SWEN30006

Software Modelling and Design

Project 2 Report (6 pages excluding diagrams)

Group 3 Monday 17:15

Ngoc Duy Tran

Nguyen Linh Dan Le

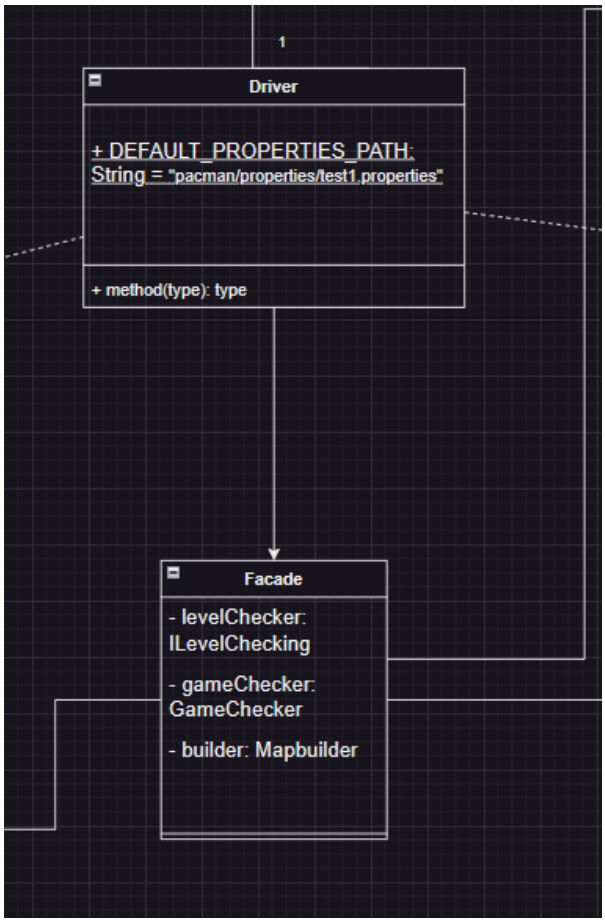
Tran Nhat Minh Dang

1. Design of editor

Originally there were 2 separate folders given, namely pacman folder and 2d-map-editor folder, which contains 2 drivers. After the 2 folders were merged, only one driver was kept which plays a role to handle both systems (Editormode and Testmode). Problem now arises as the classes from 2 system directly interact, this violates **low coupling** and **high cohesion**

1.1. Merging

- Façade class (**Façade pattern)** acts as a controller which connects the main system Game, and the subsystem 2DMapEditor. This helps hide the complexity of the logic layer (2DMapEditor) and give direct control input from outside to a single place, which helps users interact with the subsystem without knowing about the behavior of 2DMapEditor
- In our design Façade object wraps the subsystem which contains the checking processes and map editor phase in the game to remove undesirable coupling to the whole system, and increase cohesion for both Game and 2DMapEditor

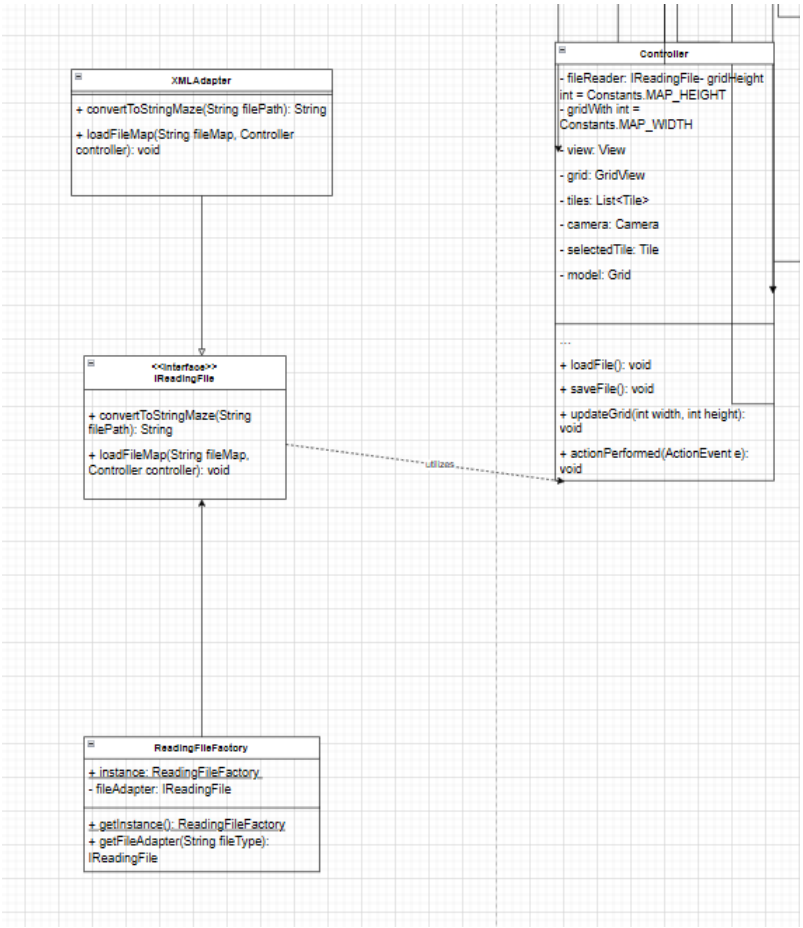


Facade pattern connects the Driver (initialization for Game) with Controller class (which handles saving and loading functions for the 2DMapEditor subsystem)

1.2. Game checking and level checking

1.2.1. Reading files

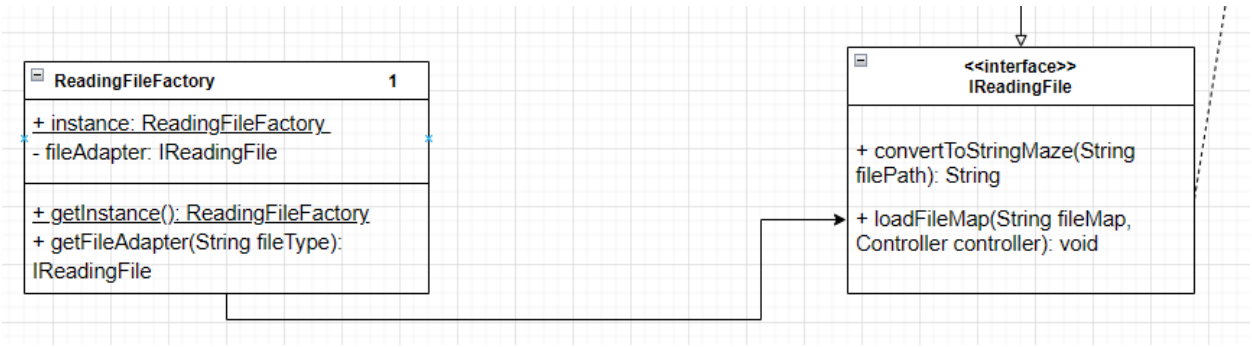
- The spec of the project suggests for the editor mode, we have 2 major tasks to process, which are Game Checking and Level Checking. As in the future, there will be further requirements to be added to the spec of the project, we chose to apply Template and Composite Strategies to the design, following by Factory and Adapter pattern.
- By applying GoF pattern, we implement **IReadingFile interface (Adapter pattern)**, as we assume, in the future if the editor changes the extensions, or the application opens to more file extensions (e.g txt, csv,…), not only .xml; the design should also be model beforehand to handle different types of extensions loaded to the reader in the future.
- The IReadingFile interface interacts with the Controller class from 2DMapEditor, which is mainly responsible for visualizations, loading and saving.
- For example, XMLAdapter, implementing IReadingFile interface, is responsible for reading XML files from the test folder/map filepath, from which the controller can utilize and load the figures.



- Another purpose of IReadingFile interface is to read XML files, and convert to type String maze, which can be input into further steps such as Level Checking and running game (to be continued).
- For further extension, other types of adapters such as CSVAdapter with similar functionality can be applied. Our aim here is to be able to design models and write code for file extension reading which is independent, features **low coupling,** and facilitates future extension.

- By applying GoF pattern, we implement ReadingFileFactory as a Singleton pattern for global access. This is useful as we might need only one instance of the object, which is responsible for producing the desired adapter based on user' input. In our case, ReadingFileFactory is responsible for creating XMLAdapter (**Creator**). This aligns with

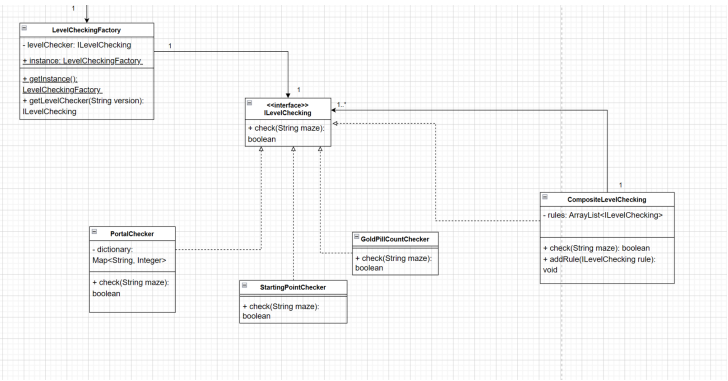
GRASP principles as this results in **Information Expert, Creator, and Pure Fabrication** (this does not exist beforehand)

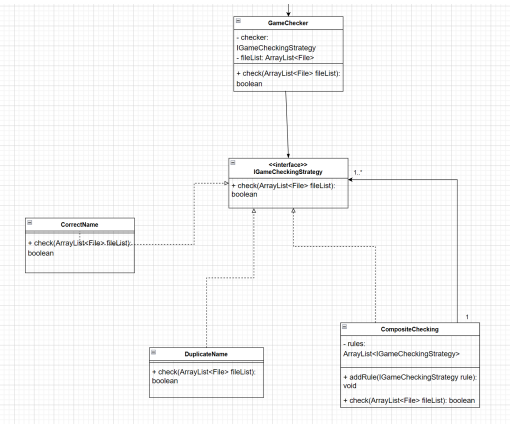


Relationship between ReadingFileFactory and IReadingFile

1.2.2. Game and Level checking

- Game checking is done before Level checking, then the error is updated on the ErrorLog.txt
- There are 2 main requirements for Game checking, which is to find duplicated maps and check for correct names. This is achieved by 2 classes: DuplicatedNames and CorrectNames in our design. Similarly, there are pre-defined set of rules for level checking, which could be further investigated in the design class diagram or the code submission.
- However, as future extensions might require any other sets of rules, we decide to use the **Composite** pattern for both Game and Level checking. In the future, there might be many more requirements (properties) for both kinds of checking, and it would be unreasonable to put all of rules inside a class. Therefore, Composite pattern allows different properties to combine with each other and perform the desired behavior. For example in LevelChecking, a composite class (ie CompositeLevelChecking) and atomic objects (ie PortalChecker, StartingPointChecker, GoldPillCountChecker) are implemented by sharing/implementing the same interface (ie ILevelChecking) to support Composite Pattern.
- **Singleton** is also applied because only one instance is demanded.
- Therefore, by adopting GoF patterns, the resulting architecture possesses better GRASP concept, in which higher cohesion, lower coupling and future extensibility are achieved.





Design diagram of both Game and Level Checking

2. Design of autoplayer:

2.1. Usage Description of the Autoplayer

The autoplayer is designed to autonomously control the PacActor character in a game environment. Its primary goal is to navigate the game maze, collect pills, avoid monsters, and reach the goal location efficiently. The autoplayer utilizes various algorithms and strategies to make decisions on the next move of the PacActor.
In this project, A* which employs a heuristic function(Manhattan distance) is implemented for finding path to Pills and GoldShortest Path Calculation.

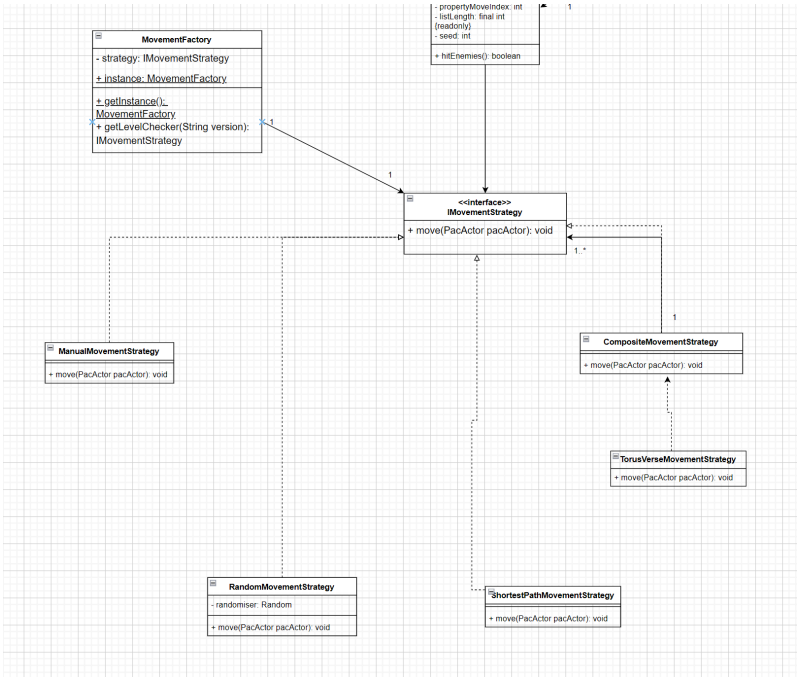
2.2 Extension Guidance:

a. *Information Modifications:*

- Monster Information: Add a new interface or data structure to store information about monsters in the game. This information can include the current location, movement pattern, speed, and any other relevant attributes.
- Pill Information: Incorporate a new interface or data structure to keep track of pill locations in the game. This information should include the coordinates of each pill on the game grid.

b. *Current Movement Interface:*

- New type of movement can be extended by the **Interface IMovementStrategy**
- For future extension with monsters and items' effects, we will use Composite Strategy Pattern along with Singleton Factory to solve this.

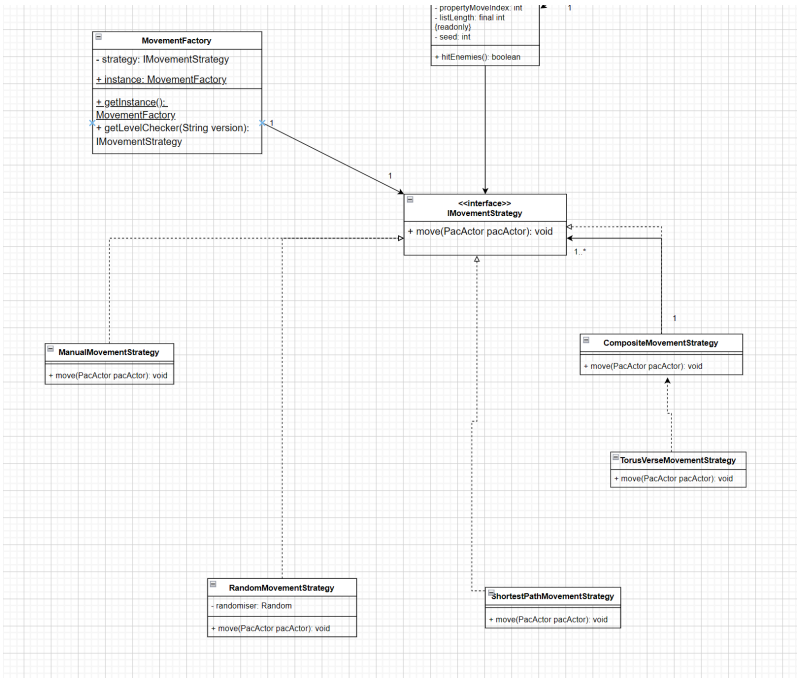


- ManualMovementStrategy allows users to interact with the Game directly.

- RandomMovementStrategy allows PacMan to go randomly when stuck in a place, this acts as a “mutation” step to find the pills and gold. ShortestPathMovementStrategy implements A* algorithm while CompositeMovementStrategy holds a combination of strategies, and TorusVerseMovementStrategy is responsible for choosing which combination. A Singleton factory is implemented to help the construction of TorusVerseMovementStrategy

c. <u>Modifications for future changes (monsters, effects)</u>

- Let’s say the new version is FinalVersion in which monsters and items’ effects are present. In this case Singleton Factory is responsible for constructing the FinalMovementStrategy which inherits from CompositeMovementStrategy, which allows it to hold a combination of strategies.
- Other atomic strategies such as ShortestPathMovementHidingMonsters can be implemented with the aim of finding the closest path to pills and gold; however, the main focus is to run away from Monsters.



- Then, when the conditions of the monsters are notified such as all monsters are frozen: the movementStrategy in PacMan could be adjusted accordingly to ShortestPathMovementStrategy and consider all monsters are walls.

```
1 usage  Alan *
public void moveInAutoMode() {
//    if all monsters are frozen:
//      movementStrategy = ShortestPathMovementStrategy
//    else:
//      movementStrategy = ShortestPathMovementStrategyHidingMonsters
//
//    if countdown > threshold:
//      movementStrategy = BFSMovementStrategy
    movementStrategy.move( pacActor: this);
}
```

- Another case is that if following A* is too consuming and PacMan is stuck inside an area (the countdown time is too large), then Breadth-First-Search might be an alternative to find all possible paths.

Assumptions:

1. For the current implementation, in the presence of monsters and ice cubes, the project currently sets the hasPacManBeenHit = false at all times, to make PacMan invisible when in contact with monsters. We can convert back to the normal version

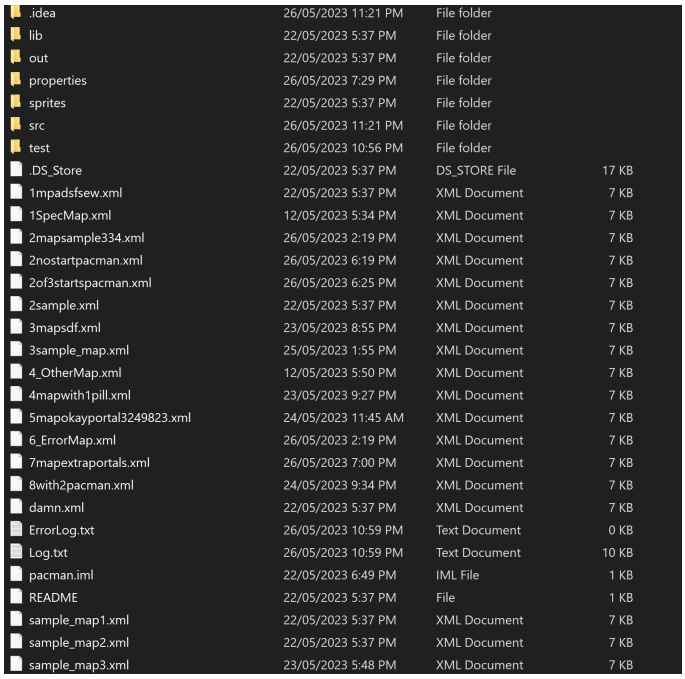
```
do {
//      hasPacmanBeenHit = troll.getLocation().equals(pacActor.getLocation()) ||
//              tx5.getLocation().equals(pacActor.getLocation());
    hasPacmanBeenHit = false;
    hasPacmanEatAllPills = pacActor.getNbPills() >= maxPillsAndItems;
```

2. The default value for PacMan.isAuto is True; however, we can easily convert to the normal case by setting to False in Facade

```
no usages  dduygaucho +2 *
public static void main(String args[]) throws IOException {
    if (args.length > 0) {
        String propertiesPath = args[0];
    }
    boolean auto = false;
    String fileMap = "test";
    new Facade(fileMap, auto);
```

3. All files end with .xml
4. Ed #306 said the start button is not required for test mode on single map.
5. The design and domain model only requires to draw classes related directly to autoplayer and game levels/maps; therefore, unrelated ones such as PacActor, Game, 2DMapEditor is visualized yet not numbered in the relationship (1..n relationship)
6. There is difference between image sizes provided in source code (32x32), while the normal version only uses 30x30 images, which results in misvisualization of portals; however, the functionality and locations of portals are reserved.
7. Xml files outside the folder are just for testing purposes. To **test, put all of the related xml files in the test folder and then just run as normal**. We decided to use a String fileMap (figure above) instead of arguments to preserve the nature of calling src.Driver.main() as highlighted in specification. If an argument is desired, a simple conversion is String fileMap = args[0], which is easy.
8. Please ignore xml files if you want they are just for reference
9. This project acknowledges the use of Chat-GPT for writing report and coding part, such as A* search which is not the aim of this subject; however, the core ideas are original, and just use Chat-GPT to increase productivity.

10. Link to draw.io: https://drive.google.com/file/d/1xS-yf8MIoDc9jq2F1s6gVWMyJBWYHn98/view?usp=sharing



.idea	26/05/2023 11:21 PM	File folder	
lib	22/05/2023 5:37 PM	File folder	
out	22/05/2023 5:37 PM	File folder	
properties	26/05/2023 7:29 PM	File folder	
sprites	22/05/2023 5:37 PM	File folder	
src	26/05/2023 11:21 PM	File folder	
test	26/05/2023 10:56 PM	File folder	
.DS_Store	22/05/2023 5:37 PM	DS_STORE File	17 KB
1mpadsfsew.xml	22/05/2023 5:37 PM	XML Document	7 KB
1SpecMap.xml	12/05/2023 5:34 PM	XML Document	7 KB
2mapsample334.xml	26/05/2023 2:19 PM	XML Document	7 KB
2nostartpacman.xml	26/05/2023 6:19 PM	XML Document	7 KB
2of3startspacman.xml	26/05/2023 6:25 PM	XML Document	7 KB
2sample.xml	22/05/2023 5:37 PM	XML Document	7 KB
3mapsdf.xml	23/05/2023 8:55 PM	XML Document	7 KB
3sample_map.xml	25/05/2023 1:55 PM	XML Document	7 KB
4_OtherMap.xml	12/05/2023 5:50 PM	XML Document	7 KB
4mapwith1pill.xml	23/05/2023 9:27 PM	XML Document	7 KB
5mapokayportal3249823.xml	24/05/2023 11:45 AM	XML Document	7 KB
6_ErrorMap.xml	26/05/2023 2:19 PM	XML Document	7 KB
7mapextraportals.xml	26/05/2023 7:00 PM	XML Document	7 KB
8with2pacman.xml	24/05/2023 9:34 PM	XML Document	7 KB
damn.xml	22/05/2023 5:37 PM	XML Document	7 KB
ErrorLog.txt	26/05/2023 10:59 PM	Text Document	0 KB
Log.txt	26/05/2023 10:59 PM	Text Document	10 KB
pacman.iml	22/05/2023 6:49 PM	IML File	1 KB
README	22/05/2023 5:37 PM	File	1 KB
sample_map1.xml	22/05/2023 5:37 PM	XML Document	7 KB
sample_map2.xml	22/05/2023 5:37 PM	XML Document	7 KB
sample_map3.xml	23/05/2023 5:48 PM	XML Document	7 KB