**International University**

**Object Oriented Programming**

**Project: Bomberman**

**Submitted by**

Lý Hoàng Vũ – ITITIU18260

Lưu Trung Đức – ITDSIU20003

Đỗ Lê Phương Nhi – ITDSIU20075

Châu Khắc Đình Phong – ITITIU20213

Nguyễn Ảnh Đô – ITDSIU20005

Lab Section: Lab 7

Course Instructor: Mr. Lam

Table of Contents

[**Introduction** 3](#_Toc123311219)

[**Gameplay** 3](#_Toc123311220)

[**Game objects:** 3](#_Toc123311221)

[**Game Interface:** 5](#_Toc123311222)

[**1.** **Starting screen, Stage Selection, character selection screen.** 5](#_Toc123311223)

[**2.** **Bomber and Enemy** 5](#_Toc123311224)

[**In-game screen** 6](#_Toc123311225)

[**Power-Up** 6](#_Toc123311226)

[**Architecture** 8](#_Toc123311227)

[**1.Screen** 8](#_Toc123311228)

[**2. Player** 9](#_Toc123311229)

[**3. Enemy** 10](#_Toc123311230)

[**4. Item** 11](#_Toc123311231)

[**Conclusion** 12](#_Toc123311232)

Icon

Description automatically generated with medium confidence

# **Introduction**

Bomberman is a strategic, maze-based game that has been popular for a long time. Our team decides to remake this well-known game for the OOP course to familiarize ourselves with OOP concepts and gain more experience in using the library LibGDX – a framework for game development written in Java.

# **Gameplay**

Like the original version, the player must place bombs to destroy all the enemies. To make things easier and more interesting, the player can absorb PowerUps (like moving faster, range explosion wider, placing more bombs at a time, …). The game ends when all the enemies are destroyed (win) or the bomber is out of health (lose).

*Control:*

Press W/S/A/D to move UP/DOWN/LEFT/RIGHT

Press E to place a bomb

*Rule:*

* If the bomber collides with the enemy or is in range of the bomb explosion, then the bomber loses 1 health.
* The initial bomber’s health is 3.
* If the bomber’s health is 0, the game ends.
* Bomber and enemy can’t go through the obstacle.
* There are 2 types of obstacles: solid and soft. The soft is destroyed when it collides with the explosion, while the solid can’t be destroyed.
* There are some PowerUps during the gameplay, the bomber can easily get the PowerUps just by passing it.

# **Game objects:**

**ScreenManager class:** designed based on the singleton structure, the implication is that the number of screens can only be one.

**Abstract class Enemy and PowerUps:** Those abstract classes are like a blueprint that allows us to create different forms of Enemies and PowerUps and help us in improving the extensibility.

**Abstract class AbstractScreen:** Provide a few common methods between each screen, allowing for extensibility without scarifying functionalities.

**StageSelectScreen.java:** we have 4 stages, this class gives 4 options to choose.

**Item.java:** an abstract class for Item. An item must be defined by height, width, and position (x, y coordinate). It has functions to draw, move, and disappear.

**Stage:**

* **GameStage.java:** this class manages everything in the game. It defines the stage background, status bar, and border size and contains the enemy, solid, soft, and powerups list.
* **HUD.java:** shows the stats of the game such as the bomber’s health, speed, explosion range, and the number of bombs that can be placed at the same time in the head of the screen.
* **Soft.java:** extends from Items class Soft can be destroyed by the bomb explosion.
* **Solid.java:** extends from Item class, Solid can’t be destroyed.
* **SpawnSoft.java:** spawns softs.
* **SpawnSolid.java:** spawns solids

**Player:**

* **Player.java:** the main object that contains everything about the Bomber like moving, placing bombs, and dead. It also tells us the current stats about the player such as speed, remaining bombs, and remaining health.
* **PlayerInput.java:** to receive the input from the player’s device and implement the action according to the key code.
* **PlayerAnimation.java:** render every bomber’s action into the screen
* **Explosion.java:** to destroy things after a moment when the Bomb is placed.
* **Bomb.java:** extends from the Items class, used to place bombs.

**Enemy:**

* **Enemy.java:** abstract class, that contains all the general properties of an enemy: speed, health, position, etc. Function death() decrease the enemy’s health when it collides with the explosion. This class used to be a parent class of all kinds of enemies, different enemies have different properties, and we will discuss them below.
* **Alien.java:** Extends from Enemy.java, expected to move live longer than normal
* **NinjaBlue.java:** Extends from Enemy.java, expected to move faster than normal.
* **Caveman.java:** Extends from Enemy.java, expected to live longer than normal.
* **EnemyAnimation.java:** This class renders actions of the enemy into the screen, when an enemy moves and dies.
* **SpawnEnemies.java:** to spawn the enemy at the chosen position, we place the enemy by hand through this class.

**PowerUps:**

* **PowerUps.java:** an abstract class that contains basic information like position, border width, and border height. When the Bomber passes it, function del() makes the PowerUps disappear, and function execute() applies a new ability to the Bomber.
* **FireUps.java:** to increase the range of explosions.
* **IncreaseBomb.java:** to increase the number of bombs that can be placed at a time.
* **IncreaseSpeed.java:** to increase the speed of the Bomber.
* **SpawnPowerUps.java:** to place the PowerUps somewhere in the map.

# **Game Interface:**

1. **Starting screen, Stage Selection, character selection screen.**

Graphical user interface, application

Description automatically generated

1. **Bomber and Enemy**

A picture containing text, clipart

Description automatically generated

**Bomber**

**Enemy**

A picture containing text, clipart

Description automatically generated

**A picture containing text, clipart

Description automatically generated**

Qr code

Description automatically generated

A picture containing text, first-aid kit, clipart

Description automatically generated

Skunk

Blue Ninja

Caveman

Alien

## **In-game screen**

A screenshot of a video game

Description automatically generated

## **Power-Up**

![A picture containing text, colorful, clipart

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAAAABABAMAAABYR2ztAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAnUExURdAAADCgWBgYGHAYAGiIoPj4+JC48ABg+ChAMPjIAPh4APgoAFBoeEtBYh8AAAAJcEhZcwAADsMAAA7DAcdvqGQAAACcSURBVEjHzZRLFcQwCEVjIRawMBawEAtYwAIWaiEWaqGiZl4Z+lnMsjB39Q7cBYdD0v6DfkD0ukDUe4XAPIYIs6pZhUC0fJhzXesEKLpTJ1Tvgb78HPJxgchMZAwcC8gXoi1ytgGUPEFVBGti9iayWa6AiGOJcuQ8wc9128xQvudcAePN6ULky0U9LrjipXvOFPz5nl9J5CyhmNbeN8a8MK6HVyMAAAAASUVORK5CYII=)

Accelerator: Speed Up BomberAccelerator: Speed Up Bomber

![A picture containing text, clipart

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAAAABABAMAAABYR2ztAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAbUExURdAAADCgWBgYGChAMFBoePj4+PgoAGiIoJC48KuCJa0AAAAJcEhZcwAADsMAAA7DAcdvqGQAAACaSURBVEjHzZXLEcAgCAVtwRZogRZogRbSAi1YdiTEwXyueWYPzujbA4O/8g/qCU3ErFakEMvcIRLxUXXbiHBChNIhMlP12GwqEiIwtxaLqq1lu1BCFOiBC9bxVnmzzkZBBBEzj2ObMsYKrsjBiFYICfNrkR8LcWCfMVJIhXmNMC5vMuLb7f5cGApNIIXrQ54xVnh+KAFKWEwpO0H1IRAmTMWCAAAAAElFTkSuQmCC)

Bomb up: Increase number of Bomb can be placed at a time

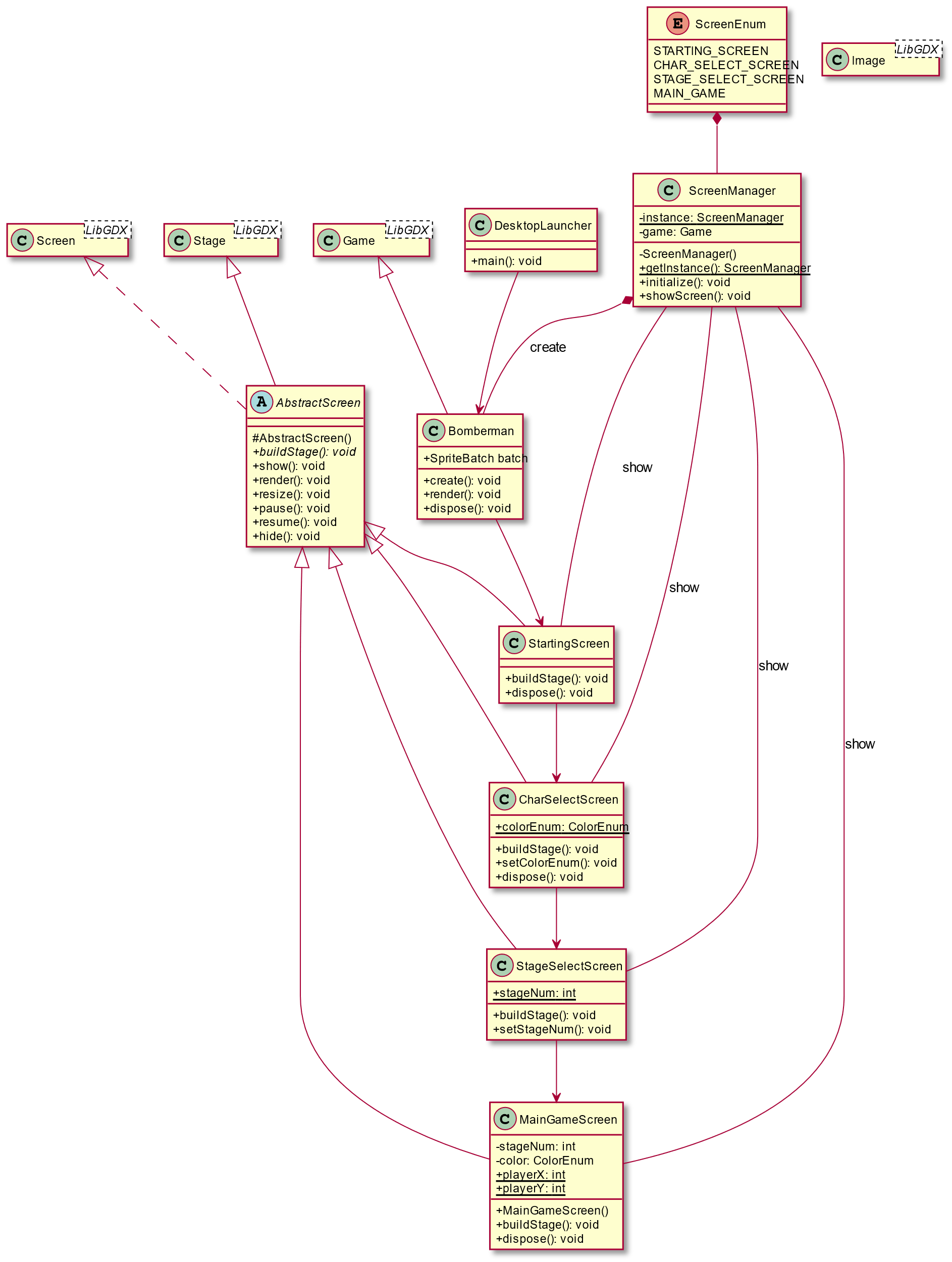
![A picture containing text, clipart

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAAAABABAMAAABYR2ztAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAYUExURdAAADCgWHAYAPh4APjIAPj4+BgYGPgoAB1WaC8AAAAJcEhZcwAADsMAAA7DAcdvqGQAAAB9SURBVEjHzZRBEcAwCARjoRawEAtYqAUsxH5hrinEwNF9MAm3PxLGP7gKImflCyJziqMaNc5cAa2IVFHZQoS3owWuUOOU2EK0zbKyBQypW0CjXzgfC1/Ah7GPHb8LhCaEshwz3LgCPg6UfULMFLDC6pAQc4UtgVzvLKGZMR5+cjj4b6DYugAAAABJRU5ErkJggg==)

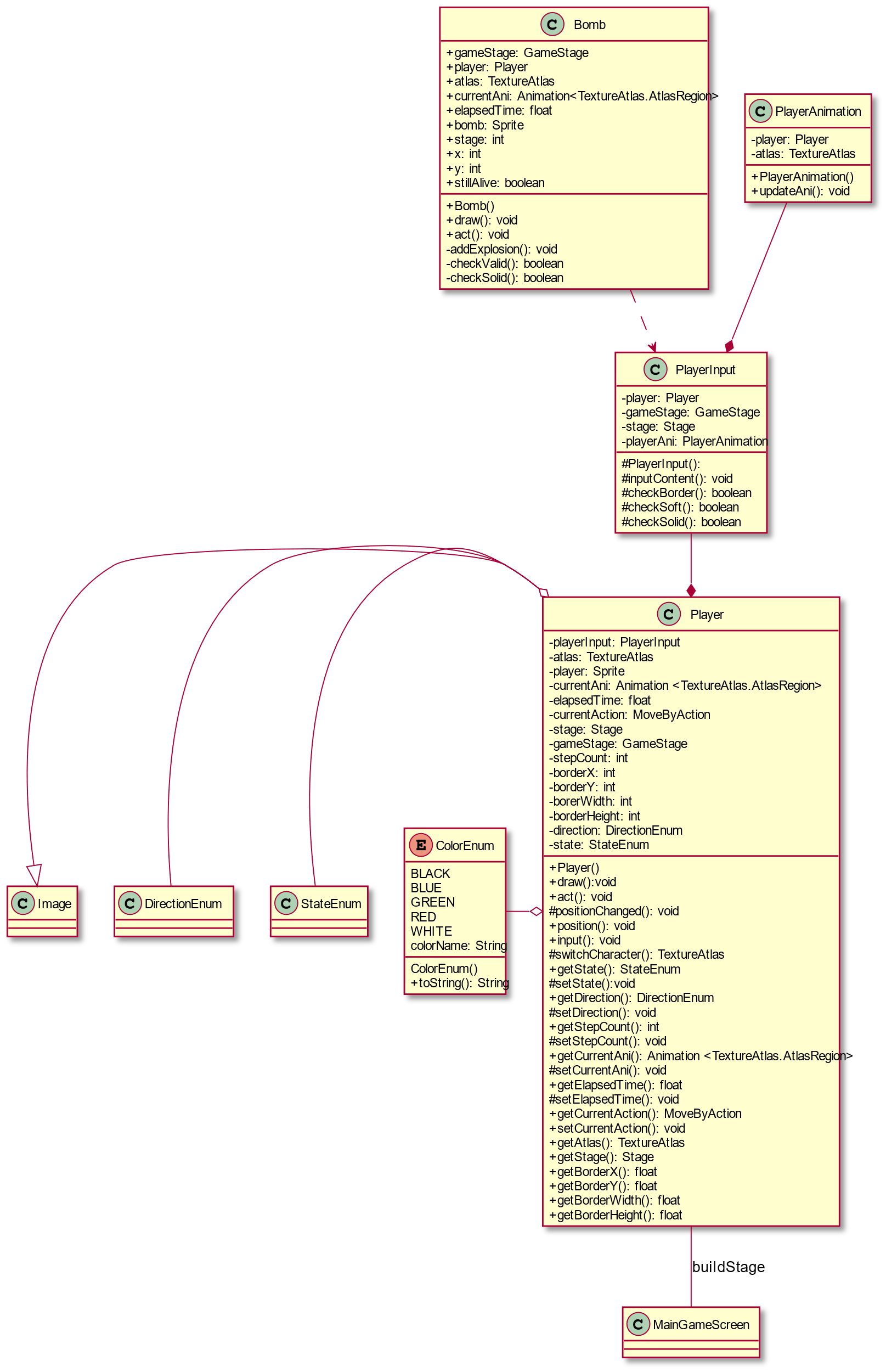
Fire up: Increase the range of explosion

# **Architecture**

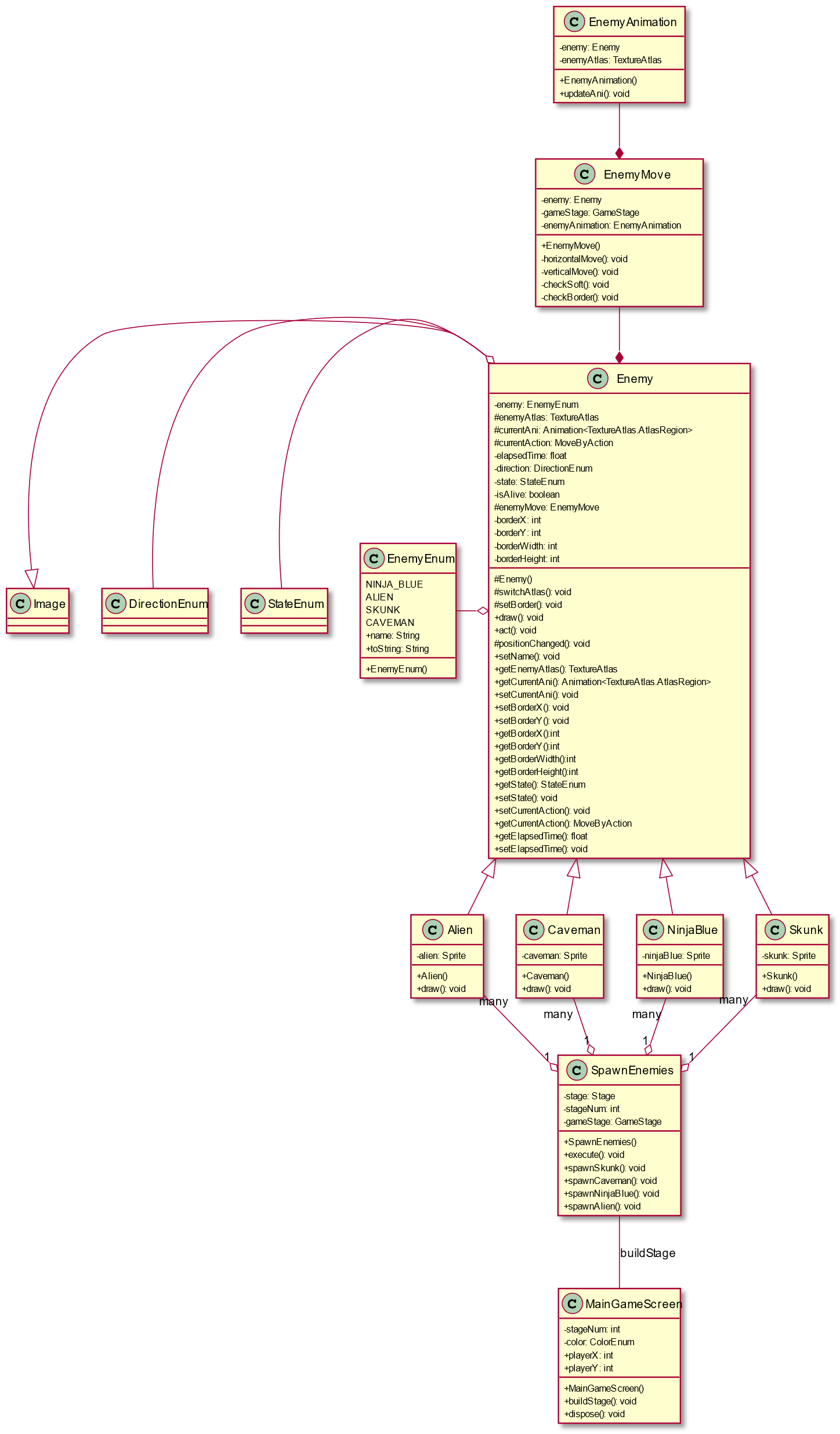
## **1.Screen**



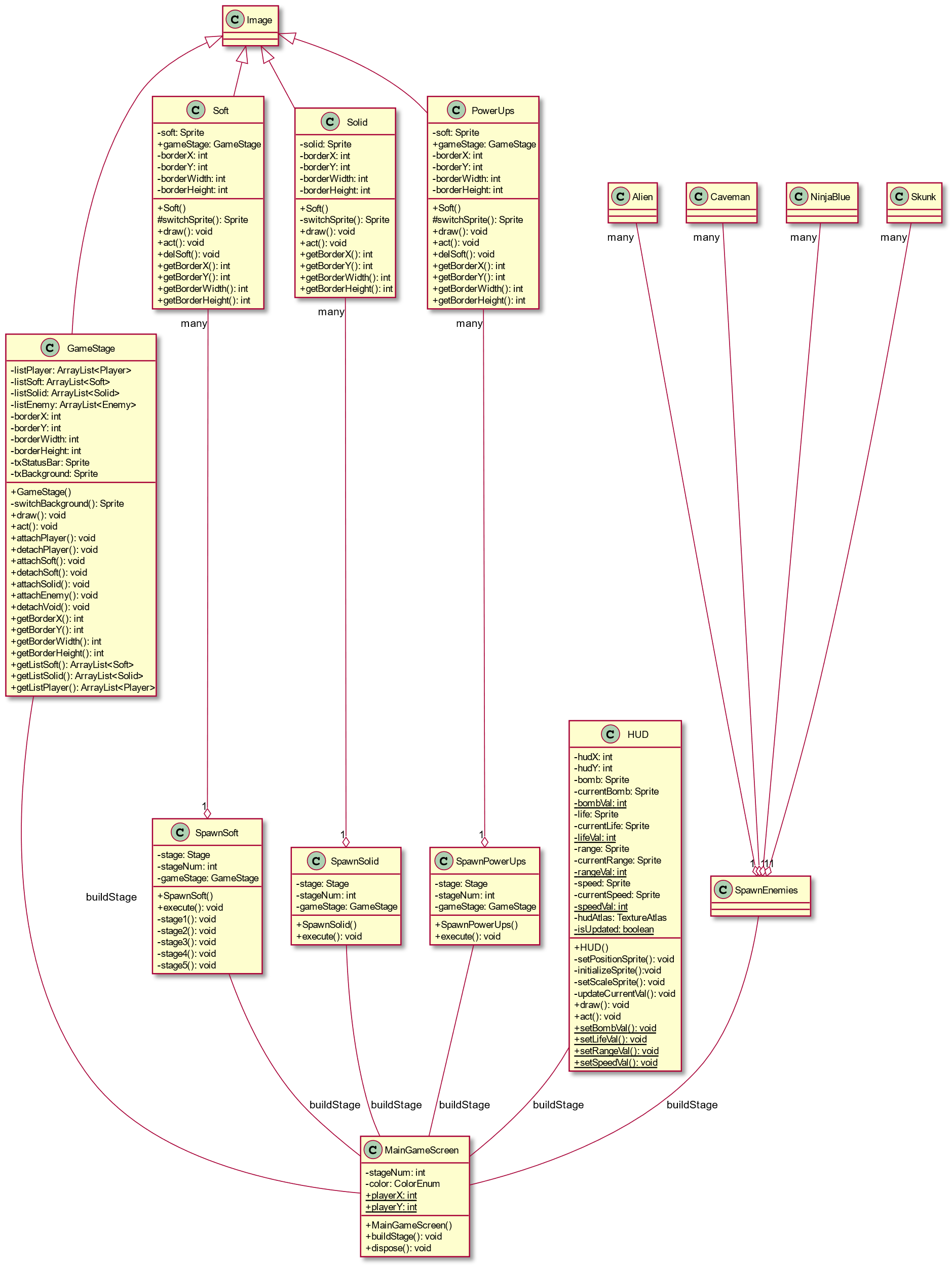
## **2. Player**



## **3. Enemy**



## **4. Item**



# **Conclusion**

From this project, our team gained more experience dealing with multithread in Java and this project also help us to review the theories we have learned in the OOP course, we know how and when to apply 4 basic properties Encapsulation, Inheritance, Polymorphism, Abstraction in a reasonable way. Also, our debugging skill is developed, with thousands of lines of code require us to collaborate in order to fix them.