1 Exercise 1 - 20-Time, Reloaded

The following new classes were created for the implementation of the new wave features. In the table the responsibilities and collaborations are presented for every class.

Class	Responsibility	Collaborates with	Super	Sub
AlienWaveFacory	creates alienwaves	alienwaveReader, alienWave		
AlienWaveReader	Creates alienwave patternsn from file	AlienWavePattern		
AlienWavePattern	Remember pattern of alienWave			
AlienWave	Data about alienWave	Alien, AlienController		

1.1 Wave Functional Requirements

A list of functional requirements considered for the implementation of different waves using the MoSCoW method described in the previous section.

1.1.1 Must Haves

The Waves must meet the following requirements:

- A new wave shall have a different pattern of aliens.
- A wave pattern shall be stored in a file.
- Patterns shall be loaded upon starting the game.

1.1.2 Should Haves

The Waves should meet the following requirements:

- Aliens shall have different sizes.
- Aliens shall have different types.
- The game shall start with a standard pattern.
- The game shall load a random pattern upon loading a wave.
- Alien types shall have different colors.
- Alien types shall have different shooting speeds.

1.1.3 Could Haves

The Waves could meet the following requirements:

- Different alien types shall take more hits before they die.
- Rows of aliens shall move in independent directions of each other.

1.1.4 Would/Won't Haves

The Waves won't meet the following requirements:

- Aliens shall have different movement speeds.

1.2 Wave UML

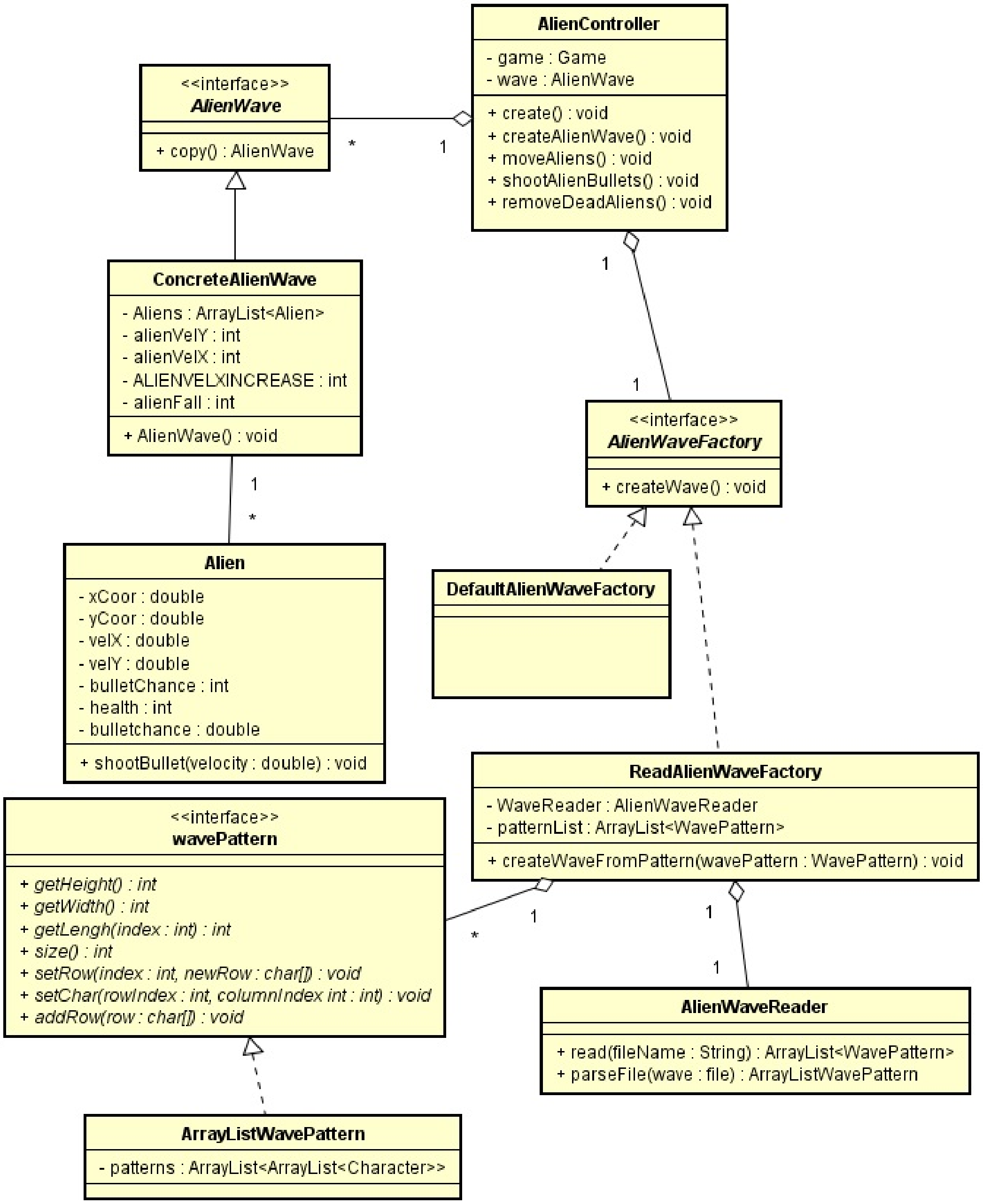
AlienController
- game : Game
- wave : AlienWave
+ create() : void
+ createAlienWave() : void
+ moveAliens() : void
+ shootAlienBullets() : void
+ removeDeadAliens() : void
<<interface>>
AlienWave
+ copy() : AlienWave
*
1
ConcreteAlienWave
- Aliens : ArrayList<Alien>
- alienVelY : int
- alienVelX : int
- ALIENVELXINCREASE : int
- alienFall : int
+ AlienWave() : void
1
1
<<interface>>
AlienWaveFactory
+ createWave() : void
1
*
Alien
- xCoor : double
- yCoor : double
- velX : double
- velY : double
- bulletChance : int
- health : int
- bulletchance : double
+ shootBullet(velocity : double) : void
DefaultAlienWaveFactory
ReadAlienWaveFactory
- WaveReader : AlienWaveReader
- patternList : ArrayList<WavePattern>
+ createWaveFromPattern(wavePattern : WavePattern) : void
<<interface>>
wavePattern
+ getHeight() : int
+ getWidth() : int
+ getLengh(index : int) : int
+ size() : int
+ setRow(index : int, newRow : char[]) : void
+ setChar(rowIndex : int, columnIndex int : int) : void
+ addRow(row : char[]) : void
1
*
1
1
AlienWaveReader
+ read(fileName : String) : ArrayList<WavePattern>
+ parseFile(wave : file) : ArrayListWavePattern
ArrayListWavePattern
- patterns : ArrayList<ArrayList<Character>>

Figure 1: Wave UML Class Diagram