EXTENSIBLE SOKOBAN
Brian Herman, Daniel Iarrera. Supervisor: Burton Ma

INTRODUCTION
Our project tackles the problem of providing early computer science students with an interactive and interesting assignment to get them to learn about classes, inheritance and introduce design patterns. We will be using the game Sokoban, a simple yet difficult puzzle game where a player must push boxes onto storage spots to win. This assignment takes a basic implementation of Sokoban and refactors it into an extensible version upon which classes can be used to introduce new elements to the game known as modifications. This gives the students opportunities to investigate the code and understand the game functions, to create their own classes. Students will also get the opportunity to use their own creativity and will be given free rein to come up with anything that they will be able to try and implement.

AIM/PURPOSE
The goal of this project was to provide future CISC 124 students with an assignment that will teach them about inheritance, classes, as well as a few design patterns. The assignment for the students should be simple enough that they will not get caught up in the base code and can focus on their classes. However, the range of modifications that they can implement should be able to support most ideas that the students can come up with so as to not hinder their creativity. Being able to support a range of unexpected modifications that the students will implement, will be the challenge of the project. The ideal code would allow students to design simple to complex modifications that work well within the game environment.

METHODS
This project was coded entirely using Java 8, and the GULI was done with Swing built on the AWT (Abstract Window Toolkit) API. Our code needed to work independently from the modifications added by the student. To do this we had to work backwards from a structure that could affect and be affected by arbitrary classes that are separate from the base code.
Our process consisted of restructuring the base code into GUL, base code, and modifications. These would consist of a GUL using Swing and AWT, the base code as a subject game board, and modifications as cloneable observers of the game board. The GUL would be similar to the original game code, with changes made to the drawing method such that each object on the board will be drawn after each game update. The objects on the board such as player, box, storage, and wall will all be considered modifications and will be cloneable observers of the board. When an event occurs on the board all modifications will be notified and updated according to the event command. This would allow modifications to act independently from the main game yet still be able to interact with the board and one another.

RESULTS/CONCLUSION
The project achieves our original goal of developing a fully extensible version of Sokoban. Along with this extensible game, we created an assignment for future CISC 124 students, provide an example solution, and gave a few extra example modifications. Students will be able to create their own versions of the game objects or entirely new objects that will interact with the game. To do this students will use their understanding of classes, inheritance, and design patterns to write classes, create levels, add images, and configure instances of their classes in the game.