Richard Sear
CSCI 2113
Project 2 - Alien Attack
Design Documentation: Points of Interest

CLASS: AlienAttack
Contains main method and starts the game by spawning a separate thread for it to run in. Starts game by creating a new AlienAttackMenu object and showing it.

CLASS: AlienAttackMenu (extends JFrame)
Generally speaking, this class has three jobs: tracking high scores, choosing a player, and instantiating an AlienAttackFrame when the user decides to start the game. During its construction, the AlienAttackMenu creates an instance of each of the three player designs so the user can see what each looks like. It also creates a JPanel for the leaderboard, but this is hidden until the user clicks the “View high scores” button. High scores are stored in a txt file called “hiscores.txt”, each score/name pair on a separate line and separated by exactly 1 space. As a part of the debugging process, I chose to have the scores stored in the txt file until they’re needed (i.e. AlienAttackMenu does have an ArrayList to store the scores, but it isn’t populated unless the user is viewing the high scores or a game just ended). When a game ends, the AlienAttackMenu receives a score from its AlienAttackFrame, and then determines whether the score is in the top 10. If it is, it prompts the user for a username, and removes spaces from this (so that it doesn’t mess up the format of hiscores.txt) and restricts it to 45 characters (to keep the leaderboard GUI nicely formatted).

Inner classes:

- **StartListener (implements ActionListener)** - ActionListener attached to the start button to begin the game when it’s clicked, as well as pass the selected player choice into the game
- **HiScoreListener (implements ActionListener)** - ActionListener attached to the “view high scores” button to hide the main menu and show the leaderboard panel when it’s clicked; this class handles reading the file into the ArrayList and printing the contents to the text areas in the leaderboard panel
- **PlayerSelector (implements MouseListener)** - MouseListener attached to each of the Player objects on the main menu to determine which one the user clicks and frame it to show that it’s selected

CLASS: HiScoreNode (implements Comparable)
A custom node for the high scores ArrayList, so that the list can hold both the score and the username associated with it. Ordinarily, this kind of pairing would be ideal for some kind of map, but a map wouldn’t allow for duplicates of a username. The node is comparable so that it can be sorted using Java’s Collections.sort method; it sorts first numerically and if the two scores are equal, then alphabetically by username. HiScoreNode isn’t an inner class of AlienAttackMenu

because in an earlier version of the game, a different game component handled the score system.

CLASS: AlienAttackFrame (extends JFrame)
During gameplay, this frame is the overarching container for the game elements. It contains the 900x900 graphics panel (where all the game action happens), a score display, and a panel with start, pause, and end buttons. It also contains the GameEngine inner class (see below). On construction, AlienAttackFrame reads AlienAttack.properties to find the custom game options. If this config file can't be accessed, there is some problem with the way the Java files are laid out or with the way the properties file is formatted, so an exception during this phase triggers a system exit. Player movement is implemented as a Queue, since the player can only move when the game clock triggers. The left and right arrow keys add commands to this queue.

Inner classes:

- **ControlsListener (implements KeyListener)** - a KeyListener attached to the content pane of the AlienAttackFrame that adds user commands for player movement to the queue of commands
- **OptionsListener (implements ActionListener)** - an ActionListener attached to the start, pause, and end buttons. Start and Pause aren't clickable at the same time, since they perform opposite tasks: Pause stops the game timer (therefore stopping the game engine from triggering movement), and Start starts it again. End stops everything, clears all aliens from the screen, and passes the current score back to the AlienAttackMenu where it can be compared to scores on the leaderboard
- **GameEngine (implements ActionListener)** - an ActionListener that is triggered by the game timer each game cycle (a custom time frame interval in milliseconds). The reason the GameEngine isn't implemented as a separate class rather than an inner class is because there are a number of times when it needs to quickly access the attributes of the AlienAttackFrame. It handles the following game actions:
    - Player movement - the next user command is taken from the queue and the player is either moved right/left, or a missile is fired
    - Alien movement - all aliens are moved downwards (the distance depends on their size) and more randomly generated aliens are spawned at the top of the screen (see Alien class)
    - Missile movement - all missiles are moved upwards and checked for collisions
    - Game difficulty check - if enough cycles have passed to where it's time to increase the game difficulty (a custom option), the AlienGraphicsPanel is told to increase the boundaries on the random spawning by the amount specified in the custom options file
    - Score update - retrieves the score from AlienGraphicsPanel and updates the score box
    - Is the player hit? - requests AlienGraphicsPanel to check all aliens on screen, and if there's an overlap (indicated by a boolean return), the screen flashes red and the player shrinks. If the player is already at its smallest size, then the game

is now over. A score and elapsed time is displayed on a dialog box, and then the end button action is triggered

The frame is repainted each cycle.

CLASS: AlienGraphicsPanel (extends JPanel)

Contains a number of static variables read from the custom options file. These are static because they are the same during any game being played, unless the game is closed and the options are changed. This panel has a null layout because all elements are moved by setting location and no automatic layout is necessary. AlienGraphicsPanel keeps track of all aliens and missiles on screen using ArrayLists. Each time it is told to by the GameEngine, AlienGraphicsPanel moves all on-screen aliens downward depending on their size (customizable option) and all on-screen missiles upwards. It also contains a random number generator that generates random integers within a range; this is used when the graphics panel is told to spawn more aliens. All the aliens' sizes, positions, and quantity are randomly generated, depending on ranges specified by the graphics panel's variables. AlienGraphicsPanel also checks (when requested by the GameEngine) to see whether the player is overlapping any aliens on screen and returns true if this is the case. When an alien moves completely offscreen or collides with a missile, it is removed from both the ArrayList and the actual JPanel, and the score is incremented by the amount that the alien was worth (custom option). Lastly, the graphics panel (when requested to by the GameEngine) clears all the aliens and missiles from the screen (ArrayList is cleared and JPanel is wiped except for the player). This happens when the player gets hit or when the game is over/ended.

CLASS: Alien (extends JComponent)

A class for the aliens on screen. This has a paintComponent method that draws a scaled polygon based on the X and Y coordinate arrays that are its member variables. The arrays are coordinates for a 30x30 alien, and are scaled up if the alien is 60x60 or 90x90. Additionally, each alien knows how big its game screen is, so after moving downward, it is able to return a boolean value to the graphics panel to tell whether it is fully off the screen.

ABSTRACT CLASS: Player (extends JComponent)

Since the player has three possible appearances, this class contains all the functionality of a player object that any instance of the player needs to be able to do. The player knows how large the game screen is, so it always spawns itself directly in the middle of the screen and restricts its own movement so that it does not move off screen. The movePlayerLeft and movePlayerRight methods handle these calculations and are called when the GameEngine determines which direction (if any) the player needs to move in the current cycle.

CLASSES Players1-3 (extends Player)

These are the actual implementations of the Player class and differ in appearance. Their component shapes are specified as member variables. Since the player shrinks when it is hit, the paintComponent method calculates a scale factor for all the coordinates of the shapes. Like

the Alien class, the coordinates of all the different painted components are for a 30x30 player, and are scaled up when the player is larger.

The concept art for the players and aliens was drawn up by my sister over Thanksgiving Break because, and I quote: “Rick, your shapes are too boring.” Hence the reason for the elaborate arrays of coordinates and sizes in these classes.

CLASS: Missile (extends JComponent)
The Missile class does very little on its own; its main purpose is to provide a nice little visual for projectiles fired from the Player. Its constructor takes a Point object as an argument, which is the starting location for the upper-left corner. Since a missile is only one size, it is unnecessary for this class to do much else -- movement is handled externally by the AlienGraphicsPanel using methods inherited from JComponent.