Exercise 1

Requirements

Players can compete against each other and work together on one computer.

Must

- Multiple players are playable.
- The players can be controlled on one computer with the keypad and the awsd keys.
- When players compete against each other, their score must be kept separately.
- When players compete against each other, powerups are applied separately.
- When players work together, score and powerups are applied to both.

Should

- The different players should have different sprites.
- Different players should have different rope sprites

Could

- The player can name their own avatar.
- A high score is recorded for the separate players.

Would

- A player can customize the controls for his player.
- A player can play in survival mode, where bubbles keep spawning.

Must

- You can enter the survival mode from the menu.
- When a player is in survival mode, bubbles keep spawning until the player dies.
- Bubbles spawn sparingly (every minute) at first and spawn more quickly (up to one bubble every 20 seconds) as the game continues.
- When the game advances, the size of the bubbles that spawn is increased.

Should

- Save the highest score.
- Display the highscore in the menu.
- The survival mode has a different background from the normal game mode.

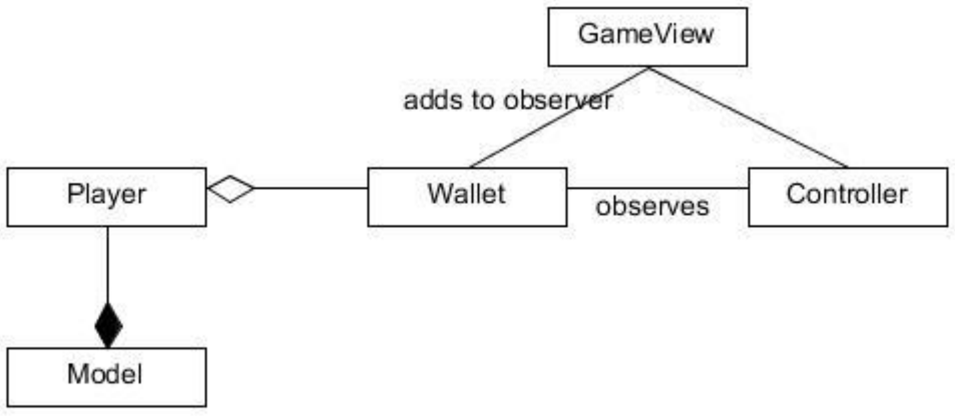
Could

- Players can compete and work together in survival mode.

Would

- The survival game has multiple ‘waves’ that are communicated through a title on the screen per wave and a rising difficulty level per wave.

This is the UML for the multiplayer:



CRC Cards
This are the CRC cards which are used for this feature. In these CRC cards only the responsibilities and the collaborations for this feature are shown.

Player	
Has a Wallet	Wallet
	Model

Model	
Contains players	Player

Wallet	
Has a value	GameView
	Controller
	Player

Controller	
Observes the wallet	GameView
	Controller

GameView	
Adds an observer to wallet	Wallet
Contains a controller	Controller

Exercise 2

1. The GameEvent class – DataClass
    a. The reason why this class is a data class, is because the class has just a constructor (which gives its fields the provided values) and getter methods. Therefore the single purpose of this class is containing data, making it a data class.
    b. << TODO
2. The Resolution class – DataClass
    a. The Resolution class is similar to the GameEvent class, it contains mainly a constructor, getters and setters. But it has additional equals() and hashmap() methods, which makes the class look like a normal class. But these methods are just methods overridden from the Object superclass. Therefore this class is indeed a data class.
    b. << TODO
3. The TimePowerupTest class - Tradition Breaker
    a. The reason this class is a tradition break is that within the TimePowerupTest only the @before statement is implemented, the other test methods don't really test anything.
    b. The design flaw is fixed by simply implementing the test class in the right way. <<TODO