



# **TEXT BASED ADVENTURE GAME**



## **A PROJECT REPORT**

*Submitted by*

**ARAVINDAN K - (2303811710421008)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 - JAVA PROGRAMMING**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

**DECEMBER- 2024**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on **“TEXT BASED ADVENTURE GAME”** is the bonafide work of **ARAVINDAN K (2303811710421008)** who carried out the project work during the academic year 2024 - 2025 under my supervision

CGB1201-JAVA PROGRAMMING  
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,  
HEAD OF THE DEPARTMENT  
PROFESSOR

**SIGNATURE**

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING  
Mr. M. SARAVANAN, M.E.,  
SUPERVISOR  
ASSISTANT PROFESSOR

**SIGNATURE**

Mr. M. Saravanan, M.E.,

**SUPERVISOR**

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 02.12.2024

CGB1201-JAVA PROGRAMMING  
Mr. M. ARMANAN A, M.E.,  
INTERNAL EXAMINER  
ASSISTANT PROFESSOR

**INTERNAL EXAMINER**

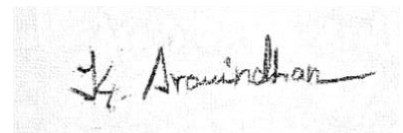
CGB1201-JAVA PROGRAMMING  
Dr. R. SETHAMILSELVI, M.E., Ph.D.,  
EXTERNAL EXAMINER  
PROFESSOR  
8138-SCE, TRICHY.

**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**TEXT BASED ADVENTURE GAME**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB120- JAVA PROGRAMMING**.

**Signature**

A handwritten signature in black ink, appearing to read 'Aravindhan', is written over a light gray rectangular background.

ARAVINDAN K

Place: Samayapuram

Date: 02.12.2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. M. SARAVANAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards

## **MISSION OF THE INSTITUTION**

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

## **VISION OF DEPARTMENT**

To be a center of eminence in creating competent software professionals with research and innovative skills.

## **MISSION OF DEPARTMENT**

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## **PROGRAM EDUCATIONAL OBJECTIVES**

### **1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

### **2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

### **3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

### **PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

### **PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

### **PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

## **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **ABSTRACT**

A Text-Based Adventure Game is a captivating form of interactive storytelling that immerses players in a richly crafted fictional world where their choices and actions shape the narrative. Unlike traditional video games, this genre relies entirely on textual descriptions and player input, creating a unique and imaginative experience. Players navigate the game by typing commands to explore various locations, interact with objects, and engage in conversations with characters. Each decision influences the unfolding story, presenting players with branching paths, challenging puzzles, and moral dilemmas. The game's success lies in its ability to paint vivid scenarios and evoke emotions through descriptive writing, drawing players into a dynamic and ever-evolving narrative. As they progress, players must use their problem-solving skills, creativity, and critical thinking to overcome obstacles and achieve their objectives. This genre provides a deeply personal and engaging adventure, where the player's imagination becomes the driving force, transforming simple text into a vibrant and immersive world.



## ABSTRACT WITH POs AND PSOs MAPPING

### CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>A Text-Based Adventure Game is a captivating form of interactive storytelling that immerses players in a richly crafted fictional world where their choices and actions shape the narrative. Unlike traditional video games, this genre relies entirely on textual descriptions and player input, creating a unique and imaginative experience. Players navigate the game by typing commands to explore various locations, interact with objects, and engage in conversations with characters. Each decision influences the unfolding story, presenting players with branching paths, challenging puzzles, and moral dilemmas. The game's success lies in its ability to paint vivid scenarios and evoke emotions through descriptive writing, drawing players into a dynamic and ever-evolving narrative. As they progress, players must use their problem-solving skills, creativity, and critical thinking to overcome obstacles and achieve their objectives. This genre provides a deeply personal and engaging adventure, where the player's imagination becomes the driving force, transforming simple text into a vibrant and immersive world.</p>	<p><b>PO1 -3</b>  <b>PO2 -3</b>  <b>PO3 -3</b>  <b>PO4 -3</b>  <b>PO5 -3</b>  <b>PO6 -3</b>  <b>PO7 -3</b>  <b>PO8 -3</b>  <b>PO9 -3</b>  <b>PO10 -3</b>  <b>PO11-3</b>  <b>PO12 -3</b></p>	<p><b>PSO1 -3</b>  <b>PSO2 -3</b>  <b>PSO3 -3</b></p>

Note: 1- Low, 2-Medium, 3- High

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	viii
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
<b>2</b>	<b>PROJECT METHODOLOGY</b>	2
	2.1 Proposed Work	2
	2.2 Block Diagram	2
<b>3</b>	<b>MODULE DESCRIPTION</b>	3
	3.1 User Authentication and Management Module	3
	3.2 Book Listing and Display Module	3
	3.3 Book Purchase Module	3
	3.4 Book Selling Module	4
	3.5 Main Menu and Navigation Module	4
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	5
	4.1 Conclusion	5
	4.2 Future Scope	5
	<b>REFERENCES</b>	6
	<b>APPENDIX A (SOURCE CODE)</b>	7
	<b>APPENDIX B (SCREENSHOTS)</b>	13

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Objective**

The objective of this project is to develop an interactive Java-based adventure game with a user-friendly graphical interface using Swing. Players navigate through a haunted house by exploring rooms, collecting items, and defeating monsters to progress. The game incorporates an inventory system to manage collected items and uses object serialization to enable save and load functionality. It aims to provide an engaging experience through dynamic room interactions and challenges. The project demonstrates the practical application of object-oriented programming principles, event-driven GUI design, and file handling techniques. It is designed as an educational tool for learning Java while creating a fun, immersive gaming experience.

### **1.2 Overview**

The Haunted House Adventure Game is a Java-based interactive application that blends text-based storytelling with a graphical user interface. Players explore interconnected rooms within a haunted house, collect items, and face challenges like locked doors and hostile monsters. The game utilizes object-oriented design with classes for rooms, items, and monsters, offering modular and extensible gameplay. A save-and-load feature allows players to preserve and resume their progress. The game emphasizes player interaction through GUI components like directional buttons and action controls, making it an engaging and educational project for Java learners.

### **1.3 Java Programming Concepts**

The game demonstrates Object-Oriented Programming (OOP) with classes like Rooms and Monster for encapsulated game entities. Swing is used for GUI development, with event handling through action listeners for user interactions. The Collections Framework manages room connections (hashmaps) and inventory (hashset). File I/O with serialization enables saving and loading game state. Control flow with conditionals and loops handles navigation, item collection, and combat logic, providing a dynamic and interactive experience.

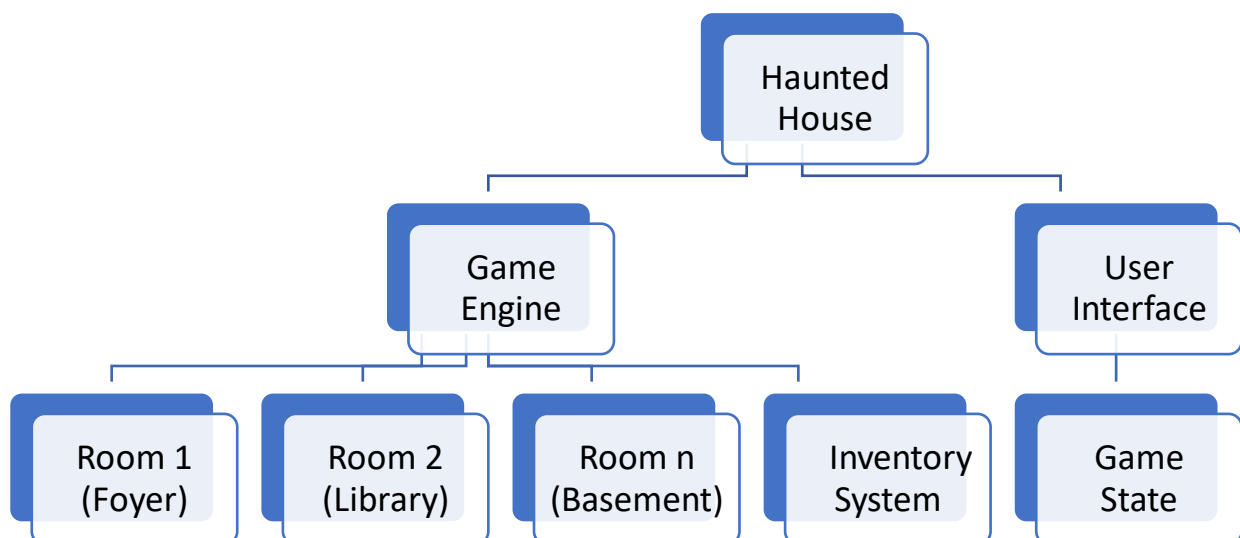
## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

The proposed work focuses on designing and developing an engaging Java-based game that integrates core gameplay mechanics with modular code architecture. Key tasks include implementing features like room navigation, item collection, combat, and puzzle-solving to create an interactive experience. An intuitive graphical user interface will be developed using the Swing library, allowing seamless user interactions. Save-and-load functionality will be incorporated using object serialization to ensure data persistence. Dynamic interactions between the player, items, and monsters will be implemented, with logical constraints such as locked doors requiring specific keys. The project will prioritize code modularity and reusability through object-oriented programming principles while ensuring logical accuracy and performance through rigorous testing and debugging.

#### 2.2 Block Diagram



## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 Room Module:**

The Room class defines the structure and behavior of game locations. Each room contains attributes such as its name, description, connected exits, items, and an optional monster. It includes methods to set and retrieve exits, manage items, and handle monster interactions.

#### **3.2 Monster Module:**

The Monster class models the game's enemies, storing details like the monster's name and description. This module integrates with the combat system, allowing players to engage and defeat monsters when equipped with appropriate items.

#### **3.3 Player Interaction Module::**

Manages the player's inventory and actions, including collecting items, fighting monsters, and navigating between rooms. It validates player inputs and ensures interactions with the game environment adhere to defined rules, such as requiring a key to unlock certain rooms.

#### **3.4 GUI Module:**

The graphical interface is built using Swing components. It includes directional buttons for navigation, action buttons (e.g., Take, Fight, Save, Load), and a text area for displaying room descriptions, inventory, and game messages. This module enhances user interaction and immersion.

### **3.5 Persistence Module:**

Handles saving and loading the game state using Java's serialization. This module stores the current room and the player's inventory in a file, allowing the game to be resumed from the last save point.

## **CHAPTER 4**

### **CONCLUSION & FUTURE SCOPE**

#### **4.1 CONCLUSION**

The Haunted House Adventure Game successfully demonstrates the application of Java programming concepts to create an engaging and interactive text-based adventure. By leveraging Object-Oriented Programming principles, the game achieves modularity and scalability with well-defined classes for rooms, monsters, and game logic. The graphical interface, built with the Swing library, enhances user interaction, making the game intuitive and visually engaging. Features such as inventory management, combat mechanics, and data persistence via save/load functionality add depth to the gameplay experience. This project serves as a practical implementation of Java skills, combining GUI design, event handling, and file I/O, while offering an enjoyable and immersive gaming experience.

#### **4.2 FUTURE SCOPE**

The Haunted House Adventure Game has great potential for expansion. Future developments could include more complex gameplay with additional rooms, puzzles, and diverse monsters. The combat system could be enhanced with turn-based mechanics, new weapons, and a health system. Multiplayer functionality would allow cooperative play, while a fully graphical interface would improve visual appeal with animations and character sprites. Adding Artificial Intelligence for smarter monsters and procedural room generation would create dynamic gameplay. The game could also be adapted for mobile platforms, expanding its reach. Sound effects and music would enhance immersion, creating a more atmospheric experience. These updates would make the game more engaging and enjoyable.



## REFERENCE

### Java Books:

#### 1. "The Complete Reference" by Schildt.H

This book is an excellent resource for mastering Java programming, covering both fundamental and advanced Java topics, including GUI development with Swing, which is used for the game's user interface.

### Websites:

#### GeeksforGeeks - Java Game Development

URL: <https://www.geeksforgeeks.org/java-game-development/>

A beginner-friendly guide to game development in Java, providing tutorials and examples for creating interactive games.

#### Oracle Java Documentation

URL: <https://docs.oracle.com/javase/tutorial/>

Official Java tutorials, covering core concepts, Swing components, and event handling, which are essential for creating graphical interfaces in Java.

### YouTube Links:

Java Game Development Tutorials - Programming with Mosh

URL: <https://www.youtube.com/watch?v=GGJ47bhdFuw>

This tutorial introduces Java game development concepts and guides you through building simple games using Java.

## **APPENDIX A**

### **(SOURCE CODE)**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.HashMap;
import java.util.HashSet;

public class HauntedHouseGame extends JFrame {
    private JTextArea output;
    private JButton saveButton, loadButton;
    private JButton northButton, southButton, eastButton, westButton, takeButton, fightButton;

    private HashMap<String, Room> rooms = new HashMap<>();
    private HashSet<String> inventory = new HashSet<>();
    private Room currentRoom;

    public HauntedHouseGame() {
        setupGUI();
        setupGame();
        showCurrentRoom();
    }

    private void setupGUI() {
        setTitle("Haunted House Adventure");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        output = new JTextArea(10, 30);
        output.setEditable(false);
        output.setLineWrap(true);
        JScrollPane scrollPane = new JScrollPane(output);
```

```

// Save and Load buttons
saveButton = new JButton("Save");
saveButton.addActionListener(new SaveHandler());

loadButton = new JButton("Load");
loadButton.addActionListener(new LoadHandler());

// Directional movement buttons
northButton = new JButton("North");
northButton.addActionListener(e -> move("north"));

southButton = new JButton("South");
southButton.addActionListener(e -> move("south"));

eastButton = new JButton("East");
eastButton.addActionListener(e -> move("east"));

westButton = new JButton("West");
westButton.addActionListener(e -> move("west"));

// Take button to pick up items
takeButton = new JButton("Take");
takeButton.addActionListener(e -> takeItem());

// Fight button for combat
fightButton = new JButton("Fight");
fightButton.addActionListener(e -> fightMonster());

// Layout the components
JPanel mainPanel = new JPanel(new BorderLayout());
mainPanel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new GridLayout(2, 4, 5, 5));
buttonPanel.add(saveButton);
buttonPanel.add(northButton);
buttonPanel.add(loadButton);
buttonPanel.add(fightButton);

```

```

buttonPanel.add(westButton);
buttonPanel.add(southButton);
buttonPanel.add(eastButton);
buttonPanel.add(takeButton);

add(mainPanel, BorderLayout.CENTER);
add(buttonPanel, BorderLayout.SOUTH);

setVisible(true);
}

private void setupGame() {
    // Define rooms
    Room foyer = new Room("Foyer", "You are in the foyer. A grand chandelier hangs above.");
    Room kitchen = new Room("Kitchen", "You see a dusty kitchen with old pots and pans.");
    Room library = new Room("Library", "Shelves of old books surround you. There's a strange
silence.");
    Room mysteryRoom = new Room("Mystery Room", "This room has an eerie feeling and a locked
door with strange symbols.");
    Room basement = new Room("Basement", "A dark, damp room with chains hanging from the
walls.");

    // Link rooms
    foyer.setExit("north", library);
    foyer.setExit("east", kitchen);
    kitchen.setExit("west", foyer);
    library.setExit("south", foyer);
    library.setExit("east", mysteryRoom);

    mysteryRoom.setExit("down", basement);

    // Add items (weapons, keys, spells)
    kitchen.setItem("key");
    library.setItem("sword");
    basement.setItem("spell");

    // Add monsters

```

```

mysteryRoom.setMonster(new Monster("Ghost", "A terrifying ghost that haunts the room.));

rooms.put("foyer", foyer);
rooms.put("kitchen", kitchen);
rooms.put("library", library);
rooms.put("mysteryRoom", mysteryRoom);
rooms.put("basement", basement);
currentRoom = foyer;
}
private void showCurrentRoom() {
    output.setText(currentRoom.getDescription() + "\nExits: " + currentRoom.getExits() + "\n");

    if (currentRoom.getItem() != null) {
        output.append("You see a " + currentRoom.getItem() + " here.\n");
    }
    if (currentRoom.getMonster() != null) {
        output.append("Beware! " + currentRoom.getMonster().getDescription() + "\n");
    }
    output.append("Inventory: " + inventory + "\n");
}
private void move(String direction) {
    Room nextRoom = currentRoom.getExit(direction);
    if (nextRoom == null) {
        output.append("You can't go that way!\n");
    } else if (nextRoom == rooms.get("mysteryRoom") && !inventory.contains("key")) {
        output.append("The door is locked. You need a key.\n");
    } else {
        currentRoom = nextRoom;
        showCurrentRoom();
    }
}
private void takeItem() {
    if (currentRoom.getItem() != null) {
        inventory.add(currentRoom.getItem());
        output.append("You picked up: " + currentRoom.getItem() + "\n");
        currentRoom.setItem(null);
    } else {

```

```

        output.append("There's nothing to take here.\n");
    }
}

private void fightMonster() {
    Monster monster = currentRoom.getMonster();
    if (monster == null) {
        output.append("There is no monster here to fight.\n");
        return;
    }
    if (inventory.contains("sword") || inventory.contains("spell")) {
        output.append("You defeated the " + monster.getName() + " with your weapon!\n");
        currentRoom.setMonster(null);
    } else {
        output.append("You have no weapon or spell to fight the " + monster.getName() + "!\n");
    }
}

private class SaveHandler implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("savegame.dat")))
        {
            out.writeObject(currentRoom.getName());
            out.writeObject(inventory);
            output.append("Game saved.\n");
        } catch (IOException ex) {
            output.append("Error saving game.\n");
        }
    }
}

private class LoadHandler implements ActionListener {
    @SuppressWarnings("unchecked")
    @Override
    public void actionPerformed(ActionEvent e) {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("savegame.dat"))) {
            String roomName = (String) in.readObject();

```

```

        inventory = (HashSet<String>) in.readObject();
        currentRoom = rooms.get(roomName);
        showCurrentRoom();
        output.append("Game loaded.\n");
    } catch (IOException | ClassNotFoundException ex) {
        output.append("Error loading game.\n");
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(HauntedHouseGame::new);
}

private static class Room implements Serializable {
    private String name;
    private String description;
    private HashMap<String, Room> exits = new HashMap<>();
    private String item;
    private Monster monster;

    public Room(String name, String description) {
        this.name = name;
        this.description = description;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public void setExit(String direction, Room room) {
        exits.put(direction, room);
    }

    public Room getExit(String direction) {
        return exits.get(direction);
    }
}

```

```

    }
    public String getExits() {
        return String.join(" ", exits.keySet());
    }
    public String getItem() {
        return item;
    }
    public void setItem(String item) {
        this.item = item;
    }
    public Monster getMonster() {
        return monster;
    }
    public void setMonster(Monster monster) {
        this.monster = monster;
    }
}

private static class Monster implements Serializable {
    private String name;
    private String description;

    public Monster(String name, String description) {
        this.name = name;
        this.description = description;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }
}
}

```



## APPENDIX B

### (SCREENSHOTS)

