Non Functional Requirements

**Extensibility:** Examples of potential for extension in our code.

*Observer:* The use of the Observer DP allows for loose coupling between the Subject and it’s Observers. It is easy to plug in an extra Observer without harming the other System components. This is due to the fact that all a Subject knows is that it has a list of Observers, and each of the Observers conforms to a simple interface defined in the Observer interface.

*Command: “*Command decouples the object that invokes the operation from the one that knows how to perform it”(1) It is easy to add new Commands as you do not have to change any existing classes. In our project we have a ConcreteCommand which implements the Command interface called ‘PlayerMoveCommand’ that defines a binding between the Player instance and it’s action(‘move’). We initially intended to have another ConcreteCommand called ‘ShineTorchCommand’. If we wanted to implement this ConcreteCommand in the future it would be simple due to the louse coupling between the Player (who carries out the request) and the Client (who invokes it).

**Portability:** Is our framework portable?

“Write once, run anywhere” Java promotes portability. This program was compiled on Java version ‘1.7.0\_15’.

*Java as a language, source code portability:* A given java program should produce identical results regardless of the OS, CPU or Java compiler. The semantics of languages such as C or C++ are not well defined. This semantic looseness means that a block of C code may produce different results depending on the operating system, compiler or CPU. Java is different. It provides more structured semantics while leaving less up to the implementer. Java’s superior portability capabilities is one of the reasons we chose it for our program.

**Performance:** Programming for performance.

Object creation in Java can be expensive. The use of Singleton and Factory Method DP’s as found in our program limit the number(Potentially not correct) of objects of various types and assist with object reuse and reduction of garbage collection

Maintainablity/Readability: talk about polymorphism

**Maintainability/Readability:** Interface for creating enemies(just enter numbers), constants class, logging(system.out.println)/ overriding the default mouse and key listeners.