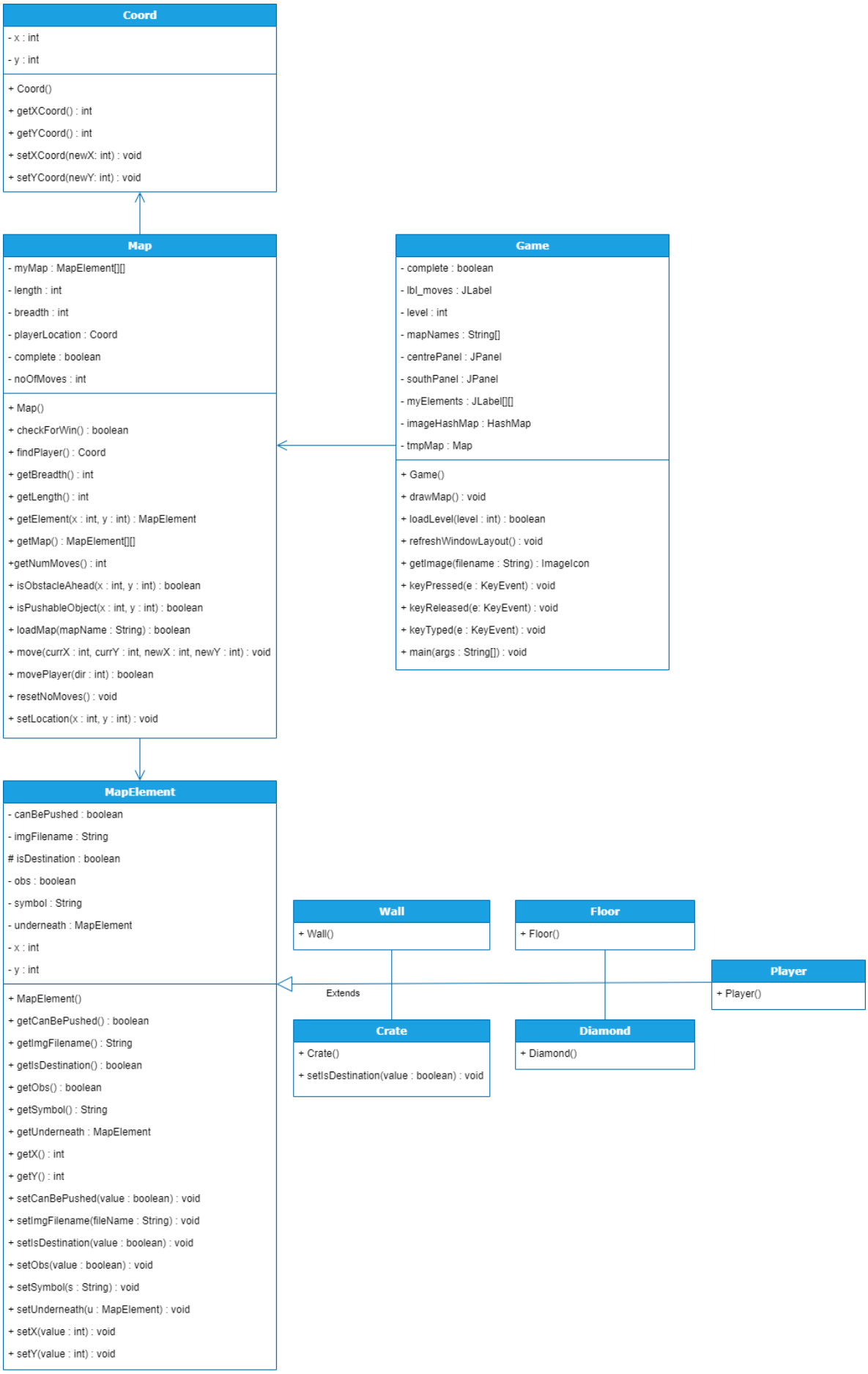
Sokoban

A Java Implementation using Swing

By Calum Lindsay

Cover Image: Warehouse Background (ViridianMoon)

UML Class Diagram



Modifications made to provided diagram

- The displayMap function was removed from the Map class as it reduces the portability of the Map class by making it depend on Swing. It also creates a circular dependency where Game and Map depend on one another as the Game class calls Map.displayMap() which would then update the Game.myElements array.

- refreshWindowLayout function was added to the Game class to refresh the layout of the window whenever a different sized map is loaded. The functionality was put in a separate function as the same code needed to run in multiple places and it also looks much neater.

- Game.dir member variable was removed as there is no need to keep track of the players direction because the icon being used to represent the player doesn't change based on direction nor is direction used in any functional way by the game.

- Game.playerLoc member variable was removed as the Map class already stores the same information and has other methods for interacting with the player class suggesting it is the more appropriate place for it.

- loadLevel function added to the Game class to reduce excessive code duplication caused by validation code required around the Map.loadMap and Map.findPlayer method calls.

- Changed Map.loadMap Function type to boolean to allow the function to return false if there is a problem loading the map.

- Added a setIsDestination method to the Crate class to handle the image changing and locking the Crate in place when it is on top of a Diamond.

- Changed the MapElement.isDestination member variable from private to protected to allow modification from child classes.

- Removed setBreadth and setLength methods from the Map class as loadMap automatically determines the size of the map and being able to change these values arbitrarily could cause ArrayIndexOutOfBounds exceptions to be thrown.

Testing

User Input & Gameplay

Testing for errors caused by unexpected or strange user input (edge cases such as if the player attempts to move out of the playable area, random key presses causing unexpected behaviour). The player should not be able to leave the playable area and the program should behave in an expected manner during any unexpected user input.

No	Test Description	Result	Action Taken
1	Check all movement keys have the desired effect when pressed (left or A key moves the character to the left, right or D Key moves to the right, etc).	Success	No Action Required
2	Check that holding a movement key causes the action to repeat and moves the character continuously until the key is let go.	Success	No Action Required
3	Check that the user is not able to move the character outside of the map area.	Success	No Action Required
4	Check that the user is blocked properly by Walls.	Success	No Action Required
5	Check that the user is properly blocked by a Crate when there is a Wall or another Crate behind it.	Success	No Action Required
6	Check that the user can push Crate's wherever there is nothing blocking the Crate from moving.	Success	No Action Required
7	Check that the image updates when a Crate is pushed on to a Diamond and is locked in place	Success	No Action Required

Game Data File Removal/Corruption

Testing for errors caused by removal or corruption of game data files. Map files should be modifiable without causing crashes as far as possible. Where garbage data is encountered the program should fail elegantly. If Image files are missing the game should fail elegantly.

No	Test Description	Result	Action Taken
1	Remove any/all the image files required by the program to run. This should result in the game showing a blank grid where the map would be, displaying an error that an image file is missing and only accept reload attempts and quitting the game as input.	Success	No Action Required
2	Remove any/all the map files required by the program to run. This should result in the game showing a blank grid where the map would be, displaying an error that a map file is missing and only accept reload attempts and quitting the game as input.	Success	No Action Required
3	Replace an image file with one of a different width/height. This should result in an error message explaining that all images must be 32x32 and the program should then only accept reload attempts and quitting the game as input.	Success	No Action Required
4	Replace a map file with one that has no player tile. This should result in an error message explaining there is no player on the map and the program should then only accept reload attempts and quitting the game as input.	Success	No Action Required
5	Replace the contents of a map file with garbage data that cannot be interpreted as a map by the program. This should result in the program displaying a message that the map is too small and must be at least 4x4 and the program should then only accept reload attempts and quitting the game as input.	Success	No Action Required
6	Replace the map with one that is under 4 tiles in width or height. This should result in the game displaying a message explaining that the minimum map size is 4x4 and the program should then only accept reload attempts and quitting the game as input.	Success	No Action Required
7	Replace the map with one that is over 30 tiles in width or height. This should result in the game displaying a message explaining that the maximum map size is 30x30 and the program should then only accept reload attempts and quitting the game as input.	Success	No Action Required

List Of References

Third party assets used in this report

VIRIDIANMOON, WAREHOUSE BACKGROUND, DIGITAL IMAGE, DEVIANTART, ACCESSED 21 APRIL 2022, < https://www.deviantart.com/viridianmoon/art/Warehouse-Background-679921220>.

Third party assets used in Sokoban implementation

RAVANOK, MALE CHARACTER, DIGITAL IMAGE, RAVANOK.ITCH.IO, ACCESSED 10 NOVEMBER 2021, < https://ravanok.itch.io/cute-rpg-icons>.