Assignment II: Dungeon Crawler (Linked Lists)
CSCI-UA 9102 – Introduction to Data Structures
Total: 30pts

Augustin Cosse

June 18, 2026

Overview

In this assignment, you will implement a small dungeon crawler game using a custom linked list.

The dungeon is made up of rooms connected sequentially:

```
Entrance -> Room -> Room -> Boss
```

Each room is stored as a node in a linked list.
The player moves forward through the dungeon and interacts with rooms.

Content

The Assignment contains 7 files that should be organized as follows :

- One main folder dungeon that includes two subfolders model and view as well as the file MainApp.java
- The subfolder model should contain the files DungeonGenerator.java, DungeonList.java, DungeonPlayer.java, Room.java as well as RoomNode.java
- The subfolder view should contain the file GameView.java

Learning Objectives

- Implement a custom linked list
- Understand node-based data structures
- Practice traversal, insertion, deletion
- Apply object-oriented programming
- Build a simple GUI using Java Swing

Part 1: Room Representation (3pts)

Task 1.1

Design the `Room` class (the class should belong to the package located in dungeon/model) with:

- String type variable name
- RoomType variable type. RoomType should be declared as an enum type with the following possible fixed set of constants: monster, treasure, trap, boss, empty.
- String type variable description

The class should contain a constructor that instantiates the variables name, type and description. I should also contain accessor methods getName(), getType() and getDescription().

Part 2: Linked List Node (4pts)

Task 2.1

Create a `RoomNode` class (the class should belong to the package located in dungeon/model and for this exercise we will consider a public Node class) containing:

- a Room object
- a reference to the next RoomNode
- A constructor that sets the value of the Room object.

Part 3: File DungeonList.java (5pts)

The file should belong to the package located in dungeon/model.

Task 3.1

Implement addRoom(Room room) to append a node at the end of the list.

Task 3.2

Implement size() to count nodes.

Task 3.3

Implement getRoom(int index).

Task 3.4

Implement removeRoom(int index).

Part 4: File DungeonGenerator.java (5pts)

Generate a dungeon with:

- 1 entrance room
- 8–12 random rooms (i.e. empty, monster, treasure or trap) (you should rely on java.util.Random)
- 1 boss room

Part 5: File DungeonPlayer.java (5pts)

Implement a player that:

- An instance of the class Player should store a variable health (initialized with a value of 100) and a variable gold initialized with a value of 0. It should also keep track of the current room through a variable current of type RoomNode (indicating the current room of the player).
- It should have a method moveForward that moves the player to the next room.
- Moreover, it should apply the effects of each room to the state of the player as follows:
  - If current. room.getType() is of type MONSTER, the health variable of the player should be decreased by 20
  - If current. room.getType() is of type TREASURE, the gold variable of the player should be increased by 50
  - If current. room.getType() is of type BOSS, the health variable of the player should be decreased by 30
- Finally an instance of the class Player should have the following accessor methods: getRoom, getHealth, getGold as well as an isDead method that should return a boolean value indicating when the player is dead.

Part 6: File GameView.java (5pts)

You will now complete the file GameView.java to generate the graphical user interface. The GameView class contains three attributes of type Jlabel and one attribute of type JButton.

- Start by instantiate those attributes (usual use of the constructor with an input given simply by the string that should appear on each label/button).
- You can then use the lines roomLabel.setAlignmentX(Component.LEFT_ALIGNMENT); as well as add(Box.createVerticalStrut(10)); add(roomLabel); to position the buttons and labels (repeat the same lines with each of the label and button components)

Final MainApp.java file (3pts)

In order to run the game, we are left with completing the main file. Within the main method, implement the following steps:

- Generate a dungeon
- Create a Player (setting as his initial room the head node from the Dungeon)
- Create an object of type GameView
- An initial call to the update() method with inputs view and Player
- The line view.moveBtn.addActionListener(e− > {) is the line that defines what happens when the button is clicked. Complete the brackets with the following elements:
  - A call to the moveForward() method of the player (The button should update the player's position)

  - A call to the applyRoomEffect() of the player
  - A call to the update() method with inputs view and Player

- Complete the update method.
  - Start by creating an instance of the Room class and initialize this room from the method getRoom() of the player.
  - Use the methods setText() from the roomLabel and descLabel attributes of the GameView object to set the labels that will appear on those objects. Use getName and getDescription attributes of the room Object to retrieve the name and description of the current room.