A presentation slide with a black background. On the left, a blue speech bubble contains the text 'Homework #1 Basic Maze and File Parsing'. To the right, the word 'Goal:' is followed by a numbered list of two items. The first item is '1. Get familiar with basic UML modeling' with sub-points 'A. Understand the given UML diagrams' and 'B. Understand Java code according to UML diagrams'. The second item is '2. Practice basic OO techniques using Java, including:' with sub-points 'A. Using classes and methods', 'B. Basic I/O', and 'C. Java compilation and configuration'. A small speaker icon is located below the list.


**Homework**

**#1**

**Basic Maze and File Parsing**

**Goal:**

1. Get familiar with basic UML modeling
  - A. Understand the given UML diagrams
  - B. Understand Java code according to UML diagrams
2. Practice basic OO techniques using Java, including:
  - A. Using classes and methods
  - B. Basic I/O
  - C. Java compilation and configuration



Your first homework assignment is a basic maze and file parsing.

Homework #1 has a few goals.

The first is to get familiar with basic UML modeling.

To do this you'll need to understand the given UML diagrams. and then you'll need to understand Java code according to UML diagrams.

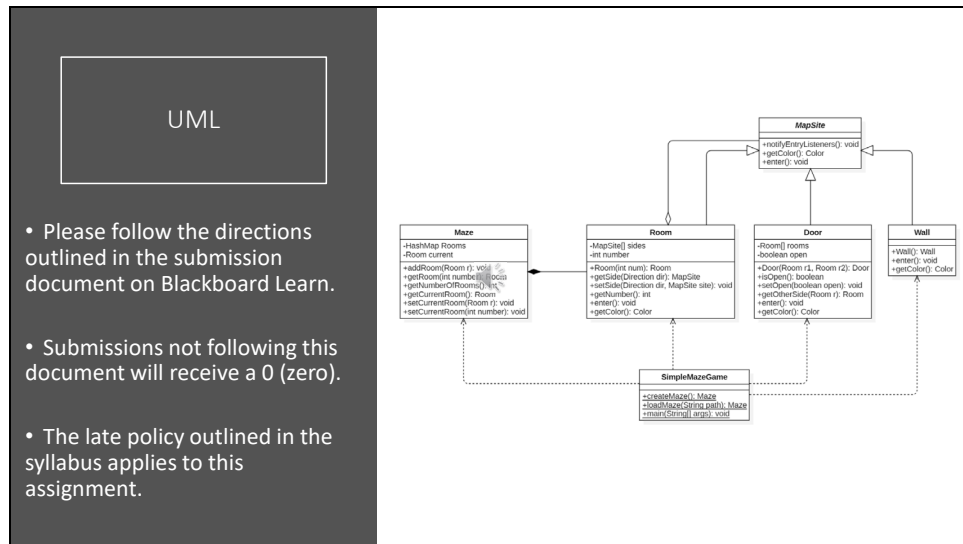
Next, you'll practice basic OO techniques using Java, including:

Using classes and methods

Basic I/O and

Java compilation and configuration

This assignment is also the first part of a lab that we'll do much later, so it's important you get this to work or you won't be able to start the lab



Observe the UML for the maze classes.

A maze has a hashmap of rooms.

A room is itself a MapSite which is an abstract class.

MapSite has multiple derived classes, which are rooms, doors and walls.

A room has an array of mapsites called sides.

Each side can be a room, door or wall

We instantiate a SimpleMazeGame to create a maze.

SE 310 – Introduction to JAVA - UML

**Requirements:**

1. Compile the given program to show an empty maze game.
2. Modify methods in the given code to add rooms to the maze.
3. Compile the modified program to show a simple 2 room maze, each with 3 walls, and connected by a door.
4. Modify the program again to read from input files that specify maze structures.
5. Compile the modified programs to show the maze using the large.maze file. If you're unable to parse the large.maze file you may use the small.maze file and receive a penalty.
6. Your final program should run with the ability to move your player through the maze for steps 4 & 5.

There requirements for your assignment are as follows:

- Compile the given program to show an empty maze game.
- Modify methods in the given code to add rooms to the maze.
- Compile the modified program to show a simple 2 room maze, each with 3 walls, and connected by a door.
- Modify the program again to read from input files that specify maze structures.
- Compile the modified programs to show the maze using the large.maze file. If you're unable to parse the large.maze file you may use the small.maze file and receive a penalty and
- Your final program should run with the ability to move your player through the maze for steps 4 & 5.

## Slide 4

SE 310 – Introduction to JAVA - UML

### Instructions

There are 3 stages

#### Stage 1: Make the program run

1. Download **HW1.zip** from Blackboard
2. Extract the contents of **HW1.zip**
3. Import the folder/directory “**HW1**” from the zip file as a new IntelliJ project. Detailed instructions on using IntelliJ and importing zip files can be found on Blackboard. Please follow the instructions.
4. Attempt to run the project. The message “The maze does not have any rooms yet!” will be displayed if you can successfully run the project.
5. Note, that this is a small part of a larger program. You do not need to understand everything that is going on. Just focus on the task at hand.

There are three stages to getting the assignment completed.

Stage 1: Make the existing program run.

- Download HW1.zip from Blackboard
- Extract the contents of HW1.zip
- Import the folder/directory “HW1” from the zip file as a new IntelliJ project. Detailed instructions on using IntelliJ and importing zip files can be found on Blackboard. Please follow the instructions.
- Attempt to run the project. The message “The maze does not have any rooms yet!” will be displayed if you can successfully run the project.
- Note, that this is a small part of a larger program. You do not need to understand everything that is going on. Just focus on the task at hand.

### Instructions

There are 3 stages

#### Stage 2: Build a maze with rooms

1. Fill in the createMaze function in the SimpleMazeGame class to create a 2 room maze.  
You must set the current room number.  
Room numbers start at 0.  
Rooms must be complete with walls and doors .  
If you receive an error related to UI or 'painter' you did something wrong. ***There is nothing wrong with the supplied jar file.***
2. Compile and run the new program to show a two-room maze.

#### Stage 2: Build a maze with rooms

- Fill in the createMaze function in the SimpleMazeGame class to create a 2 room maze.  
You must set the current room number.  
Room numbers start at 0.  
Rooms must be complete with walls and doors .  
If you receive an error related to UI or 'painter' you did something wrong. There is nothing wrong with the supplied jar file.
- Next, compile and run the new program to show a two-room maze.

### Instructions

There are 3 stages

#### Stage 3: Load a maze from small.maze and large.maze

The two maze files are included in the zip. They should be included in your IntelliJ project, inside the project folder but outside of your source folder.

The maze file contains the following information:

1. Object Name (room/door)
2. Object Identifier (i.e. Room number or Door number)
3. If it's a room, four entries follow that are either a:
  - i. Wall
  - ii. Door – Object identifier for a door
  - iii. Room – Object identifier for a room

The four identifiers are in the order North, South, East or West.

For stage 3:

Load a maze from the files small.maze and large.maze

The two maze files are included in the zip. They should be included in your IntelliJ project, inside the project folder but outside of your source folder.

The maze file contains the following information:

- Object Name (room/door)
- Object Identifier (i.e. Room number or Door number)
- If it's a room, four entries follow that are either a:
  - Wall
  - Door which is an object identifier for a door
  - Room which is an object identifier for a room

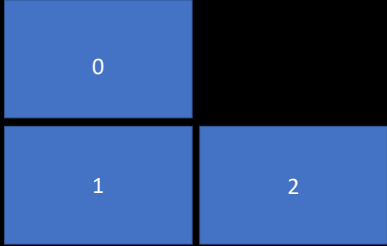
The four identifiers are in the order North, South, East or West.

## Slide 7

SE 310 – Introduction to JAVA - UML

**Instructions**  
There are 3 stages

**Stage 3: Load a maze from small.maze and large.maze**  
For example, the following maze:



has the following file format:

```
room 0 wall 1 wall wall
room 1 0 wall 2 wall
room 2 wall wall wall 1
```

Load a maze from small.maze and large.maze

For example, the following maze which shows 3 rooms has the following file format shown on the screen.

In this case there are three entries, all are rooms.

The rooms are identified as 0, 1, and 2.

The first room, room 0, has a wall to the north, room 1 to the south, and a wall to the east and west.

Similarly, the 2nd room, room 1, has a room to the north, room 0, a wall to the south, a room, room 2 to the east and a wall to the west.

Finally, the 3rd room, room 2, has a wall to the north, south, and east and a room, room 1 to the west.

### Instructions

There are 3 stages

#### Stage 3: Load a maze from small.maze and large.maze

3. Initially, to sanity check your loading functionality, test the system with the small.maze file and work up to large.maze.

Parsing code for small.maze will need to be modified to handle parsing large.maze.

Once you're able to parse large.maze set your main() function to automatically use large.maze and default to using the loadMaze() function instead of createMaze().

Initially, to sanity check your loading functionality, test the system with the small.maze file and work up to large.maze.

Parsing code for small.maze will need to be modified to handle parsing large.maze.

Once you're able to parse large.maze, set your main() function to automatically use large.maze and default to using the loadMaze() function instead of createMaze().



Slide 9


SE 310 – Introduction to JAVA - UML

**Submissions**

Please follow the directions outlined in the submission document on Blackboard Learn.

Submissions not following this document will receive a 0 (zero).

The late policy outlined in the syllabus applies to this assignment as does the academic honesty policy.



To submit the assignment:

- Please follow the directions outlined in the submission document on Blackboard Learn.
- Submissions not following this document will receive a 0 (zero).
- The late policy outlined in the syllabus applies to this assignment as does the academic honesty policy.