Galaxy Defenders - Final Specification

**Game Summary:**

Galaxy Defenders is similar to Space Invaders in that the human player controls a Sprite (Rocket Ship) and fights off Aliens by shooting bullets upwards. However, we have added a few twists to the game to make it unique. For example, the Aliens slowly inch toward the player in an attempt to take over the galaxy. If the player is unable to shoot down the Aliens in time, the galaxy is lost and the player loses. We have also added an occasional power-up to differentiate the game from the original Space Invaders. Every fifth shot, the player fires a golden bullet which destroys all Aliens in its path.

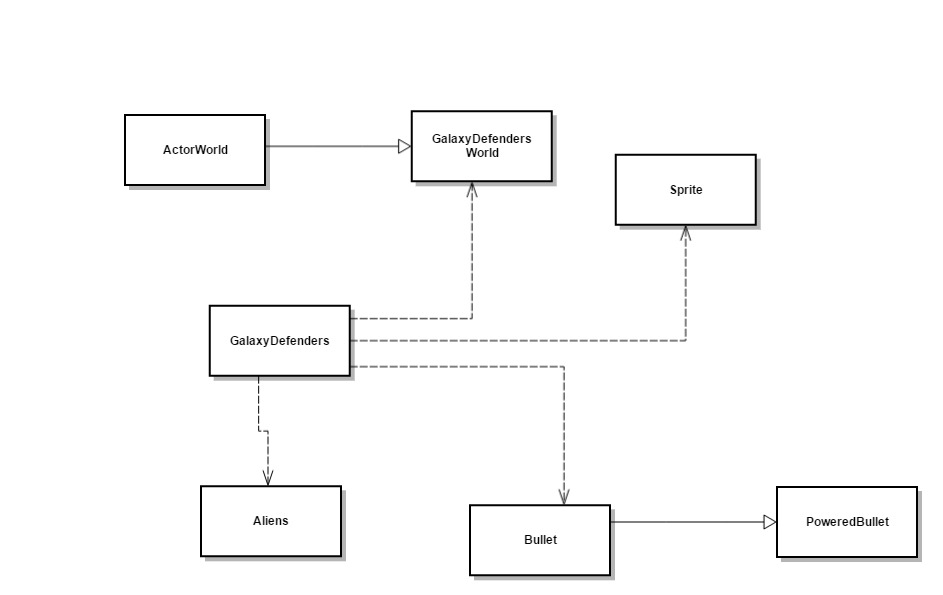
Our design process for Galaxy Defenders consists of four parts. The first part is structural design, which determines which data structures will be used in the program. The second part is object-oriented design, which determines the types of objects to be defined and the classes and interfaces to be written. The third part is detailed design, which determines the fields, constructors, and methods in all the classes. The fourth part is developing a test plan.

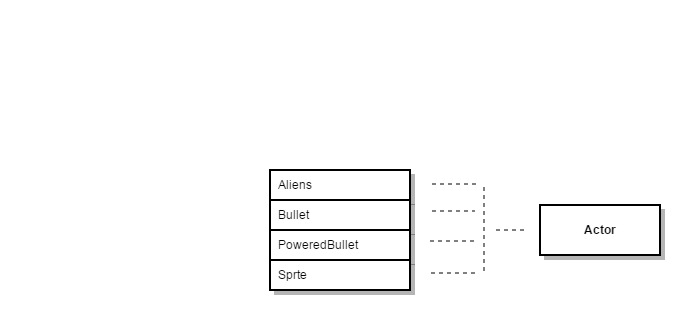


**Structural Design:**

Since Galaxy Defenders uses the design of GridWorld, each Alien is stored in an array as displayed on the player interface. We decided to store the Aliens in a 3-row checkerboard pattern. Every other row and column had an Alien with the first and third rows consisting of 12 Aliens and the second row containing 11. The array of Locations would make it easier for the Aliens to move side-to-side and eventually downward toward the player.

We decided to store each Alien into a LinkedList in order to determine when the game was over. The LinkedList is part of the GalaxyDefendersWorld class in that Aliens are added to the list when they are called into the ActorWorld. When an Alien is within the path of a bullet shot by the player-controlled sprite, the Alien is removed from both the ActorWorld and the LinkedList. The size of the LinkedList would also serve as a counter displayed to the player indicating how many enemies are left on the screen. Once the size of the LinkedList becomes zero, the game is determined to be won and the player will receive a message indicating the result. We decided to use a LinkedList rather than a PriorityQueue because we wanted each Alien to be in order in respect to when they are added and removed. Each Alien has the same priority and therefore, a PriorityQueue is not needed.

**Object-Oriented Design:**



The Aliens, Bullet, PoweredBullet, and Sprite are all of the type Actor.

The GalaxyDefendersWorld is associated with ActorWorld. We based our game off of the GridWorld design, making our Sprites, Aliens, Bullets, and PoweredBullets all of the Actor type. The GalaxyDefenders class holds the main method and calls the Actors from each respective classes when the game begins.

The GalaxyDefendersWorld class contains helper methods for the functionality of the game, which includes a keyPressed method, a contains method, and a get method.

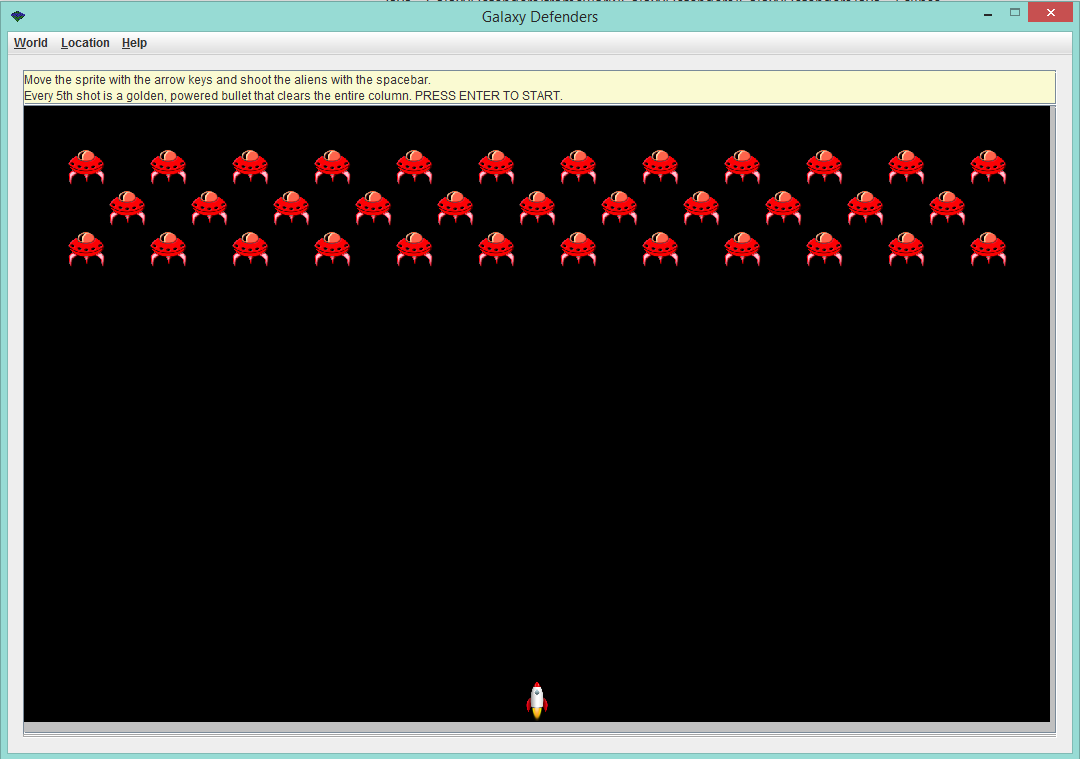
GalaxyDefenders has the main and starts the program, providing instructions in a message box at the top and places the sprite at the bottom of the screen and the aliens at the top of the screen.

The purpose of the Sprite class is to be able to create the images for the respective sprites (aliens, rocketship, bullets).

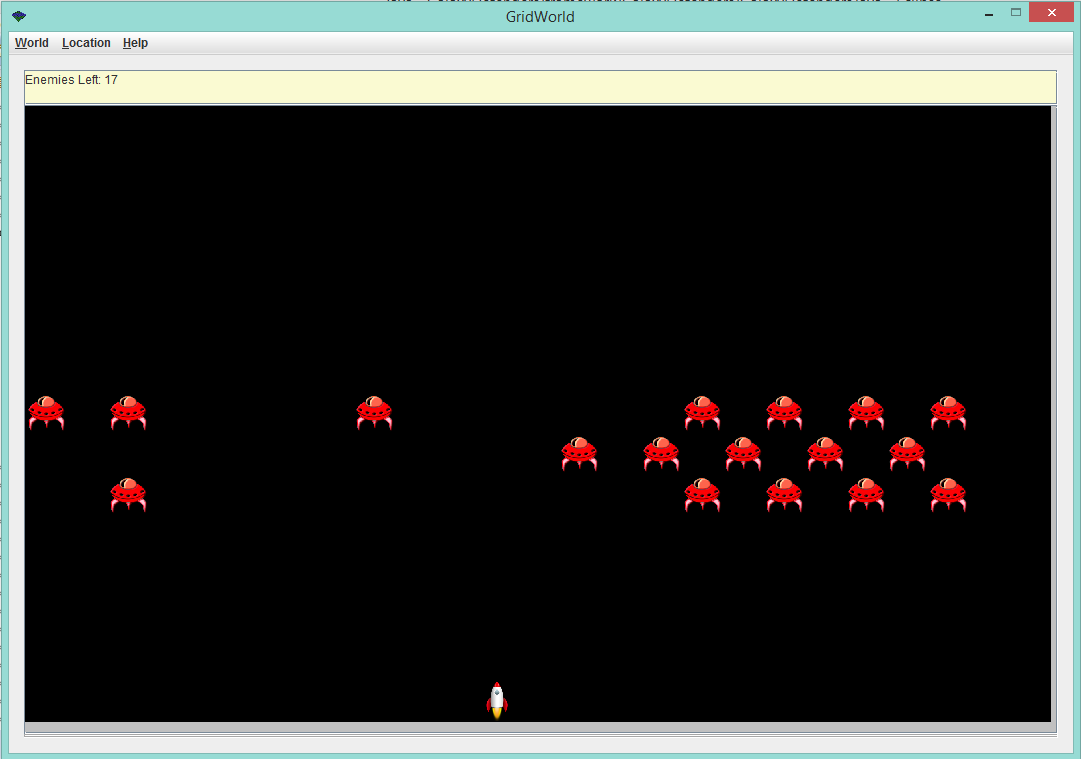
The Aliens class describes the alien movement and uses a timer to determine the speed at which the aliens move.

Bullet gives functionality to the bullets that are shot out by the sprite controlled by the player (rocketship). It detects when the spacebar is pressed, triggering a bullet shot.

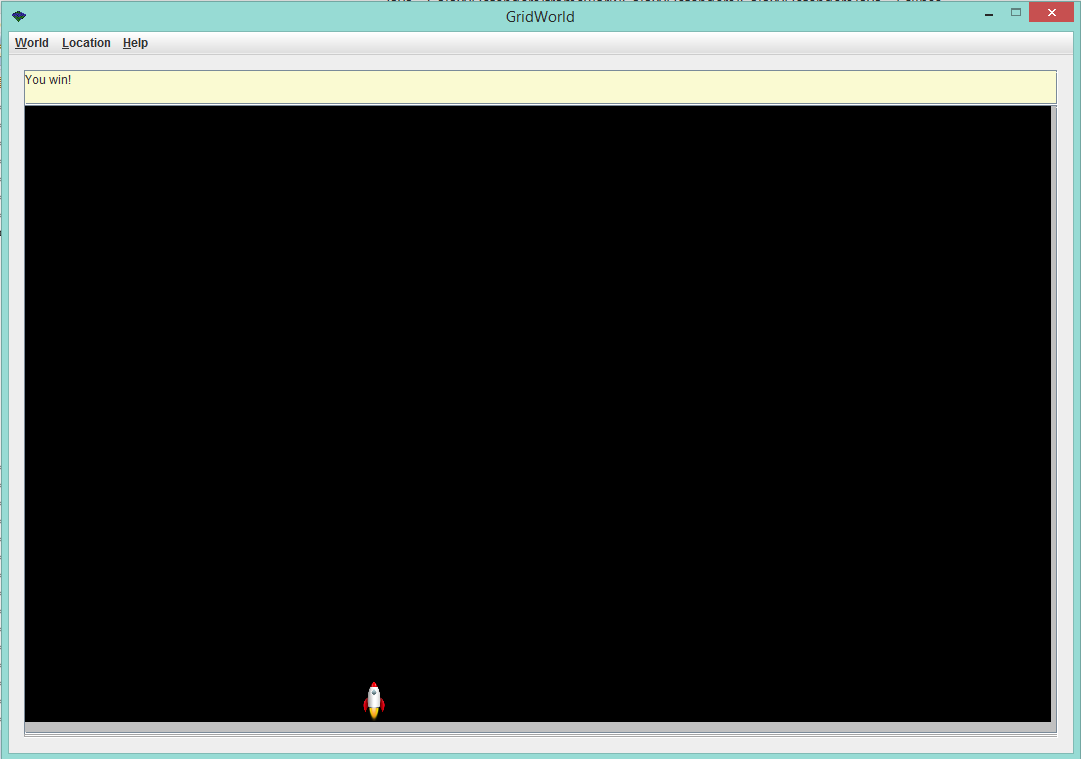
PoweredBullet extends the Bullet class and enhances the capability of the bullet, allowing it to clear an entire column of aliens.



When the game is first opened, the screen will look like the image provided above. The aliens will be in 3 rows near the top of the screen and the rocketship sprite will be at the bottom of the screen. The initial message that is displayed near the top of the frame gives instructions regarding how to play the game and a description about the powered-up bullet

After beginning the game and eliminating an alien, the initial message with the instructions will be replaced with a message that displays the number of aliens that are left on the board. In the sample image above, there are 17 aliens that still need to be eliminated.

If the aliens manage to reach the column directly above the spaceship sprite, then the aliens stop their side to side movement and the message that counts the number of aliens is replaced by a message that says “You lose!”. If for some reason the user keeps on moving the spaceship after the losing message is displayed, the aliens will continue towards the bottom of the screen.



When all of the aliens are eliminated from the screen, the message that displays the number of aliens that are left will be replaced with a message saying “You win!”.

**Detailed Design:**

The detailed specs for the GalaxyDefenders classes have been generated from the Javadoc comments in the source files and are provided in the GalaxyDefendersDocs.zip file.

**Testing:**

We used JUnit to test the methods in each class, along with common actions that the game used several classes to accomplish.

We used indirect methods for many of our checks: for classes that had their own constructors, we initialized the classes and then checked the values of their properties as an assurance that the constructor had executed properly.

For our checks on movement, we used Gridworld’s Location class to compare the characters’ actual locations after moving certain distances with their expected locations. Finally, for our tests of adding characters to the class, we used our contains() method to verify the presence of the characters on the grid. Seeing as how the contains() method was previously checked in the test of the GalaxyDefendersWorld class, we decided that it was a viable method to use in our tests.