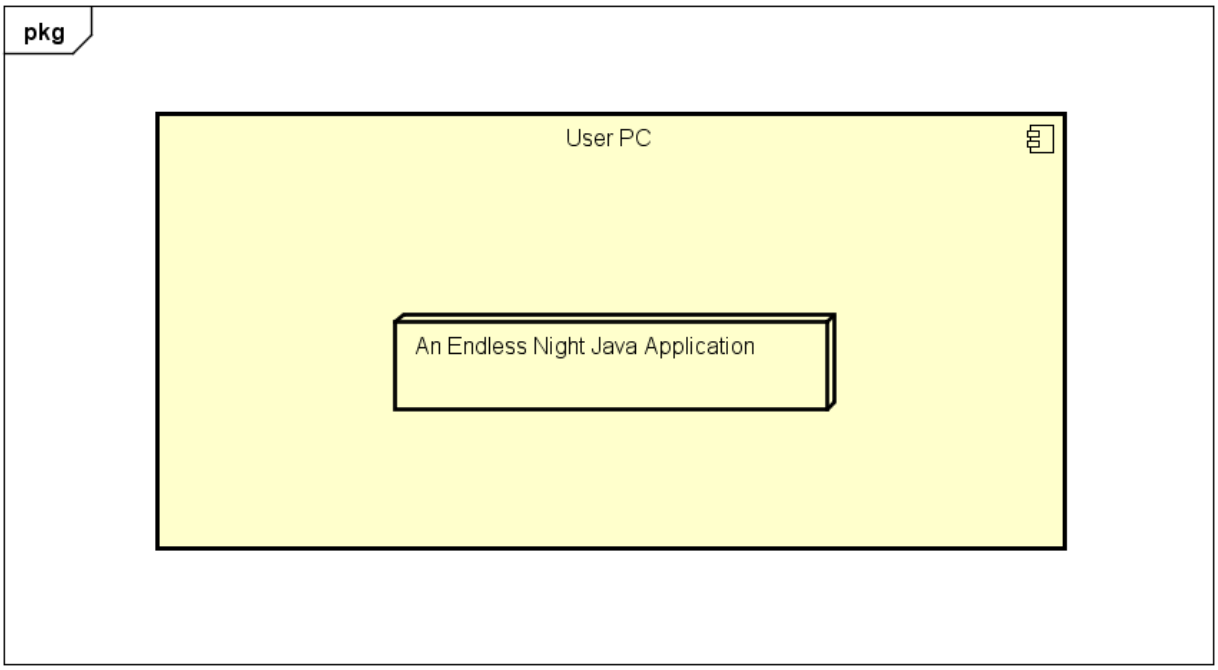
4.0. –Data Structure Design: *This section will describe in more detail the four Subsystems contained in our game.*

5.0. –Flow of control: *This section is omitted from our document; it was to contain any changes in our sequence diagrams.*

6.0. –User Interface Design: *This section will contain a description of the user interface design as it pertains to the effectiveness of our design.*

7.0. –Help System Design: *This section will describe the functions associated with the help system.*

**2.0.Deployment Diagram**




**3.0. System Design**

Subsystem Descriptions

• ***Combat Subsystem:*** *The combat System is responsible for handling the interaction between hero and monster in which both character types will be able to attack one another. Users will be able to enter commands for the combat system through the interface subsystem.*
• ***Room Management Subsystem****: The room management subsystem will be responsible for the generation of rooms.*
• ***Interface Subsystem****: The Interface subsystem will be responsible for the interpretation of user input and the interaction between subsystems. Beginning tasks including but not limited to combat, artifact manipulation, and room interaction.*
• ***Inventory Subsystem****: The inventory subsystem will be responsible for keeping track of all artifacts available within the game system.*

Design Goals

*To create a text based adventure game, where the user will be prompted to enter commands based on menus. The game will take advantage of multiple types of inheritance in order to increase usability. The game is divided into four subsystems, combat, rooms, artifact, and interface subsystems.*
*The interface subsystem will host all boundary objects as well as a game object and will be the primary intermediary between all other subsystems.*

# 4.0. Data Structure Design

## *4.1 Subsystem 1*

4.1.1 Combat Subsystem (Aaron Knobloch)-

*The combat subsystem is responsible firstly for containing the data associated with the Hero and the monsters, collectively known as characters. The combat subsystem will also be responsible for handling the interactions between the characters through combat. Combat involves calculating damage between characters by first accessing their damage, defense and health attributes and then adjusting their health accordingly.*

4.1.2 Description of Each object with attributes and methods described:

- Character:
  - CharacterID – The unique identifier of this character.
  - Health – The value associated with the current health of this character. Once this reaches zero, the game will end.
  - Strength – The value associated with the attack power of this character, or the amount that the character can reduce from the health of another character.
  - Name – The String representation of this character's identification.
  - Location – The current room ID that the character is located in.
  - DefenseValue – The value associated with the defense of this character
  - getCharacterID – returns the CharacterID
  - getHealth – returns the health of the character.
  - doDamage(int damage) – substracts the damage amount from the health.
  - Heal(int amount) – adds the amount specified to the health
  - getStrength – returns the strength of character.
  - setStrength(int amount) assigns the specified amount of strength to the character
  - getDefense – returns the amount of defense associated with this character
  - setDefense(int amount) – assigns the specified value to this characters defense
  - getName – returns the name of the character
  - getLocation – returns the ID of the room this character is located in
- Hero:
  - statusConditions – An array of the status conditions inflicting this character
  - equippedWeapon – The weapon that the hero currently has equipped
  - equippedArmor – The armor that this hero currently has equipped
  - playerInventory – an array of all the items equipped in this character's inventory
  - getEquippedWeapon – returns the weapons the hero has equipped
  - setEquippedWeapon(Weapon weapons) – equips the weapon specified
  - getEquippedArmor – returns the armor the hero has equipped
  - setEquippedArmor(Armor armor) – equips the armor specified.
  - getStatusConditions – returns an array of the status conditions afflicting this hero
  - addStatusCondition(StatusCondition) – adds this condition to the hero
  - removeStatusCondition(StatusCondition) – removes this condition from the hero
  - getPlayerInventory – returns an array of all of the items in a player inventory
  - addArtifactToInventory(Artifact item) – adds an item to inventory
- Monster:
  - ProbabilityOfAppearing – the probability that this monster has of appearing in a room

- DroppedItem – an item that this monster drops.
- getProbabilityOfAppearing – returns the ProbabilityOfAppearing
- getDroppedItem – returns DroppedItem
- setDroppedItem – cycles through possible items and sets it

### *4.2 Subsystem 2*

4.2.1 Room management Subsystem (Estephanie Gonzalez)-

*The room management system will consist of three classes, room, puzzle, and door that will handle the position and interaction of the player as they traverse through rooms and interact with objects/NPCs found in rooms. The Room management system will also display information to the user regarding artifacts, puzzles, and location. The Room management subsystem will interact with the interface subsystem to communicate commands entered when in rooms.*

4.2.2 Description of Each object with attributes and methods described:

- **Room:**
  - Room ID– The unique identifier for room.
  - Monster– The association for the type of monster available to encounter in the corresponding room.
  - Puzzle– The association for the type of Puzzle available to solve/encounter in the corresponding room.
  - Description– A string of words to describe the rooms and features to the Player.
  - Doors– An array list of doors to keep position and mapping for the Player's location/traversal.
  - Artifact– An array list of artifacts to keep inventory of current artifacts for use.
  - Visited– Boolean value to keep track of rooms visited as the Player traverses the game in order to unlock areas, reveal monsters, and display information pertaining to position.
  - Name– A String associated with the room of the current position of the Player.
  - getMonster()
  - setMonster(Monster monster)
  - getPuzzle()
  - setPuzzle(Puzzle puzzle)
  - getDescription()
  - getDoors()
  - getArtifact()
  - addArtifacts(Artifact artifact)– Allows administrator to add individual artifacts to Player's inventory
  - addAllArtifacts(ArrayList<Artifact> artifact)– Allows administrator to add all artifacts to Player's inventory
  - getVisited()
  - setVisited(Boolean visited)
  - getName()
- **Puzzle:**
  - Puzzle ID– The unique identifier for a puzzle.
  - Description– A string of words to describe the puzzle to the Player.
  - Solution– A string of words describing the solution of the puzzle.
  - isSolved– A Boolean to check if the puzzle has been solved.

- attemptsMade– A value to display the attempts made for solving a puzzle.
- Hint– A string of text to display if the Player needs a hint to solve the current puzzle.
- attemptsAllowed– A value to display the number of attempts allowed for solving a puzzle
- getDescription()
- setDecription(String description)
- getSolution()
- getIsSolved()
- setIsSolved(boolean isSolved)
- getHint()
- setHint(String hint)
- getPuzzleID()
- getAttemptsMade()
- incrementAttemptsMade()
- getAttemptsAllowed()

- **Door:**

  - Door ID– The unique identifier for a door.
  - Connected Rooms– An array list used to show the rooms near the current Player position.
  - getConnectedRooms()
  - getDoorID()

### *4.3 Subsystem 3*

4.3.1 The Interface Subsystem (Jory Alexander)-

*The Interface System will consist of a series of menus that will prompt the user to input commands. The system will then create a control object based on the entered command that will perform the intended task. The primary entity object found within the subsystem is the Game Object. The Game object will be responsible for holding all room and hero data. This is the object that will be output and read during a save and load respectively. The inter face subsystem will interact with the Inventory, Combat, and Room subsystems.*

4.3.2 Description of Each object with attributes and methods described

- **Game:**

  - Rooms – An array list of all rooms within the game.
  - Hero – The hero of the game contains information regarding the player's health strength armor and inventory.
  - getRooms() - Returns an array list of rooms
  - getHero() - Returns the Games Hero Object

### *4.4 Subsystem 4*

4.4.1 The Inventory Subsystem (Caleb Sears)-

*The Inventory Subsystem will consist of five classes, Artifact, Consumable, Weapon, Armor, and KeyArtifact. The Consumable class contains the information of items that can only be used once, and communicates with the Artifact class. The Weapon class contains the information of items that can cause damage to monsters and also communicates with the Artifact class. The Armor class contains the information of items that can decrease the amount of damage taken by the user. It also communicates with the Artifact class. The KeyArtifacts class contains the information of items that*

*are imperative to the progression of the game and communicates with the Artifact class as well. The Artifact Class contains the information that it's subclasses would share. It also communicates with the Interface subsystem to allow sharing of item information to other subsystems.*

4.4.2 Description of each object with attributes and methods described:

- **Artifact:**
  - Name – A String that contains the name of an artifact.
  - Description – A String that contains the description of the artifact.
  - ArtifactID – An int that contains a unique identifier of the artifact.
  - getName()
  - getDescription()
  - getArtifactID()
- **Consumable:**
  - HealAmount – An int for the amount of health given by a consumable
  - getHealAmount()
- **Armor:**
  - Defense – An int for the amount of damage an armor prevents
  - getDefense()
- **Weapon:**
  - Strength – An int for the amount of damage a weapon can cause
  - getStrength()
- **KeyArtifact:**
  - getPuzzleID()
  - isKey() –A boolean to compare keys
  - IsKey – a Boolean to check if the key is correct
  - PuzzleID – An int to uniquely identify a puzzle

## 5.0 Flow of Control

*n/a*

## 6.0 User Interface Design

*The user will be presented with a variety of input options on the console. The user will enter their command into the console and new information will be displayed based on the command. The UI will be uncluttered and simple for ease-of-use.*

## 7.0 Help System Design

*At any time the user will be able to enter in the word help. The system will recognize the word regardless of case. The user will be given a list of general commands, relative to the context of their current menu, followed by a brief description of each command.*