Testing Strategy

Part I

- **Test board configuration load:** Ensure that the board configuration loads properly
- **Test level configuration load:** Ensure that the level configurations are loaded correctly
- **Test that the level increments when the player beats a level:** Ensure that, when a level is beaten, the game goes to the next level.
- **Test that the lives are tracked properly:** Ensure that the player's number of lives is decremented correctly.
- **Test that the board and squares are drawn properly:** No actual JUnit tests here, but as development occurs, ensure that the board and squares are rendered properly.

Part II

- **Test minion advance:** Ensure that the minions advance at the correct speeds and that they stay on the paths.
- **Test tower fire angle:** Place a minion in the firing range of a tower and ensure that when the tower fires the minion dies.
- **Test tower attack mechanisms:** Ensure that each type of tower only attacks the type of minion that it is capable of killing. (For example, ensure that the AirTower only kills AirUnit minions.)
- **GUI Testing:** No actual JUnit tests here, but as development occurs, ensure that the HUD, towers, and other GUI elements are drawn properly.

Development Strategy

Part I

- Create the classes from the UML with unimplemented functions
- Load board configuration
- Load level configuration
- Game logic (level increment, lives tracking)
- Draw board and squares

Part II

- Allow the player to place towers
- Minions attack on path
- Towers attack minions, implement the various types of attacks
- Implement HUD GUI

Functionality if Time

- Multiplayer over LAN