Christiaan Jacobs
20111703

**1.** The main class that needs to be compiled and run is **StellarCrush.java**.

**Instructions:**

Set class path: Path where main class is stored Ex: cd H:\RW E214\DrJava\Project\Final Project
Compile: javac -cp ".;StdLib/stdlib.jar" StellarCrush.java
Run: java -cp ".;StdLib/stdlib.jar" StellarCrush

**2.** My program makes use of an interface called IViewPort (provided in skeleton) which is implemented in my PlayerObject class. Variable "holder" of type IViewPort is declared in Camera class and assigned with the argument of type IViewPort received in the constructer argument of the class. This variable is used to obtain data from player object to render first person view.

My program also make use of an additional interface called Viewable which is implemented by GameObject class, Enemy class and Bullet class. I add all the objects from these respective classes to a collection of type Viewable. I then use this collection to sort all the objects according to distance from the player.

**3.** My program contains two classes called GameObject and PlayerObject where PlayerObeject extends GameObject. Both classes contain a function called draw. This ensures that whenever an object of type GameObject which is also of type PlayerObject calls the draw function the function in the PlayerObject class gets excecuted, meanig the draw function in the GameObject class gets override. This is necessary because the player needs to be drawn differently than the other GameObjects. The draw function in PlayerObject sill makes use of the variables “r” and “radius” from GameObject class (polymorphism).

**4. Additional work:**

- Adding additional objects to universe with class Enemy.java where they are constructed. The enemies shoot at a constant speed towards the player by updating the facing direction of the enemy according to the player position.
- Extensive in-game interactions: Both player and enemies are provided with a set of bullets to shoot at each other. Bullets get removed as they hit other objects or move out of screen. Player loses mass with each bullet that he shoots. Needs to “reload” by absorbing planets smaller than himself.
- Multiple in-game levels: New level spawns when player kills all enemies. When player reaches a new level, a new random universe are generated with an extra enemy than in the previous round.

- Reading/ Writing to files: Saves the game state at the beginning of each level by writing all the data into text files. This data can be used to load the previous game state when player is dead or program has been closed.

- Extensive and interactive menu: Settings screen - Show game controls.
  - Option to change number of initial bodies.
  - Option to set initial health of player.

  Extras screen - Suppy gameplay information.
  - Show list of highscores created by previous players.

  Option to load previous game state from text files.

- Health points: Health added to player. Player loses health when hit by enemy. Game over screen appears when player health equals zero.

- Keeping track and displaying high scores: File contains previous high scores. Reads in highest score at the beginning of the game and compare it to the score obtained at the end of the game. If a new high score has been achieved the player should enter his name and it will be added to the list of scores. Only showing the top 10 scores in descending order in Extras menu.

- Sound - Background music for better game experience.

  - Sound effects to make game more realistic: shooting, colliding, breaking other objects, hit by enemy, reloading.

- Prevent object from moving out of universe by rebounding them back into the world.

- Added images (including self-made), showing game info on screen, added crosshair in first person view.

- Making program more robust and realistic: Checks that objects do not spawn inside one another when generating object with random positions to prevent unrealistic behavior. Place the "camera" in front of the player to view from instead of his center point.

**5. Deviation from skeleton:** None

## 6 & 7. Additional libraries:

Added the newest standard library “stdlib.jar” in a subfolder named StdLib in class path. This was necessary to prevent my program from lagging at certain times. I experimented with the class files inside the standard library provided on the H drive and the latest one and concluded that the StdAudio in the provided library caused the lagging. The Draw class and StdDraw class in the newest library also gave a slightly better frame rate. Link for latest standard library: http://introcs.cs.princeton.edu/java/stdlib/

## 8. Class diagram