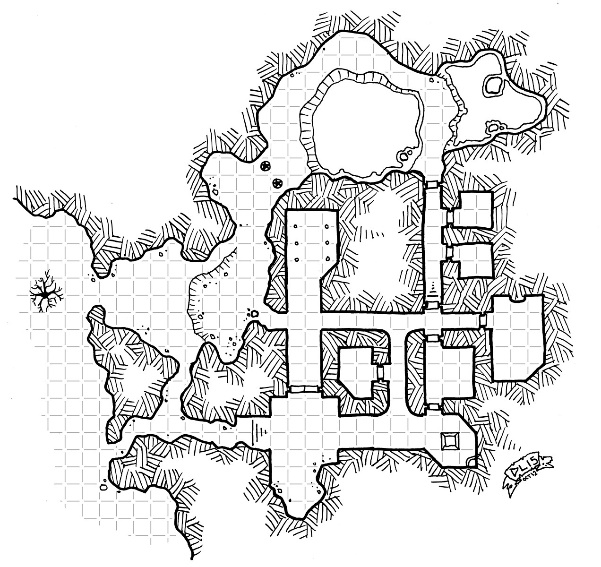
Assessment 2: Project (Escape from Javalice)



***Escape from Javalice***

Introduction

**Generative AI tools cannot be used in this assessment task:** In this assessment, you must not use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task.

This document specifies the programming component of the Assignment which is an individual assessment task and must be your own work. You must attribute the source of any part of your code that you have not written yourself. Please note the section on plagiarism in this document. *In this assessment, you must* ***not*** *use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task.*

**The assignment must be done using the workspace environment on the Ed platform.**

The Java source code for this assignment must be implemented according to the **Java Coding Standards for this unit.**

The game will only use a text-based interface and no graphics are required.

Any points needing clarification may be discussed with your teaching staff.

Completion of this assignment contributes towards the following learning outcomes:

1. *design, construct, test, and document small computer programs using Java;*
2. *interpret and demonstrate software engineering principles of maintainability, readability, and modularisation;*
3. *explain and apply the concepts of the "object-oriented" style of programming.*

Specification

In the game, *Escape from Javalice*, the player is a wielder of magical powers. The new king of the land has made the practice of magic illegal

. As such, the player must try to escape Javalice as soon as possible by opening a magical doorway (i.e. a portal) to another realm. To prevent this, the king has created a series of mazes which must be navigated first. The aim of *Escape from Javalice* is for a player to choose the correct magical portal which will lead them to the final exit so they can leave Javalice or land up in jail.

The magic wielders of Javalice have also developed a ‘Get out of Jail’ portal. It allows them to travel back to their previous location, so they can try an alternative exit. However, the ‘Get out of Jail’ portal can only be used a maximum of 3 times. When travelling back to their previous location, they must land in the exact same portal location with the same exits available to them.

The king’s magic police are triggered when a magic wielder jumps through a portal, and chase the magic wielder through the maze. The magic wielder must avoid the magic police at all cost. The longer they keep using the portals, the higher chance the magic police will find them. If they land in the same room using the portals, the magic wielder can only jump backward to escape them. If that is not possible, they get caught. If they use the portal and there is no exit and they do not have any ‘Get out of Jail’ backward jumps left, once again, they lose the game.

Game play

The *Escape from Javalice* game gameplay can be described as follows:

1. When the game begins the system will read the exit details from the file called ‘**exits.txt**’. This file can be found on Ed and must not be modified. The format for the file is as follow:
   1. Portal direction – North, South, East, West
   2. Probability of portal being open – percentage chance the portal is open or not
   3. Probability of finding exit – percentage chance the exit portal can be found
   4. Probability of magic police encounter – percentage chance of encountering the magical police in this direction
2. The game can only read the file ONCE at the start of the game.
3. Once the file is read, all the exits must be configured within the game with the associated probabilities.
4. Then the game begins with a simple screen which prompts the user to enter their name, which must be between 3 and 12 characters long (both inclusive). If this criterion is not met, the player is asked to re-enter their name until correct.
5. The player has 10 coins at the start of the game.
6. Next the player is placed in the middle of a room. The rules for the room are as follows:
   1. Each room may have up to 4 portals possible which allows the player to go from one realm to another – to the north, to the south, to the east, and to the west.
   2. The rules for the portal are as follows:
      1. The probability for each portal occurring is read from the file.
      2. Once the player selects a portal direction, if the exit is not found, then the following probabilities are calculated for the selected portal direction only:
         1. The probability of finding the exit portal is randomly either increased or decreased by 1% and 5%. The probability range should always be between 0% and 100%.
         2. The probability of magic police encounter is randomly either increased or decreased between 1% and 5%. The probability range should always be between 0% and 100%.
      3. The above details are then updated and saved for the next iteration when the player has to choose again.
   3. The system will randomly generate which portals are available to the user and ask the player to make a choice. For each direction, the system will also display the current probability of finding the exit portal as well as the probability of encountering the magic police.
   4. The players’ main goal is to find the final escape portal while avoiding the magic police.
      1. The final escape portal is an independent probability calculation based on the value read from the file.
   5. The player will then make a choice to go in one available direction.
   6. When the player moves to the next room, the system will generate a random probability if the magic police is present in that room or not based on the probability read from the file.
   7. Also upon entering a room, the player has a 50% probability of finding a magic box which can contain one of the following three items:
      1. Coins – Random between 10 and 35. 30% probability of occurring.
      2. Magic police alarm – Raises ‘Probability of magic police encounter’ in ALL directions by 3%. 25% probability of occurring.
      3. Invisibility Cloak – Allows player to escape from a room with the magic police present. 15% probability of occurring.
      4. Coal – Does nothing. 30% probability of occurring.
   8. If the player finds a magic box:
      1. The player can either choose to open it or to not open it.
      2. If the player opens it, they can choose to add the item found to their inventory. Coins, police alarms, and coal are not part of inventory.
      3. The player can only have a maximum of 3 items in their inventory.
   9. If the player lands in jail:
      1. The player has the option to jump backward.
      2. The probabilities are not reversed when the player jumps back. As such, when the player jumps backward a new set of exits are generated and they can encounter the magical police there the second time.
      3. The player can only jump back a maximum of three times. If there are no more jumps available, the game ends.
   10. If the player lands in a room with the magical police:
       1. The player may choose to use an ‘Invisibility Cloak’, if they have one, and then continue to the next room.
       2. If no invisibility cloaks are available, the player may choose to “Bribe” the police using coins. The amount requested is random between 0.5x – 1.5x (x = number of coins the player has). If the player cannot pay, the player is caught.
   11. The game will continue until the player ends up in jail with no backward jumps available, is caught by the magic police, or manages to find the final exit portal.
   12. The game will conclude by writing the outcome to a file called ‘Outcome.txt’.

Game Requirements

* You must use the workspace environment in the Ed platform to code all parts of your program. You must not copy paste huge chunks of code from somewhere else.
* You must ensure that your program code meets the coding standard requirements outlined in the course.
* You must ensure to use appropriate collections within your program and be able to justify your choice.
* You must appropriate good design within your program by correctly applying abstraction and modularization techniques.
* Your program must only read and write to the file one time (at the start of the program to read and at the end of the program to write)
* Your program must ensure to use exception handling correctly. Your program must not crash no matter what the user enters.
* The main class in your program **MUST** be called **Game.java** and it should contain the **main()** method to start the program.

Hints and Suggestions

* Remember to modularize your assignment. Coding without breaking down the ‘how’ will get you nowhere very quickly. If unsure how to modularize, discuss this with your teaching staff
* Get your program working for a single action first and then add other actions once your code is working for the single action.
* Save the finding items for the end. It might prove a bit of a challenge so get the rest of your assignment working without it and then include it in later.
* Remember when testing your programs, to display probabilities on the screen to make the gameplay more understandable.

Video demonstration

You will be required to create a video (maximum of 10 minutes) in which you demonstrate all the requirements/functionalities outlined within the assessment instructions. Your video should be published as unlisted on YouTube. You will then need to create a file within your project workspace and include the link to your video in this.

Your video will need to include a screencast of your program, a voice-over in which you describe the functionality and decisions that you have made and you will need to include webcam footage of yourself as you demonstrate this (a small frame within the screencast is suitable).  
  
**The video should explain your code/design and functionality using a class diagram(You can use any tool to draw the class diagram). The video will focus on your understanding of the code, not the actual code itself.**

[Record video with Panopto](https://guides.lib.monash.edu/learning-tools/video-assignments#s-lib-ctab-18455526-3) (Monash 2021) is a good guide for using Panopto to record your screencast, although you will need to download your video and upload it to YouTube. [OBS Studio](https://obsproject.com/) (OBS studio 2021) is also a great tool for this task.

Instructions on uploading videos to YouTube and ensuring that they are unlisted can be found in [Upload videos](https://support.google.com/youtube/answer/57407?hl=en-GB&co=GENIE.Platform%3DDesktop&oco=0#zippy=%2Cdetails%2Cvisibility) (Google 2021) and [Change video privacy settings](https://support.google.com/youtube/answer/157177) (Google 2021).

Assessment

The following outline the marking criteria for this assignment:

* 5% - Coding Standards
* 25% - Design
* 40% - Functionality
* 10% - Test Strategy for the Player class.
* 20% - Video(10 Minutes): Explanation on Code and Design

Questions and Clarifications

This assignment has an open design that can be adopted by students to achieve the outcome. However, should there be questions or aspects of the assignment which require further clarification, please use the discussion forums to seek help from your teaching staff. You are allowed to add any additional aspects to the assignment to make it more engaging provided you do not miss out on any of the requirements outlined within this specification document.

Plagiarism

Cheating and plagiarism are viewed as serious offences. All submitted code will be subjected to a similarity checker and any submissions determined to be similar to another current or past submission will be investigated further. The outcome of the decision pertaining to plagiarism and/or collusion will be determined by the faculty administration. To ensure compliance with this requirement, ensure to do all your code in the Ed workspace environment itself and do not copy-paste any code into the workspace environment.