1 - Code Clean Up

1.1

- Logger changed “if(instance == null)” with new Logger() in private static final, thus we can skip the null check and avoid the if-statement.
- Instantiate the background image in the constructor without checking for null, since we do not want to have a case where the image background is null.
- Remove the if-statement of the Launch class, since this is not a usefull function for the method, it was only used to log a value to the console as a quck ‘test’.
- PowerupFactory: use boolean operators with the conditions as return value instead of the if-statements. Some of the operations were not necessary to use in the if/else-statements.
- Use a NullComponent, which is a constructor with null values. This is used to avoid using the null check.
- Use boolean algebra to refactor unnecessary if-statement:

  ```
  if(A == null){
          return false;
  }
  return A.method(B);

  refactored:
  return (A != null) && A.method(B);
  ```
- Refactor by using polymorphism. SinglePlayState and MultiPlayState extend the abstact DefaultPlayState class. We let the SinglePlayState and MultiPlayState use the same method in the DefaultPlayState class to avoid code duplication(thus less if-statements): updateAsteroid(List<Asteroid> asteroids, Player player) and updatePowerups(List<PowerUp> powerUps, Player player).
- Use a NullPowerUp, which is a constructor with null values. This is used to avoid using the null check.

2 - New Feature

2.1

Requirements difficulty:

Must Haves

- The game shall have two difficulties: medium, hard
- The game shall set the default difficulty to medium
- The player will be able to select a difficulty in the menu using a difficulty selector
- The game shall set the asteroids speed to a higher value depending on the difficulty
- The game shall spawn more asteroids when the difficulty is higher
- The game shall spawn less powerups when the difficulty is higher
- The game shall give more score depending on the difficulty level (this feature is for when we may implement high scores so the easy mode wouldn’t be best for getting higher scores)

Should Haves

- The game shall have an easy difficulty
- The game shall display the difficulty in the top right corner

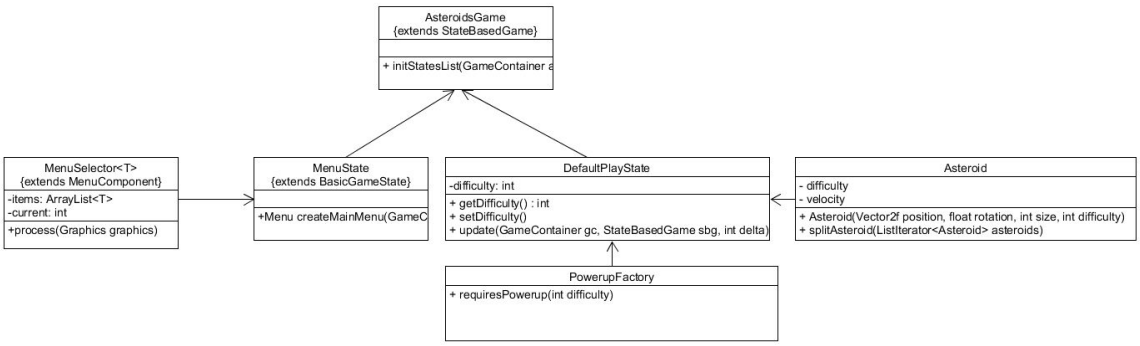
Could Haves

- The player's spaceships shall be harder to control when the difficulty is higher
- The player's bullets shall move less far the higher the difficulty

Would Haves

- The game shall provide the option to change the difficulty during a game
- The game shall change the background music according to the difficulty

2.2



DefaultPlayState

BasicGameState

SinglePlayState, MultiPlayState

Get and Set difficulty	Difficulty element (integer)

Asteroid

ExplodableEntity

-

Spawn and movement according to difficulty	Difficulty element (integer)

PowerUpFactory

-

-

Spawn according to difficulty	Difficulty element (integer)

3 - Code Review

6 packages:

- classes
- factories
- launcher
- spaceships
- states
- updaters

4 grading criteria

- (1) quality: OOP principles
- readability
  - (2) formatting: language conventions
  - (3) naming: variables, methods, classses
  - (4) comments: javadoc, extra comments

For all review commentary a number and a plus or minus will be added between brackets at the end to indicate the type and good/bad. Before all commentary square brackets with line numbers will be placed.

All classes

- Missing javadoc (4-)

classes:

- Asteroid
  - [12-13] Extra comments added to explain velocities (4+)
  - [47-49] return null (1-)
  - [50-54] messy construction of Asteroid objects (1-)
  - [66-73] didn't recycle '(points[i] * Math.pow(2, size - 1)' , but calculated it twice (1-)
  - [84-87] strategy used to check boundaries (1+)
- BoundaryStrategy
  - [3] unused ArrayList import (2-)
- DoomPlayer
  - [18-56] use enum for keys (2-)
- Doomship
  - the 'Doom' classes are all located in the classes package, shouldn't the ship be in the spaceships package
  - [20] magic number '3f'(1-)
- DoomStrategy
  - [all lines] Nice separate strategy class (1+)
  - [16] 'asteriods.getCenterX/Y()' called twice (2-)
- Entity
  - [56-69] why is there a 'checkBoundaries()' method in this class when you already have a strategy for this (1-)

    - [89-129] ‘points[i]’, ‘getCenterX/Y()’ called many times(2-)
- Laserball
    - [29] ‘createPoints()’ misleading name (3-)
    - [14] rename ‘travel’ to ‘traveledDistance’ (3-)
    - [14] init ‘travel’ when you create a new object(2-)
    - [94] better option to create ‘updateTravelDistance()’ method where you add the distance to the ‘traveledDistance’
- Laserbeam
    - [8] init ‘timer’ when create new object (2-)
    - [21] ‘else’ not necessary (1-)
- Logger
    - [all lines] comments and javadoc for all methods (4+)
- NormalStrategy
    - [13-26] ‘getCenterX/Y()’ called twice (2-)
- Player
    - [60-65] magic number ‘3f’ (2-)
    - [84] ‘this.s’ can be more discriptive (3-)
- Powerup
    - [72-116] ‘createPoints()’ method is same as in ‘Laseball’ class, maybe locate it in ‘Entity’

factories:

- AsteroidFactory
    - [15] ‘aList’ not discriptive enough (3-)
    - [51-52] magic number ‘2f’
    - [57-116] the ‘create’ methods have four lines in common maybe split the methods into a generic create method and a left/right/top/bottom separate method.

launcher:

- Game
    - [45-49] commented out code (2-)
- Launcher
    - [14] ‘Assteroids’ (1-)

spaceships:

The decorator design pattern is used in this package. The base class is the SpaceShip, extended by PowerShip and DoomShip. The PowerShip class is extended by the BeamShip, FreezeShip, and InvuShip. These represent the ship when it has picked up a powerup. The DoomShip is used in the the Doom game mode.

- BeamShip
    - [all lines] the cleanest and most understandable class so far (1+)
- FreezeShip

    - [all lines] also good, nothing to add (1+)
- InvuShip
    - [all lines] nothing to add (1+)
- PowerShip
    - [10] use more discriptive name instead of 's' (3-)
- SpaceShip
    - [68-77] 'ACCELERATION_RATE * delta / Game.deltaSecond' can be assigned to a variable (2-)
    - [83-101] 'DECELERATION_RATE * delta / Game.deltaSecond' can be assigned to a variable, looks neater (2-)
    - [152] dutch comment (4-)
    - [182-193] magic numbers (2-)
    - [196-202] commented out code (2-)
    - [all lines] just press CTRL+SHIFT+F sometimes in Eclipse
- SpaceShipTimers
    - [all lines] It is neat to hide the timers in a separate class, but I don't think it is necessary in this case. Creating a class for just adding a delta to two ints seems a bit much. That's just my opinion though.

states:
- GameOver
    - [18]"//variables" unnecessary comment(4-)
    - [22-52]inconsisten indention in methods (1-)
    - [24-32]hardcoded strings, should use variables (3-)
- GamePaused
    - [67-77] extra comments added to explain lines (4+)
    - [69-81] getWidth is called twice (2-)
    - [21] "//variables" unnecessary comment(4-)
    - [33-81] hardcoded strings, should use variables (3-)
- MainMenu
    - [34-137]hardcoded strings, should use variables (3-)
    - [20-135] extra comments added to explain lines (4+)
    - [82-136] getWidth is called multiple times (2-)
    - [82-136] getHeight is called multiple times (2-)
    - [82-136] calculations with getWidth and getHeight in if-statements should be refactored into methods to enable code re-use (2-)
- PlayDoom
    - [all lines] nothing to add (1+)
- PlayNormal
    - [36-133] extra comments added to explain lines (4+)
- Transition
    - [38-47] commented out code (2-)
    - [21] "//variables" unnecessary comment(4-)

updaters:

- AsteroidUpdater
  - [45-49] commented out import (2-)
  - inconsistent javadoc usage (4-)
- CollisionUpdater
  - [50-80] comments locations are inconsistent (4-)
  - [42-103] multiple nested for loops, consider alternative patterns for increasing efficiency (1-)
  - [22-23] inconsistent naming in variables pList and powerList (3-)
- DoomSpawner
  - [all lines] nothing to add (1+)
- LaserballUpdater
  - [all lines] inconsistent javadoc usage (4-)
- LaserbeamUpdater
  - [all lines] inconsistent javadoc usage (4-)
- PlayerUpdater
  - [22-23] inconsistent naming in variables pList and playersLives (3-)
  - [36-39]hardcoded strings, should use variables (3-)
  - [49-51] inconsistent spacing in rest of class
- PowerupUpdater
  - [all lines] nothing to add (1+)
- Spawner
  - [12-14] inconsistent naming in variables powerList and aList (3-)

Conclusion

The project has implemented certain design patterns very well, but on some occasions code is not reused. Try to look at the comments provided above and see where your main problem is. If possible solve this problem using an appropriate design pattern.

FINAL GRADE: 6