Data Structures and Algorithms

COMP 103 2019-20 Semester 2

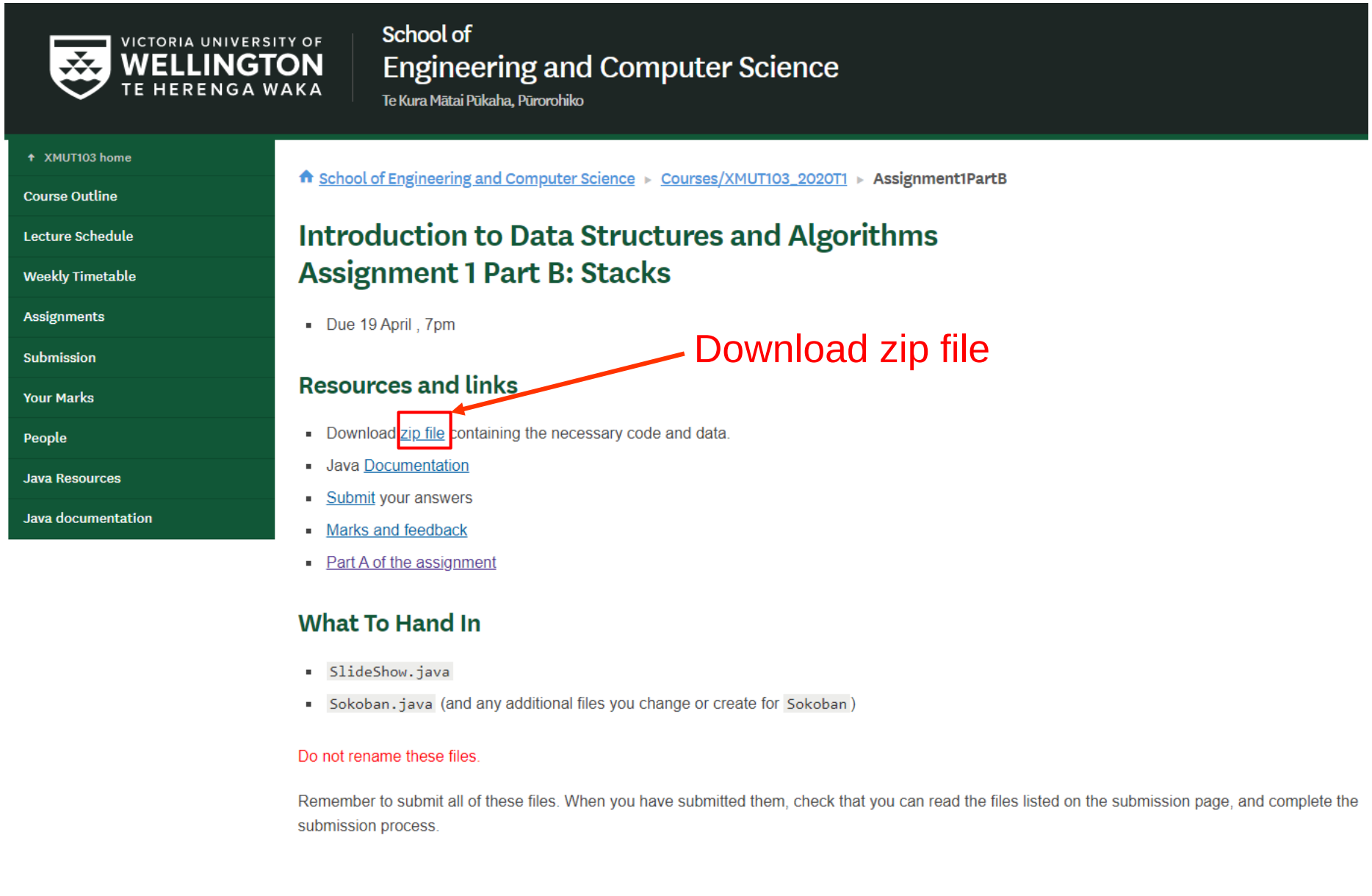
Lecture 03b

Dr. Kerese Manueli

kerese.manueli@ecs.vuw.ac.nz

Victoria University of Wellington

https://ecs.wgtn.ac.nz/Courses/XMUT103_2020T1/Assignment1PartB



https://ecs.wgtn.ac.nz/Courses/XMUT103_2020T1/Assignment1PartB

Undo for Sokoban (Weight: 1/3)

Sokoban is a computer game of the puzzle variety that was created in 1980, and is available on many computer platforms. The game involves controlling a worker in a warehouse who has to push boxes around the warehouse to get them onto their destination spots. The worker can only push one box at a time, and cannot pull boxes. It is easy for the player to get stuck in a deadlock where it is no longer possible to move some of the boxes. Some of the levels are extremely difficult to solve.

You can find out more about the Sokoban game from the web.

We have provided a simple implementation of the Sokoban game with four levels. You can control the worker with buttons or keys.

Our implementation of Sokoban is frustrating because if you make a mistake, you have to restart the game from the beginning. The game desperately needs an *undo* facility, so that you can undo actions if you discover you have made a mistake. It should be possible to undo all actions, all the way back to the beginning of a level.

For this assignment, you need to add an **undo** button to `Sokoban`. It should use a `Stack` that contains the history of actions since the beginning of the current level. Every time the player performs a (successful) action, the action should be recorded on the stack. Every time the player clicks the `undo` button, the program should pop the top action from the history and "undo" it: if it was a move, the worker should be moved back (in the opposite direction of the recorded direction); if it was a push, the worker should be moved back, along with the box that was pushed.

The amount of code you have to write is quite small, but to work out what is needed and where to put it, you will have to read and understand a lot (if not all) of the Sokoban program. This will not be trivial and you should expect to spend time reading the program carefully.

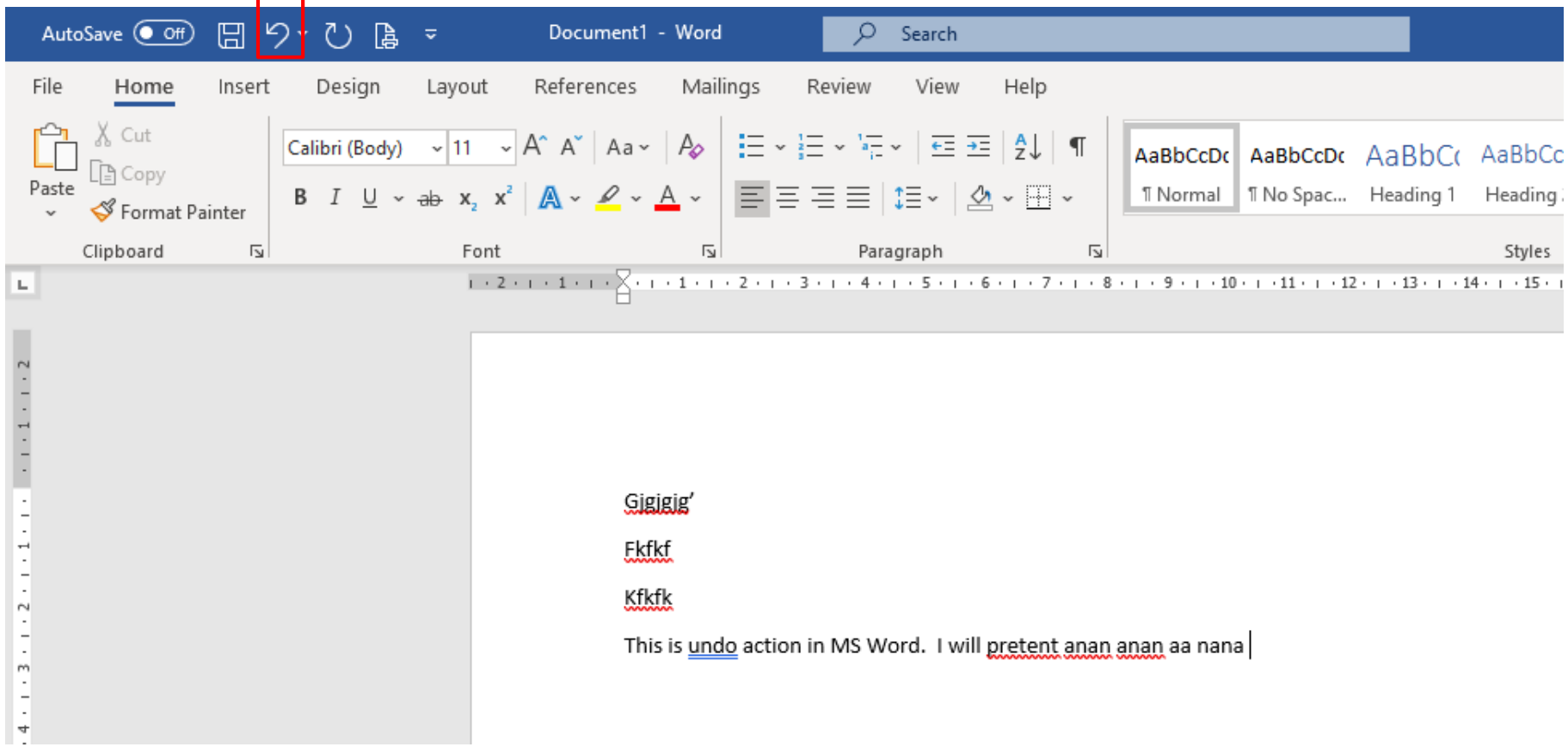
To help you with implementing the undo functionality, we have provided an `ActionRecord` class for recording information about actions.

To implement the undo facility, you need

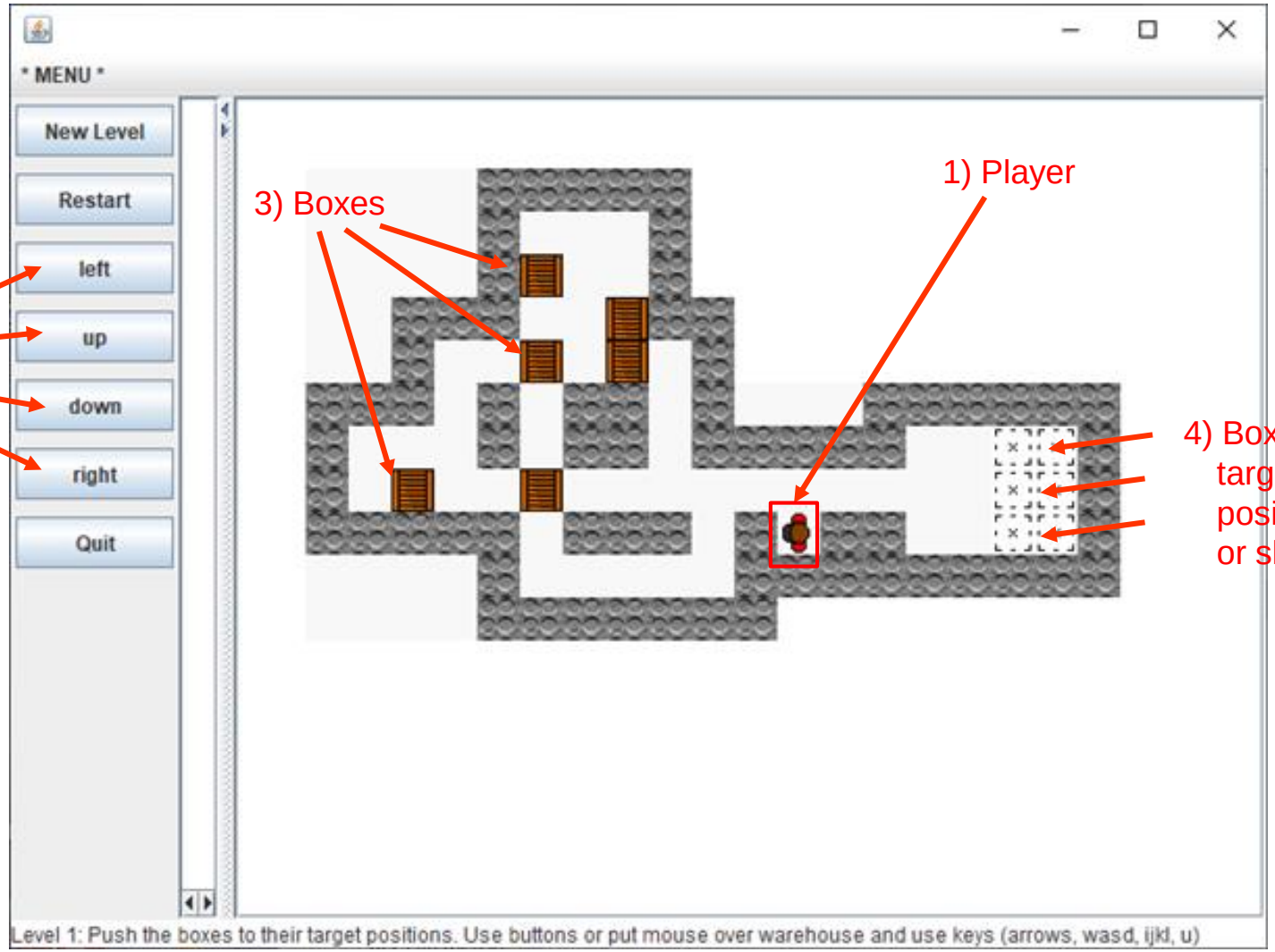
- a `history` field that contains a `Stack` of `ActionRecord`
- extra lines of code in the methods that perform the actions to create an `ActionRecord` object and push it on the history stack.
- extra lines of code in the `load` method to reset the history stack when the level is reset.
- an "Undo" button (plus optionally a key mapping from "u" to the undo method).
- an `undo` method that is called when the "Undo" button is clicked, which pops the top ActionRecord off the stack, and reverses the action.

Core

Undo implementation in MS Word



Sokoban game



1) Player

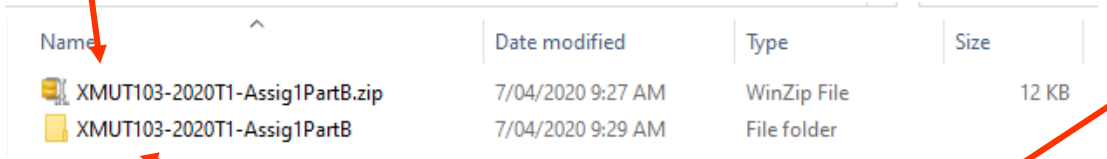
2) Player's navigation buttons

3) Boxes

4) Boxes target positions or shelves

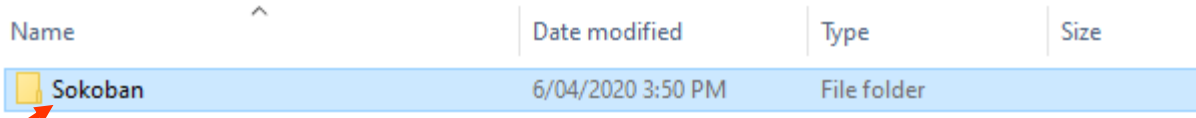
Download XMUT103-2020T1-**Assign1PartB.zip**

1) Download zip file from assignment web page

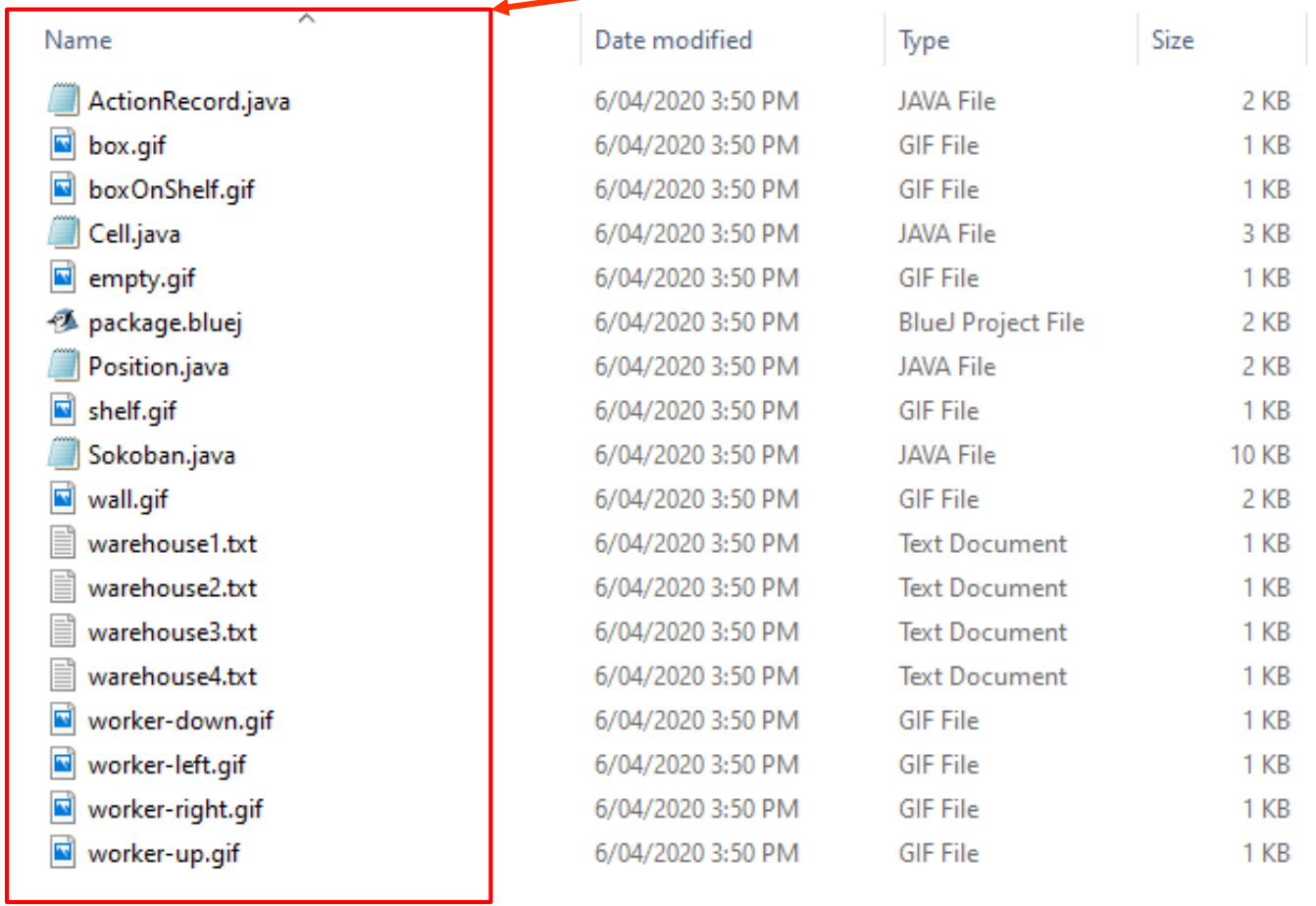


2) Unzip file & you will get a folder as shown

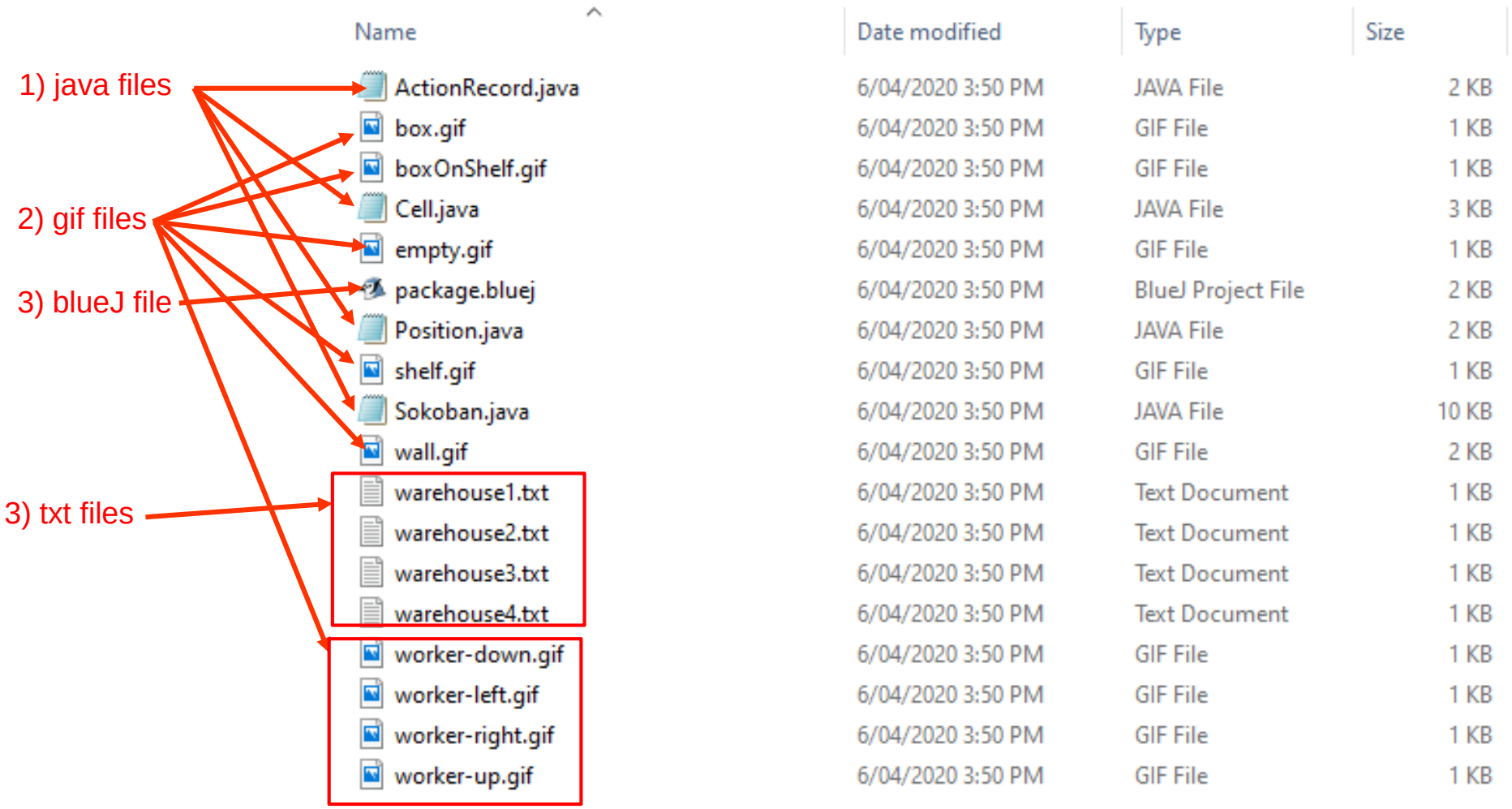
3) Open this folder and you will find Sokoban folder



3) Open Sokoban folder and you will find these 18 files



Files in the **Sokoban folder**



Name	Date modified	Type	Size
ActionRecord.java	6/04/2020 3:50 PM	JAVA File	2 KB
box.gif	6/04/2020 3:50 PM	GIF File	1 KB
boxOnShelf.gif	6/04/2020 3:50 PM	GIF File	1 KB
Cell.java	6/04/2020 3:50 PM	JAVA File	3 KB
empty.gif	6/04/2020 3:50 PM	GIF File	1 KB
package.bluej	6/04/2020 3:50 PM	BlueJ Project File	2 KB
Position.java	6/04/2020 3:50 PM	JAVA File	2 KB
shelf.gif	6/04/2020 3:50 PM	GIF File	1 KB
Sokoban.java	6/04/2020 3:50 PM	JAVA File	10 KB
wall.gif	6/04/2020 3:50 PM	GIF File	2 KB
warehouse1.txt	6/04/2020 3:50 PM	Text Document	1 KB
warehouse2.txt	6/04/2020 3:50 PM	Text Document	1 KB
warehouse3.txt	6/04/2020 3:50 PM	Text Document	1 KB
warehouse4.txt	6/04/2020 3:50 PM	Text Document	1 KB
worker-down.gif	6/04/2020 3:50 PM	GIF File	1 KB
worker-left.gif	6/04/2020 3:50 PM	GIF File	1 KB
worker-right.gif	6/04/2020 3:50 PM	GIF File	1 KB
worker-up.gif	6/04/2020 3:50 PM	GIF File	1 KB

1) Contents of the **ActionRecord.java** file

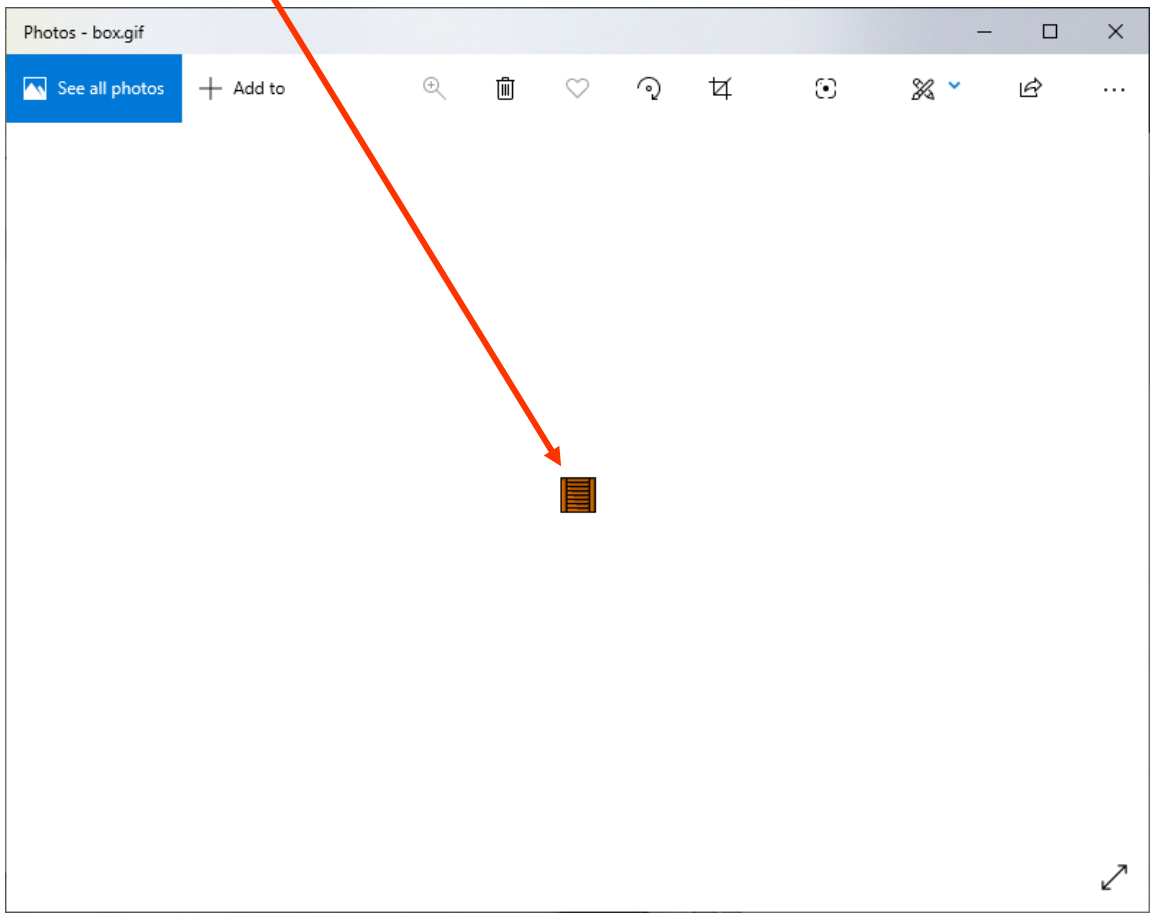
ActionRecord.java - Notepad

File Edit Format View Help

```
// This program is copyright VUW.
// You are granted permission to use it to construct your answer to a XMUT103 assignment.
// You may not distribute it in any other way without permission.

/* Code for XMUT103 - 2020T1, Assignment 1
 * Name:
 * Username:
 * ID:
 */

/**
 * Object containing the record of an action (move or push) in
 * a given direction.
 * Used for the Undo process.
 * Every move or push should put an ActionRecord on the history stack
 * Undo should pop an ActionRecord off the history stack and
 *  undo the recorded action.
 */

public class ActionRecord {
    private final boolean isPush;   // if it is not a "push", it is a "move"
    private final String direction; // direction of the move or push

    /**
     * Constructor
     */
    public ActionRecord(String action, String dir) {
        isPush = (action.equalsIgnoreCase("push"));
        direction = dir;
    }

    /**
     * Is the recorded action a push?
     */
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

2) box.gif file



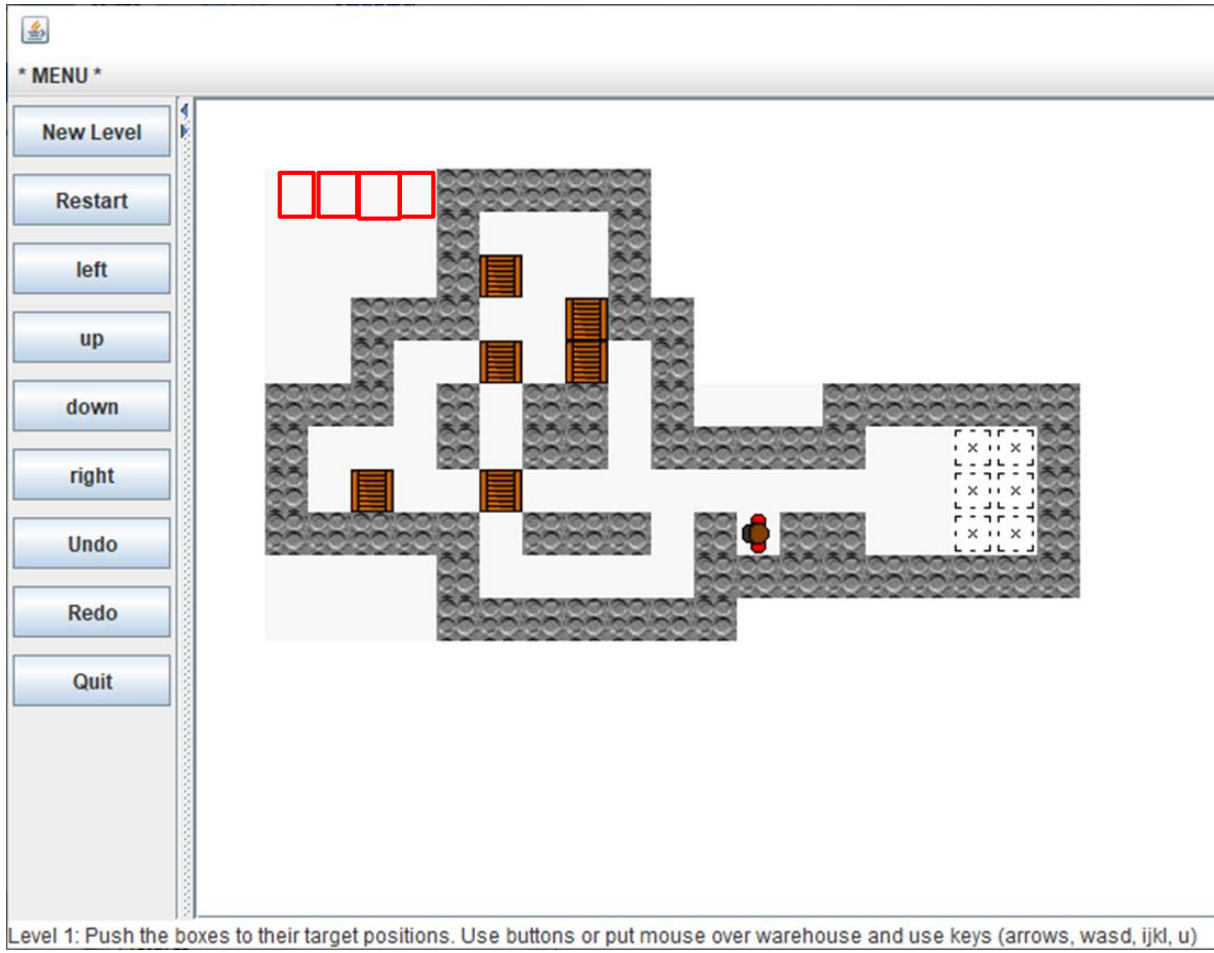
Contents of the **warehouse1.txt** file

warehouse1.txt - Notepad

File Edit Format View Help

```
....#####
....#...#
....#b..#
..###..b##
..#..b.b.#
###.#.##.#...######
#...#.##.#####..ss#
#.b..b.........ss#
#####.###.#w##..ss#
....#.....#########
....#######
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8



Contents of the **warehouse1.txt** file

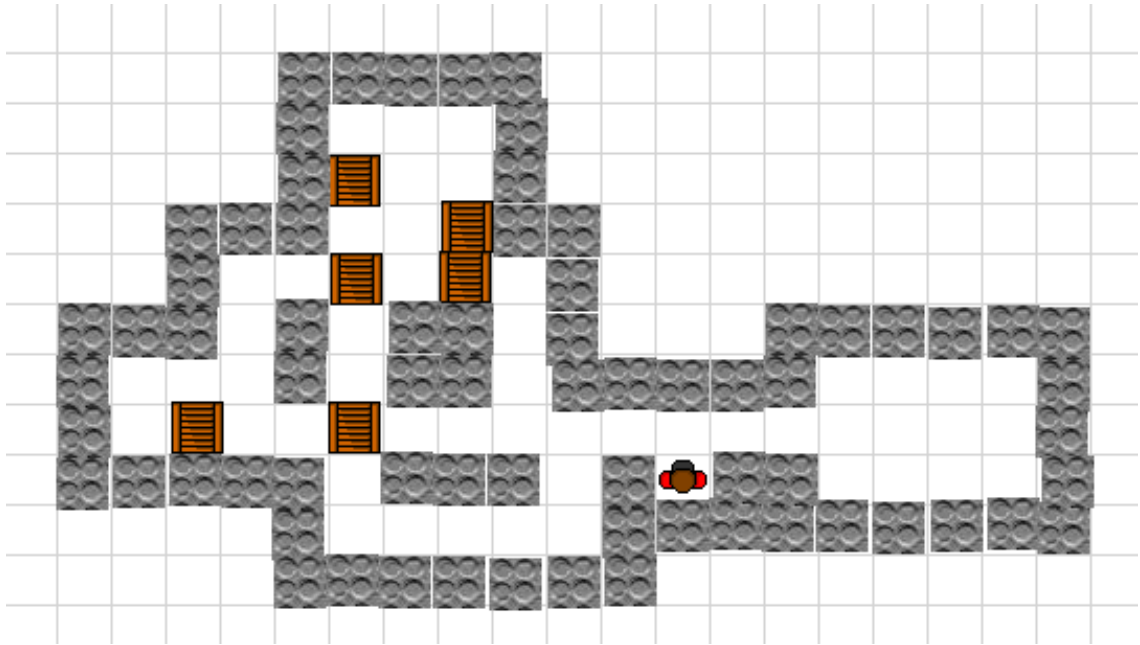
warehouse1.txt - Notepad

File Edit Format View Help

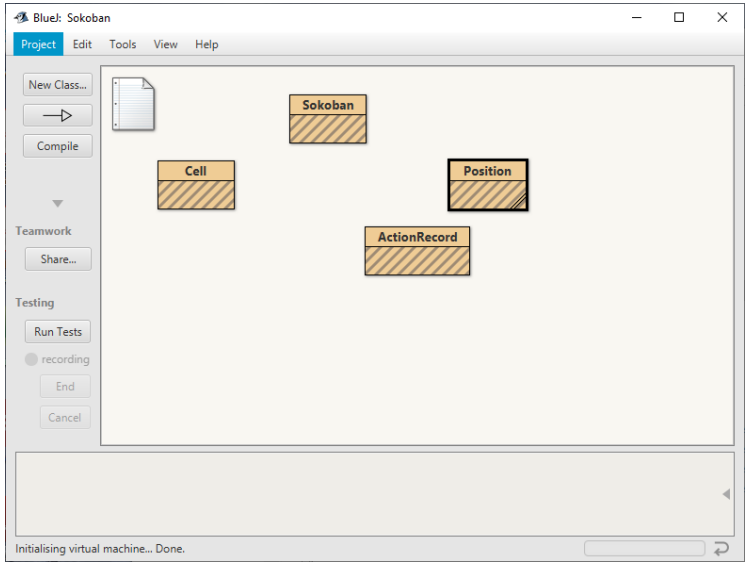
```
....#####
....#...#
....#b..#
..###..b##
..#..b.b.#
###.#.##.#...######
#...#.##.#####..ss#
#.b..b.........ss#
#####.###.#w##..ss#
....#.....#########
....#######
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

. - dot
- hash
b - box
s - shelf or target position
w - worker



1) Open the **package.bluej** file



2) Compile & run the Sokoban program