Requirements

Refactoring

Must haves

- The code must have a separate class for each Powerup.
- The code must have a separate class for each Utility.
- The code must have a separate class for each Weapon.
- The code must have a separate class for each Bubble.
- All the pickups must handle their functionality in their own class, instead of in the Player class.
- Each method must have a maximum of 30 lines.
- The update method of Game must be split up into small blocks.
- The collision handling in the Game update method, must make use of the Quadtree for every collision, instead of also using List<E>.
- All refactored code must be tested.
- The Weapon class must have an instance of Player in the constructor to enforce a composition relationship.

Should haves

- The Resources class should be wrapped in a class that can be instantiated, so the Resources can be mocked during tests.
- The code should enable a Player to be initialized on any spot in a Level.
- The onCollision method in the DefaultCollisionHandler should be refactored so the method contains less than 30 lines.
- The code should have comments for every class and every method (except basic getters and setters).
- Any code ‘smells’ that are encountered during refactoring, should be evaluated and possibly refactored.

Could have

- Old, untested code, could be tested, to increase the line coverage.
- The tests could work on Travis, instead of being skipped.
- The responsibilities of Player could be reduced.
- The GameState class could be refactored.