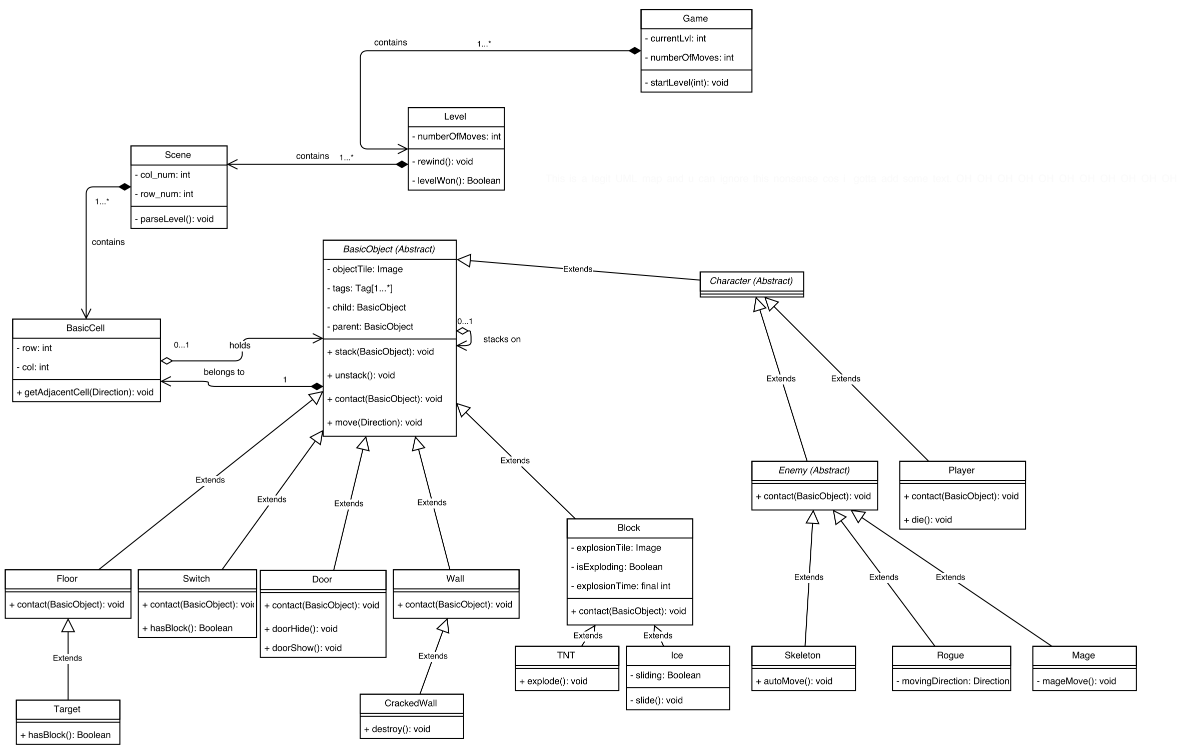
OOSD project2 reflection

Changes made after project2A:

No major changed occurred as I finished the implementation of project2B before the due date of 2A and handed in the UML for my actual implementation. The previous UML diagram looked like this:



Difficulties during implementation:

One of the major problem that I came across during the coding time was that I continuously wanted to change the design of the classes such that my code could be neat and tidy, and therefore I wasted a large amount of time that was not of necessity, because changing structure of the project is quite painful and requires a lot of refactoring and debugging to make it work again.

Another big obstacle was definitely the handling of memory leak in Java. In Java although the concept of pointers were replaced by references and they were autonomously controlled by JVM, there could still be cases where there is a memory leak. Because I had previous experience of iOS and Android dev experience before so I have heard of the term “Retain Cycle” quite often but never got to understand it. It means when 2 objects each has a referenced towards the other, the object never gets recycled and stays in the memory. That is exactly the case in my project because I used a lot of dual-references. Fortunately I got to solve the problem after finding out what happened.

Knowledge learned:

One thing I learned from the practice of the project is delegation and separation of concerns. As mentioned above I obviously have a object oriented programming background so I was quite familiar with the concepts. But I never could truly understand their importance until I have experienced the difficulties of handling things with a bad design of classes. It was real pain and the time spent on fixing corresponding problems was very unnecessary. As the functionalities required for the project gets more and more sophisticated is becomes even more crucial that a clean and neat design is usually a better option. A bad design sometimes might achieve a better performance, but it is compromising the convenience and it makes further maintenance harder as things are not easy to change. In order to make everyone’s life easier, we must apply the concept of separation of concerns to our design. When I was doing mobile development, it might not be a big deal because there were not as many design as this subject, everything was pre-defined for you and you just need to use it. However, OOSD truly made me feel the paradigm of object oriented programming languages, how I need to improve and do better, and even as a developer, how to make designs that can make the entire dev team to understand and be more productive.

Things I would change in a similar project:

Definitely I would spend more time on the design rather than actual implementation. It might still be quite difficult because a lot of potential problems are not considered in the design phase. Therefore it is not that easy to determine how the classes would look like when a person has not done any similar project implementations. But I would still attempt to take every single detail into consideration and come up with a structural design that does not require any further change in the implementation phase.