Project Overview: The A-Maze-Ing Labyrinth Game

1. Introduction:

We are recreating a board game called “The aMAZEing Labyrinth”
There is one game board with 16 fixed path tiles, 34 path tiles that will be placed on the board and can be moved, 24 treasure cards that will be divided amongst the players, and they would have to find it on the board, and 4 playing pieces representing the players playing.

In this enchanted labyrinth, players set out to search for mysterious objects and creatures. By cleverly sliding the paths players try to find their way to the coveted treasure. The first player to find all their treasures and return to the starting square is the winner.

2. Objectives:

-Emulate the game's rules, mechanics, and interactions.
-Create a graphical user interface (GUI) for a user-friendly experience.
-Implement features like player turns, movement, treasure collection, and game state management.

3. Classes and Objects:

Board: Represents the game board with a dynamic maze.
Player: Represents each player with attributes like position, collected treasures, etc.
Cards (Abstract): It is used to manipulate the cards of the game.
Maze Cards: Represent the moveable cards in the maze.
Fixed Maze Cards: Represent the immovable cards in the maze.
Treasure Cards: Represent the treasures that each player has to find.
Coordinates: Used to set and retrieve coordinates and draw objects.
Game: Represent the players, their turns, and statistics.
Stack: Store Treasure Cards in the stack.
Queue: Store player details in the queue.

4. Game Mechanics:

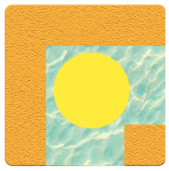
Players take turns navigating the labyrinth to find treasures.
The labyrinth consists of movable tiles that players can shift to create paths.
Players collect treasures by reaching them through connected paths.
The first player to collect a certain number of treasures wins.

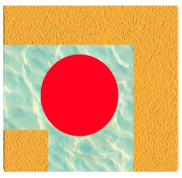
5. User Interface:

-Develop a GUI to visualize the game board, players, and treasures.
-Include interactive elements for players to navigate the maze and make moves.

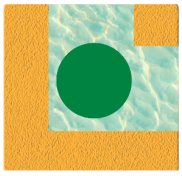
-Display relevant information like player turns, collected treasures, and game status.

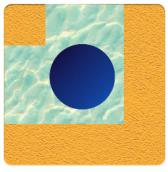
Screen:

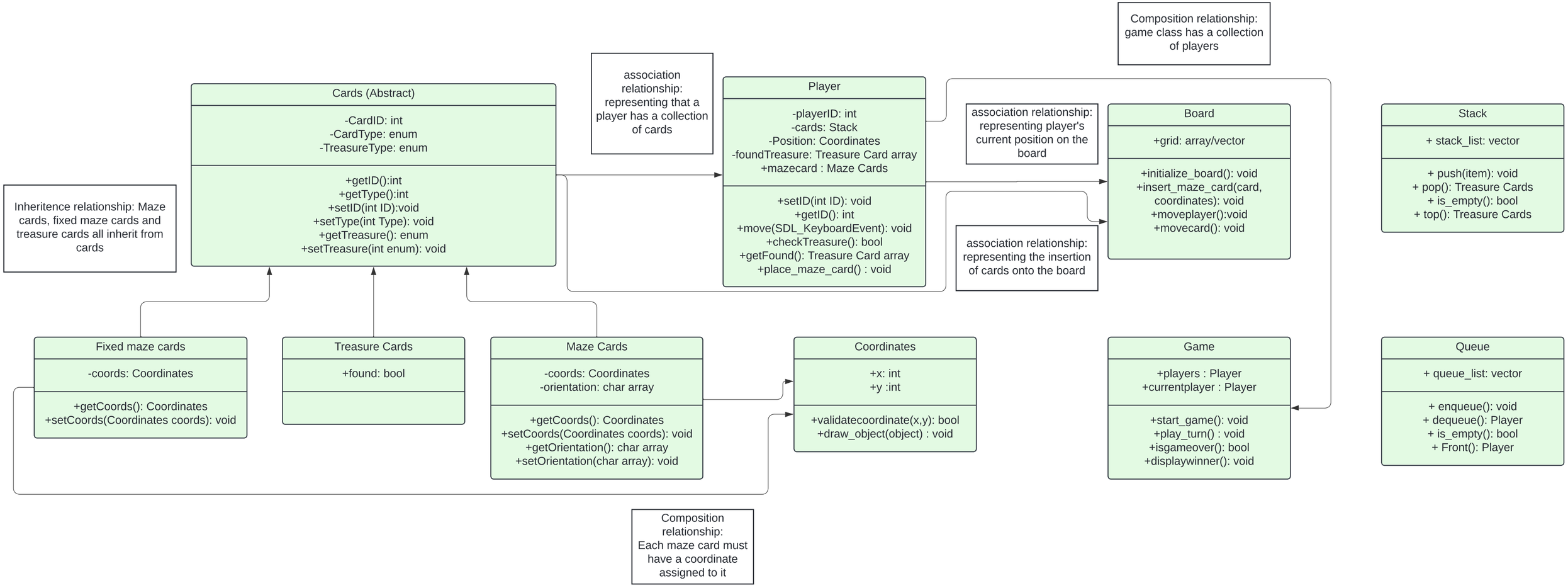





Composition relationship: game class has a collection of players
association relationship: representing that a player has a collection of cards
Cards (Abstract)
-CardID: int
-CardType: enum
-TreasureType: enum
+getID():int
+getType():int
+setID(int ID):void
+setType(int Type): void
+getTreasure(): enum
+setTreasure(int enum): void
Player
-playerID: int
-cards: Stack
-Position: Coordinates
-foundTreasure: Treasure Card array
+mazecard : Maze Cards
+setID(int ID): void
+getID(): int
+move(SDL_KeyboardEvent): void
+checkTreasure(): bool
+getFound(): Treasure Card array
+place_maze_card() : void
association relationship: representing player's current position on the board
Board
+grid: array/vector
+initialize_board(): void
+insert_maze_card(card, coordinates): void
+moveplayer():void
+movecard(): void
Stack
+ stack_list: vector
+ push(item): void
+ pop(): Treasure Cards
+ is_empty(): bool
+ top(): Treasure Cards
Inheritence relationship: Maze cards, fixed maze cards and treasure cards all inherit from cards
association relationship: representing the insertion of cards onto the board
Fixed maze cards
-coords: Coordinates
+getCoords(): Coordinates
+setCoords(Coordinates coords): void
Treasure Cards
+found: bool
Maze Cards
-coords: Coordinates
-orientation: char array
+getCoords(): Coordinates
+setCoords(Coordinates coords): void
+getOrientation(): char array
+setOrientation(char array): void
Coordinates
+x: int
+y :int
+validatecoordinate(x,y): bool
+draw_object(object) : void
Game
+players : Player
+currentplayer : Player
+start_game(): void
+play_turn() : void
+isgameover(): bool
+displaywinner(): void
Queue
+ queue_list: vector
+ enqueue(): void
+ dequeue(): Player
+ is_empty(): bool
+ Front(): Player
Composition relationship: Each maze card must have a coordinate assigned to it