# *Design Patterns to use in XLPOOLsion*

*The Design Patterns that are expected to be used in this project are the following:*

- MVC (Model View Controller)

  Used in order to facilitate modularity by reducing dependency between the game's graphics, model and internal logic.

- Singleton

  Used in several classes to ensure that they have only one instance at any given time (e.g. NetworkRouter, GameController and GameModel).

- Observer

  CollisionController, in the Controller package, acts as a collision observer (aka listener) for collisions in the game world (This design pattern is implemented by LibGDX - to be more specific, Box2D).

- Object Pool

  An object pool is being used for creation of Bomb and Explosion Models, because they are created and destroyed frequently, but do not usually last very long, thus reducing the overhead of object creation and destruction (In the future, it might also be used to create views so that they can have independent animation "stateTimes"). This was implemented with the aid of LibGDX's Pool container.

- Factory

  To be used to simplify the creation of several elements of the game (Buttons, Textures, etc)

- Factory Method

  To be used for the creation of Views from EntityModels, for example.

- State

  Both client and server applications have their own inner state that changes with events.

- Update Method

  The update method pattern is used to ensure a consistent rate of update for the GameController (mostly physical world) and GameModel (decay of game objects - bombs, explosions).

*The Game Loop pattern is also implemented by LibGDX.*