Excercise 1

Exercise 1.1:

To follow the responsibility driven design we started with the requirements. Looking add the requirements we came up with the following classes:

- Game
- Alien
- Aliengrid
- SpaceShip
- EnemyBullet
- Bullet
- MovementController
- PlayerBullet
- Barricades
- Powerup
- Soundcontroller
- Explosion
- EnemyController
- SpaceshipController
- Main
- Player
- UI
- UIController
- Bulletcontroller

For this candidate classes we made CRC cards and distributed this cards to the team. We walked through scenarios to uncover unneeded and misssing classes and responsibilities. Finally we created the following list of classes with their responsibilities and collaborations:

Class	Responsibility	Collaborates with	Super	Sub
Game	All units in the game	all the controlles		
SpaceShipController	Activities of the spaceship	Game		
SpaceShip	Shooting a bullet	SpaceShipController	Unit	
ShipBullet		bulletController	Bullet	
Player	The score and lives of the player	Game		
AlienController	The movements of the aliens	Game		
Alien	Shooting a bullet	AlienController	Unit	
AlienBullet		Bullet	Bullet	
BulletController	Movements of the bullets	Game		
Bullet		BulletController	Unit	AlienBullet ShipBullet
ExplosionController	The development of the explosions	Game		
Explosion	Counter of the explosion	ExplosionController	Unit	
BarricadeController	The changes of the barricades	Game		
Barricade	Health of the barricade	BarricadeController	Unit	
Unit	Location, seize and speed of a unit			Alien, Explosion, Barricade, Bullet, SpaceShip
Collisions	The collisions between all types of units	Game		
Draw	The drawings of elements on the screen	SpaceInvadersUI		
Drawbarricade	The drawings of the barricades	Draw		
DrawBullet	The drawings of the bullets	draw		
DrawSpaceShip	The drawings of the spaceship	draw		
DrawAlien	The drawings of the aliens			
SpaceInvadersUI	The presentation of the game on the screen	GameUIController		
GameUIController	The keyboard inputs and refreshing the frames	SpaceInvadersUI		
Main	The launch of the game	Game, GameUIController		

The main difference with our actual implementation is that that game and gamUIController in our implementation have much more responsibilities.
It would have been better if we had split these classes to smaller classes as in the responsibility driven design we made. Because then it will be easier to make changes to the code while you can just replace just a small class without changing a very big class.
Classes we must add to do this are the draw and control classes.

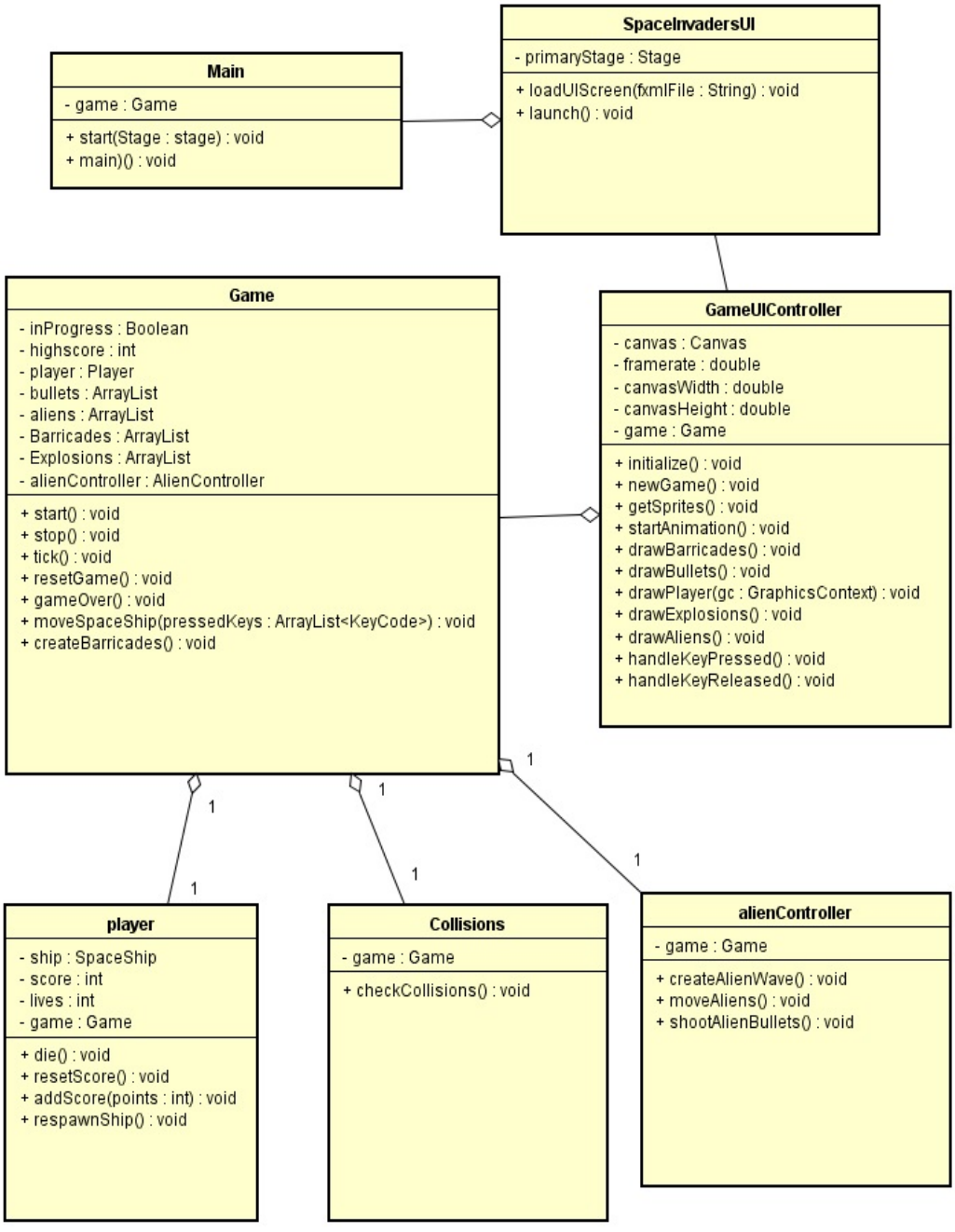
Exercise 1.2:

The main classes we implemented in our project are the game and gameUIController. The game class has the responsibility to handle the movements of the spaceship and the bullets, the creation of barricades and to monitor the game status.
This class also handles the lists of all the units in the game. It collaborates with the alien-controller, collisions and player class.
Another main class we implemented is the gameUIController which deals with the drawings of all the elements and animation, loading of the sprites, creation of a new game, and the key inputs. This class collaborates with the game and SpaceInvadersUI classes.

Exercise 1.3:

Some of the classes are less important as the classes mentioned in previous exercise, because they has less responsibilities and could easier be merged with some other class.
They are also a lot smaller what makes it easier to merge them in another class. For example we have a ShipBullet and a AlienBullet class which are subclasses of the Bullet class.
The ShipBullet and AlienBullet class could have been merged together in the bullet class.
The reason we did not do this is because we prefer to have shipbullet and alienbullet as separate classes.
It is now possible to check with instance of if a bullet is a ship or an alien bullet.
With this construction it is also easier to add attributes or methods to the alien or ship bullet only.

Exercise 1.4:


Main
- game : Game
+ start(Stage : stage) : void
+ main)() : void
SpaceInvadersUI
- primaryStage : Stage
+ loadUIScreen(fxmlFile : String) : void
+ launch() : void
Game
- inProgress : Boolean
- highscore : int
- player : Player
- bullets : ArrayList
- aliens : ArrayList
- Barricades : ArrayList
- Explosions : ArrayList
- alienController : AlienController
+ start() : void
+ stop() : void
+ tick() : void
+ resetGame() : void
+ gameOver() : void
+ moveSpaceShip(pressedKeys : ArrayList<KeyCode>) : void
+ createBarricades() : void
GameUIController
- canvas : Canvas
- framerate : double
- canvasWidth : double
- canvasHeight : double
- game : Game
+ initialize() : void
+ newGame() : void
+ getSprites() : void
+ startAnimation() : void
+ drawBarricades() : void
+ drawBullets() : void
+ drawPlayer(gc : GraphicsContext) : void
+ drawExplosions() : void
+ drawAliens() : void
+ handleKeyPressed() : void
+ handleKeyReleased() : void
1
1
1
1
1
1
player
- ship : SpaceShip
- score : int
- lives : int
- game : Game
+ die() : void
+ resetScore() : void
+ addScore(points : int) : void
+ respawnShip() : void
Collisions
- game : Game
+ checkCollisions() : void
alienController
- game : Game
+ createAlienWave() : void
+ moveAliens() : void
+ shootAlienBullets() : void