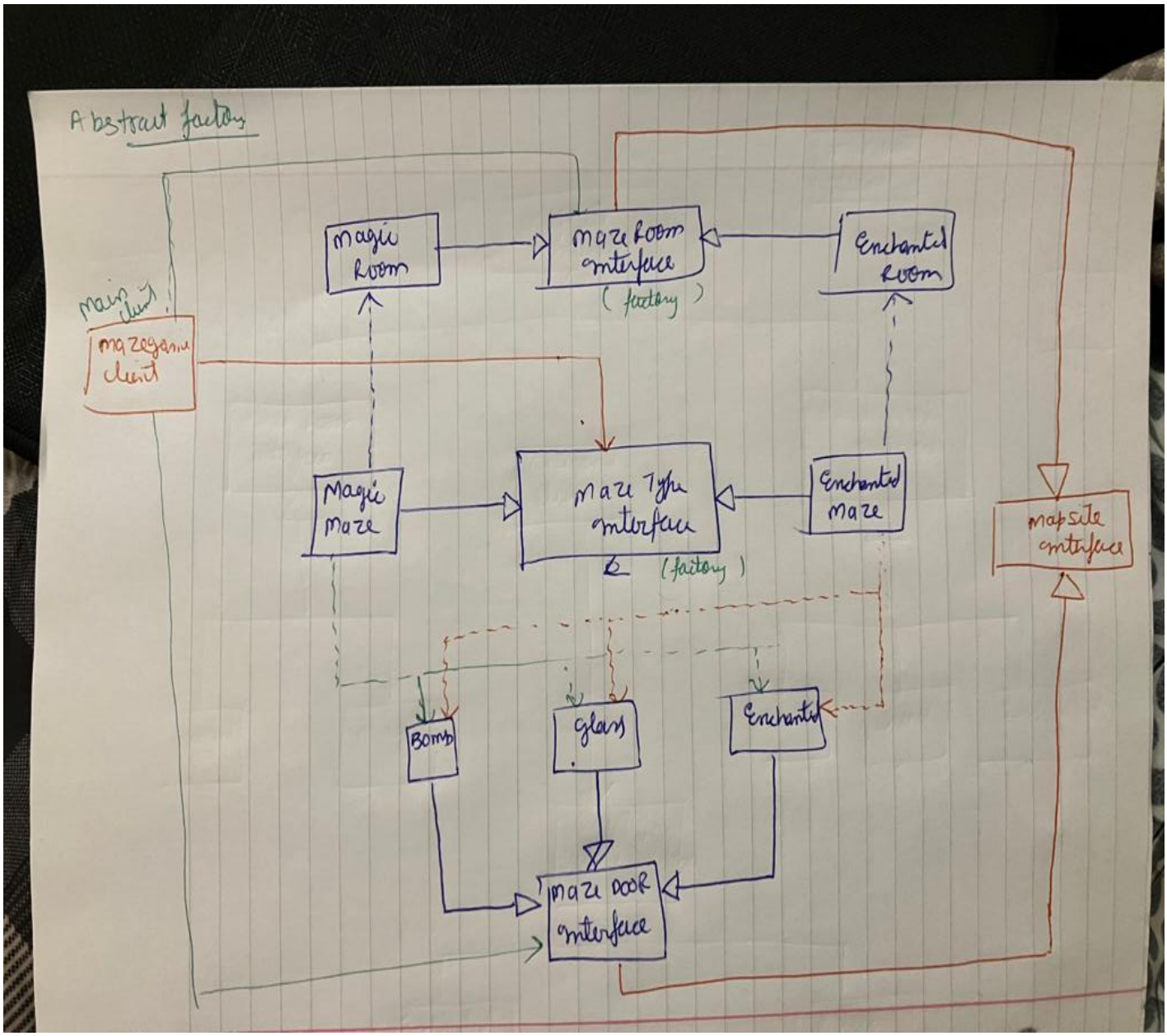
PROJECT 1:

My UML diagram.



In UML diagram I have

Abstract Factory : MazeTypeInterface
Abstract product Room: MazeRoomInterface
Abstract product Door : MazeDoorInterfaces
Abstract Maze Types : Magic Maze, Enchanted Maze
Client class : MazeGame
Room Interface child: Magic room, Enchanted Room
Data access objects of MAZE: Room, Door, Maze

Singleton used in: EnchantedMazeFactory, MagicMazeFactory, EnchatedMazeRoom, MagicMazeRoom

Pros of my design:

1. We can easily extend the design by adding a new type of maze.
2. We can easily extend types of Rooms, types of Doors, in the maze by extending the interface
3. Each class is having single responsibility
4. Client/MazeGame has only access to the MazeFactoryType interface
5. Modularity is maintained throughout the project
6. Less coupling between interfaces and concrete classes
7. Interface segregation has been done properly
8. The Liskov substitution principle has been implemented accurately.
9. Data access objects has been created for Room, Door and Maze so in future if Database is added it will be easy to implement
10. Due to the singleton pattern in the maze factory I have restricted myself to the creation of new objects in one single game.
11. If wall object is required later on then we can easily add the wall with minimal code change
12. Adding more properties in maze class won't affect the whole code.
13. Both the Room and Doors are independent
14. Proper Exception handling has been done
15. Proper Error Message and user directions have been given to run the program.

TradeOffs:

1. While extending the design, more complexity will be introduced, this complexity might result in coupling to some reason but law of demeter will be followed.
2. Due to no use of prototype pattern, cloning of rooms object has increased the code replication

3. Abstract factory pattern is good to use but using builder along with abstract factory might have given more better result in terms of coupling and cohesion
4. Due to more number of child class, there is duplication of some of the code in child classes

About the code running guidlines:

1. Driver packed consist of main class from where code will be run.
   Main class is the entry point of the code
2. User is asked to enter type of maze required.
3. User then enters the number of rooms required
4. User then enters the type of door required in the maze
5. User then select the layout required
6. Output will be shown.

Sample output:

```
---We have two types of maze. Kindly select the maze type by entering choice (1 or 2)---
1. Enchanted Maze   2. Magic Maze
1

---Please enter number of rooms required in a MAZE---
3

Door Types available for the Maze are
1. Bomb Door      2. Enchanted Door  3. Glass Door
Enter the choice from 1 to 3
1
doorChoice 1
Enter your choice for maze layout (1/2)
1

.....YOUR DESIGNED MAZE WILL LOOK LIKE .....

Room : 1
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 2
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 3
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 4
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 5
```

```
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 6
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 7
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 8
    Room has doors :  NORTH    SOUTH    EAST    WEST
Room : 9
    Room has doors :  NORTH    SOUTH    EAST    WEST

----------WILL BRING YOUR MAZE TO HTML DESIGN SOON----------

THANK YOU FOR USING OUR LAYOUT

Do you want to continue again (Y/N)???N

 EXITING THE GAME. GOOD BYE!!!

Process finished with exit code 0
```

GITHUB LINK TO SOURCE CODE: https://github.com/05satyam/MazeGame
**** THE REPOSITORY IS PRIVATE AS OF NOW. IF ACCESS REQUIRED KINDLY LET ME KNOW DR. ALEX.**

NOTE:
There are more changes which can be done but I have fulfilled all the requirements of the document. I have tried to enclose as many comments as required if something is missed kindly let me know.
Thank You

Satyam Mittal