# Sokoban – Project Description

This project is designed following the MCV paradigm.

The UML diagram is at the end of the doctument.

## View

The view package holds the classes in charge the GUI. The icons for the GUI are placed inside the package's folder.

### Sokoban

This class hodls the Main() method for the project.

This class is a JFrame implementation. It holds an instance of a **BoardModel** that it then parses to instances of all the controllers. In addition, It uses an instance of the **BoardDraw** class to draw the board. It receives messages from the Controllers (mainly a view update request) and uses the **BoardDraw** class to do so.

### BoardDraw

This class contains the "drawing" logic of the game board. It receives Cell arrays and returns an appropriate **JPanel** instance. The Sokoban class has an instance of this object that it uses when it receives a view-update request.

## Model

The view package holds the classes in charge the GUI.

### BoardModel

Holds a 2D **Cell** array representing the board. This class calculates moves and verifies if victory is achieved. The class receives messages from the classes in the Controller package.

### Direction (Enum)

The **ENUM** represents a direction on the board – *UP*, *DOWN*, *LEFT*, *RIGHT*.
It is used for the **BoardModel**.makeMove() method.

## LevelLoader

The levelLoader package holds the classes that load different levels from text files.

### LevelLoader

This class reads .txt files and returns a corresponding 2D **Cell** array.

### Cell

This class is the building block of the game board. It represents a single cell in the game.

# Controller

The Controller package contains the listener classes. This "listens" to user input and sends messages to the model package and view package accordingly.

## MainController

The main controller in the project.

Upon arrow key-press will send a message to the **BoardModel** to make a move in the designated direction

Upon the user pressing the Reset button, the controller sends a message to the **BoardModel** to reset.

Upon user choosing an element in the level list the controller sends a message to the **BoardModel** to load a new level accordingly.

After a message is sent to the Model package, a message is then sent to the Sokoban class that instructs it to update the view according to the board.

## UndoController

This class is the action listener for the Undo button.

Upon the user pressing the button calls the **BoardModel**.undo() method. Afterwards a message instructing a view update is sent to the Sokoban class.

## BoardDraw
<<Java Class>>
view

- _characterIcon: ImageIcon
- _floorIcon: ImageIcon
- _crateIcon: ImageIcon
- _storageIcon: ImageIcon
- _wallIcon: ImageIcon
- _correctCrateLocation: ImageIcon

- BoardDraw ()
- drawGameBoard(Cell[][]):JPanel
- createBoard(Cell[][]):JLabel[][]
- transpose(Cell[][]):Cell[][]

## MainController
<<Java Class>>
controller

- _levelList: JList<String>

- MainController(Sokoban,BoardModel)
- setBoard(BoardModel):void
- getBoard():Cell[][]
- getNumberOfLevels():int
- valueChanged(ListSelectionEvent):void
- keyPressed(KeyEvent):void
- actionPerformed(ActionEvent):void
- updateJList(JList<String>):void
- boardChanged():void
- keyReleased(KeyEvent):void
- keyTyped(KeyEvent):void

## EnumDirection
<<Java Class>>
model

- EnumDirection()

## Direction
<<Java Enumeration>>
model

- NONE: Direction
- UP: Direction
- DOWN: Direction
- LEFT: Direction
- RIGHT: Direction

- Direction()

## Cell
<<Java Class>>
levelLoader

- _x: int
- _y: int
- _isFloor: Boolean
- _isStorage: Boolean
- _hasPlayer: Boolean
- _hasBox: Boolean

- Cell(int,int)
- Cell(int,int,Boolean,Boolean,Boolean)
- isFloor():Boolean
- isStorage():Boolean
- hasPlayer():Boolean
- hasBox():Boolean
- set_isStorage(Boolean):void
- set_hasPlayer(Boolean):void
- set_hasBox(Boolean):void
- isEmptyFloor():boolean
- clone():Cell

## Sokoban
<<Java Class>>
view

- _levelList: JList<String>
- _scoreLabel: JLabel
- _score: int
- _scoreText: String
- _fullWindow : JSplitPane

- Sokoban()
- main(String[]):void
- getScore():int
- setScore(int):void
- getBoard():BoardModel
- setBoard(BoardModel):void
- victory():void
- paintNewBoard(Cell[][]):void
- createLevelPanel():JPanel
- createRightPanel():JSplitPane

## UndoController
<<Java Class>>
controller

- UndoController(Sokoban,BoardModel)
- actionPerformed(ActionEvent):void
- setBoard(BoardModel):void

## BoardModel
<<Java Class>>
model

- BoardModel()
- BoardModel(int)
- BoardModel(int,String)
- getCellArray():Cell[][]
- getNumberOfLevels():int
- makeMove(Direction):boolean
- checkVictory():boolean
- undo():boolean
- cloneCellArray(Cell[][]):Cell[][]
- loadNewLevel(int):void
- storageVectorInit():void
- getNextCell(Cell,Direction):Cell
- getPlayerCell():Cell

## LevelLoader
<<Java Class>>
levelLoader

- CHAR_WALL: char
- CHAR_FLOOR: char
- CHAR_PLAYER: char
- CHAR_STORAGE: char
- CHAR_BOX: char
- CHAR_BOX_IN_STORAGE: char
- CHAR_PLAYER_AT_STORAGE: char

- LevelLoader()
- load(String):boolean
- getLevelsCount():int
- get(int):Cell[][]
- parseCell(int,int,char):Cell

-_storageVector   0..*
-_levelGrid   0..*
-_undoStack   0..*   -_levels   0..*
-_levelLoader   0..1
-_board   0..1
-_board   0..1
-_board   0..1
-_controller   0..1
-_game   0..1
-_undoController   0..1
-_game   0..1