**2. Software Architecture**

**2.1 Overview**

In order to reduce coupling among subsystems, system will be divided to sustainable subsystems. For architectural style, we have main controller so that decomposed system will be synchronized properly.

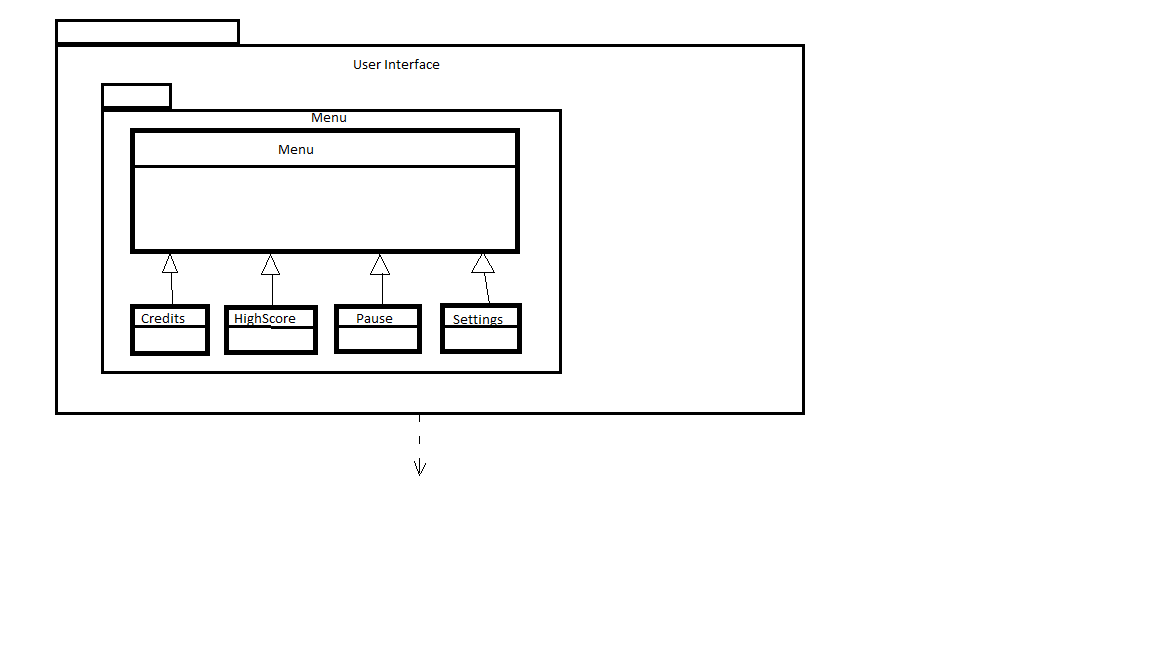
**2.2 Subsystem Decomposition**

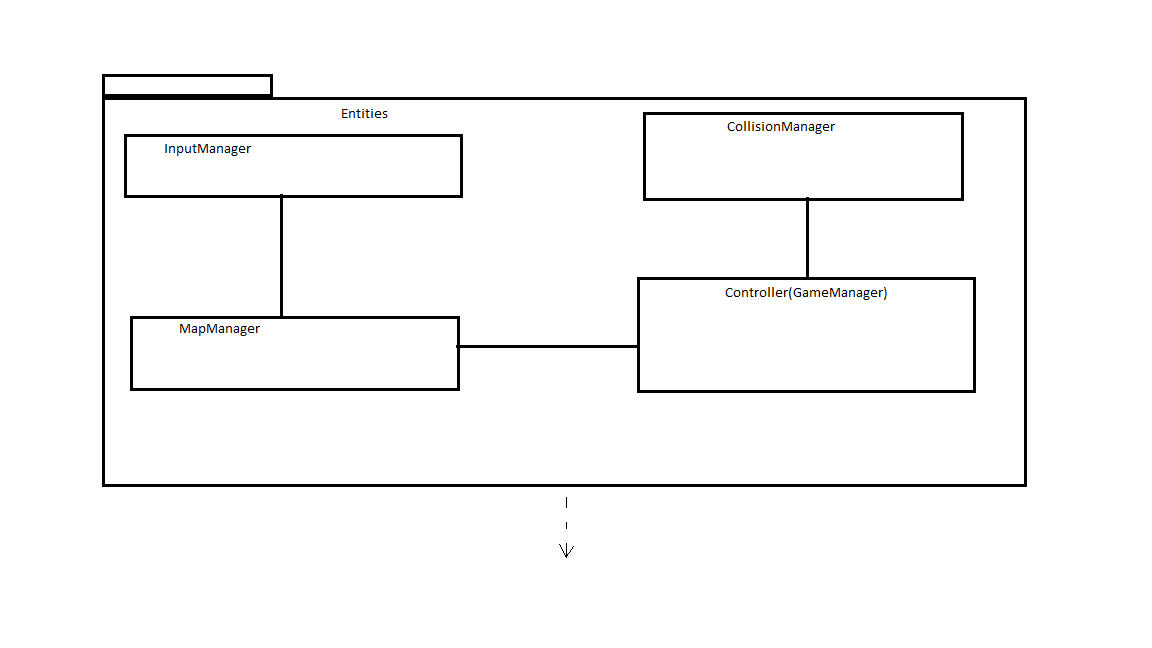
Our system is seperated into synchronized and independent parts for proper architecture and organization. Thus, we will be able to control our system much healthier. Moreover there will be considerable performance increase and ease on modification and extension via this architecture. In this way, both functional and non functional requirements division are playing significant role for high quality software.

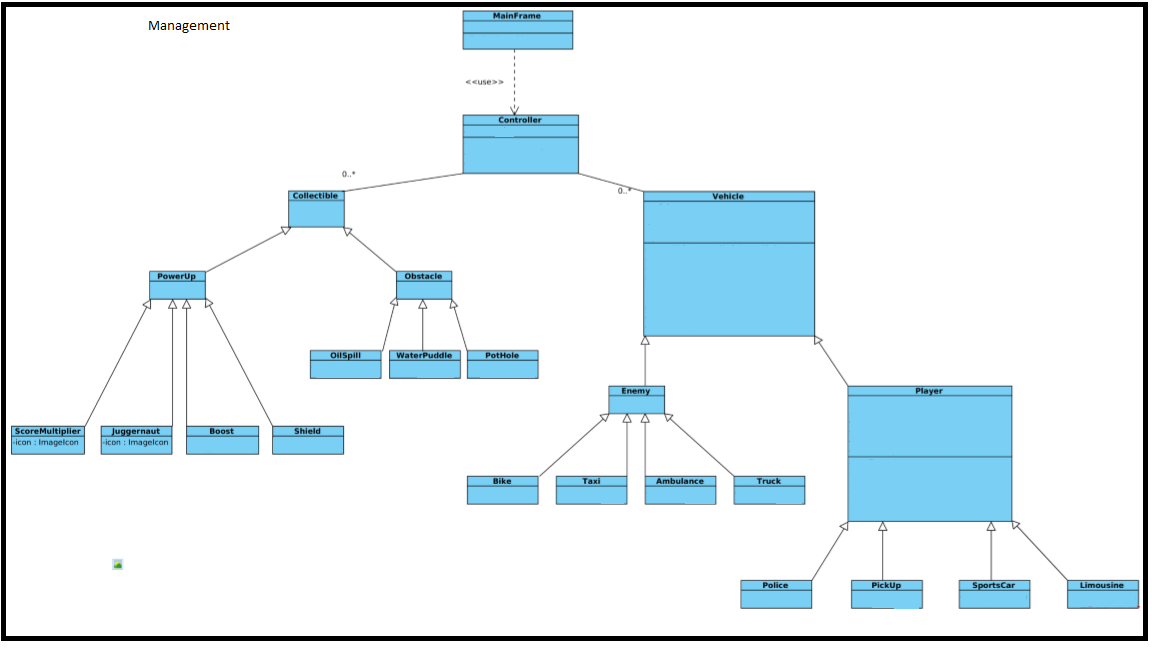
Our subdivided system is shown below figure. Main general divisions, subsystems are User interface, Management and Entities( synchronized with controller ). As it is shown, there is connection between each subsystem. To identify connection between Management and Entity subsystems, controller is emphasized in each system. This provides enhanced synchronization among them. There is many inheritance usage in our systems, which will lead that if any error is occured, only its parent will be responsible from this specific error.

In detail, it is shown that children classes who are performing similar tasks, are gathered for optimization of systems. This organization will ease the subsystems in terms of controlling and modification if needed.

Consequently, subsystems of our project is is investigated to implement our goals effectively and with high quality. High cohesion and loosely coupling are implemented to reach our goals as well. This caused flexible and simple system that can be modified or upgraded easily when needed.



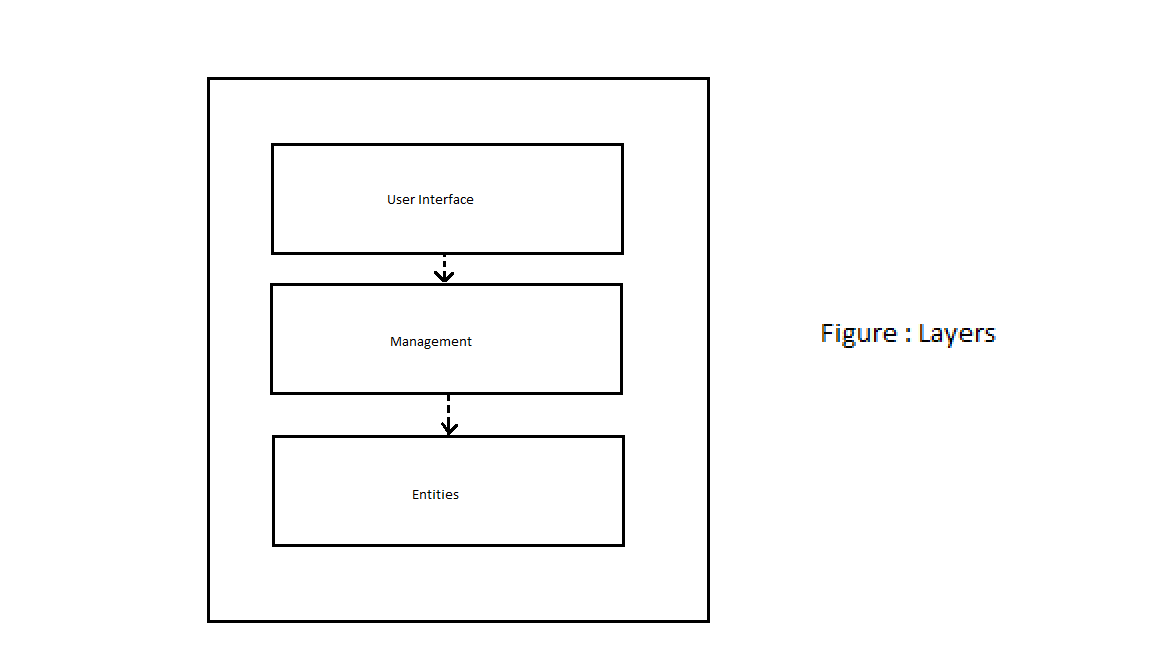




**2.3. Archirectural Styles**

**2.3.1 Layers**

As it is mentioned above, our system is divided into three subsystems which are User interface, Management and Entities. We will use these three subsystems as layers to improve the quality of hieararchy. For instance, User interface layer will be at the top of hierarchy so that other subsystems will not be able to use User interface. On the other hand, Entities layer will be at the bottom of the hiearchy because entity object are brought and used together in our game. The architecture of layers is shown figure below.



**2.3.2 Controller**

In our project design, we will have one main controller as it is implemented in 2.2. This main controller will be able to modify classes which are settings, collectiable(and its children), vehicle(and its children). For example, main user interface will not able to be modified by controller. This hierarchical main controller will be also responsible by starting, pausing game,checking and updating highScores, generating and updating obstacles, enemies and players. By our hierarchy design, if we modify the user interface, this wont effect controller and other subsystems.

**2.4 Hardware / Software Mapping**

We will use Java language during our game implementation. Hence JDK 1.7 or above will be used during software process. For inputs we will have KeyListener and MouseListener so keyboard and Mouse will be required. System requirements will be minimal as we are implementing the game through simple in Java with no complicated game engine. We will be using photoshop cs6 for our designs. Also java platform will be used for designs.

Internet will not be required for our game and we will store highScores in txt file. Thus for storage, our java implementation will read from txt file.

**2.5 Persistant Data Management**

We will store the Project data in the hard disk drive, for highScores our desing will not be through database because the simplicity of our data which are scores and players. Image and model datas will be stored independently so that they will be easily modified.

**2.6 Security**

Since the game is not required internet, it is resictricted in specific pc. Secondly, for data corruption in specific pc, by our hierarchical design, will be much easier to not effecting noncorrupted datas.

**2.7 Boundary Conditions**

**- Initializing the game**

No install will be required. The game will be initialized by single jar file.

**- Termination**

Quit game button in the interface will be provide user to terminate the game. If user wants to quit during game, controller will warn the user if he/she is sure or not. Also standart “x” termination button will exist.

**- Error**

In case of errors which are caused by user’s pc, current player’s at that time game data will be lost.