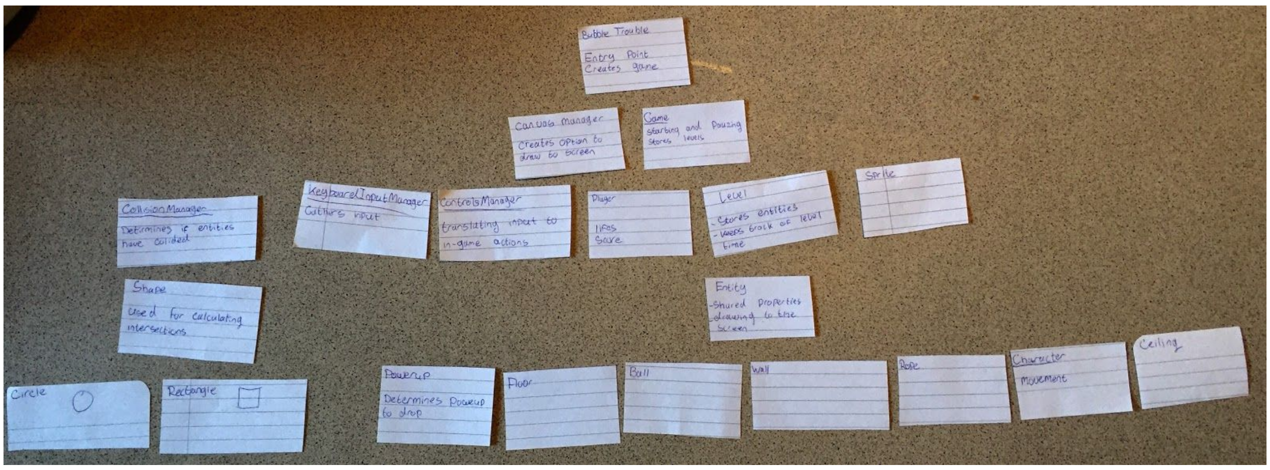
Assignment 1

Group 42

Exercise 1 - The Core

Exercise 1.1



<u>Responsibilities</u>

- BubbleTrouble - creates the game

- Game - handles the level and pauses
- Level - makes sure the player can play different games
- Player - keeps track of the score and lives of a player

- Entity - serves as a basis for all entities
  - Powerup - improves the skills of the character
  - Floor - makes sure objects stay within the frame
  - Ball - This class makes sure that there exist different balls (size, colour, bounce height). It also makes sure that the balls bounce and that the right things happen if a ball collides with the floor (block), a wall, the rope. For example, a ball should split into two smaller balls if it collides with the rope.
  - Wall - makes sure objects stay within the frame
  - Rope - This class makes sure that there exists a rope if the player presses ‘shoot’. The player can shoot one rope at a time that is present until it collides with a ball or the floor. This class also handles what happens after the rope and a ball or the rope and the floor hit each other.
  - Character - This class makes sure that there exists a player that can walk around. The class also remembers whether a character is dead or alive. In addition, the character also handles collisions with balls, the floor (block) and a wall.

    -

- Shape - serves as a basis for different shapes
    - Circle - needed to track collisions with all circle shaped objects
    - Rectangle - needed to track collisions with rectangle shaped objects

- Sprite - displays an image
- GameCanvasManager - make the canvas
- KeyboardInputManager - handles the input from the keyboard
- ControllsManager - transforms input to actions

<u>Collaborations</u>

- Entity and Shape - Shape defines the Entities collision shape
- Powerup and Character - Powerup improves Character's skill
- BubbleTrouble and Game - BubbleTrouble starts a new Game
- Game and Levels - the Game class loads levels using the Level class
- Level and Entities - for every level the Level class creates and manages Entities, this way all levels are different
- Sprite and GameCanvasManager - Sprite uses GameCanvasManager to get the canvas to draw on
- Player and Character - the Player class (real person) keeps track of the stats of a Character
- ControllsManager and Character - controls a character in the Game

<u>Compare results</u>

When comparing the previous result with the actual implementation we come across two major differences. First of all, we have made a distinction between the real-life player and the player in the game, though we have not yet implemented this in our code. We created the Player class and the Character class. The Player class will keep track of the score and lives of a player, while the Character class will make sure that the character can walk around and can shoot a rope. Another difference that our actual implementation does not contain a Powerup class. We have not added this class before, since we have not implemented any powerups so far.

Exercise 1.2

Quite similar to the result of the previous exercise:

<u>Responsibilities</u>

- BubbleTrouble - creates the game

- Entity - serves as a basis for all entities
    - Ball - This class makes sure that there exist different balls (size, colour, bounce height). It also makes sure that the balls bounce and that the right things happen if a ball collides with the floor (block), a wall, the rope. For example, a ball should split into two smaller balls if it collides with the rope.

    - Rope - This class makes sure that there exists a rope if the player presses ‘shoot’. The player can shoot one rope at a time that is present until it collides with a ball or the floor. This class also handles what happens after the rope and a ball or the rope and the floor hit each other.
    - Player - This class makes sure that there exists a character that can walk around. The class also remembers whether a character is dead or alive. In addition, the character also handles collisions with balls, the floor (block) and a wall.
    - Block - makes sure objects stay within the frame

- Shape - serves as a basis for different shapes
    - Circle - needed to track collisions with all circle shaped objects
    - Rectangle - needed to track collisions with rectangle shaped objects

- GameCanvasManager - make the canvas
- KeyboardInputManager - handles the input from the keyboard
- ControlsManager - transforms input to actions

Collaborations
- Entity and Shape - Shape defines the Entities collision shape
- ControllsManager and Player - controls a character in the Game

Exercise 1.3

The classes that were not included in the previous exercise are: Game, Level, Powerup, Floor, Wall and Sprite.

Game
Even though, the Game class does a great responsibility (starting up the game, tracking if it is paused and tracking the level), it was not considered a main class. That is because we think it could be merged the BubbleTrouble class. Both classes share quite similar responsibilities.

Level
The Level class is responsible for the different levels in the game. Since, having different levels is not a Must Have, we do not consider this class to be a main class.

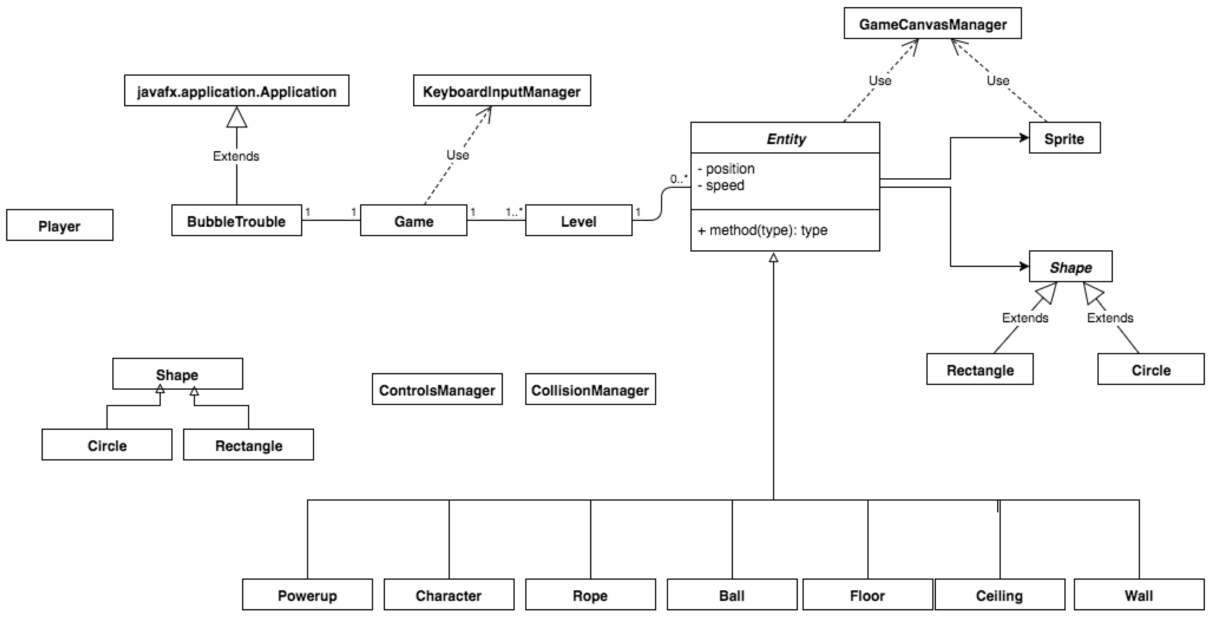
Powerup
Having powerups in the game is an additional feature. It is not necessary in order for someone to play the game.

Floor and Wall
Floor and Wall have similar responsibilities (using collision to make sure objects do not leave the frame) and could easily be merged into one entity that we could name Block.

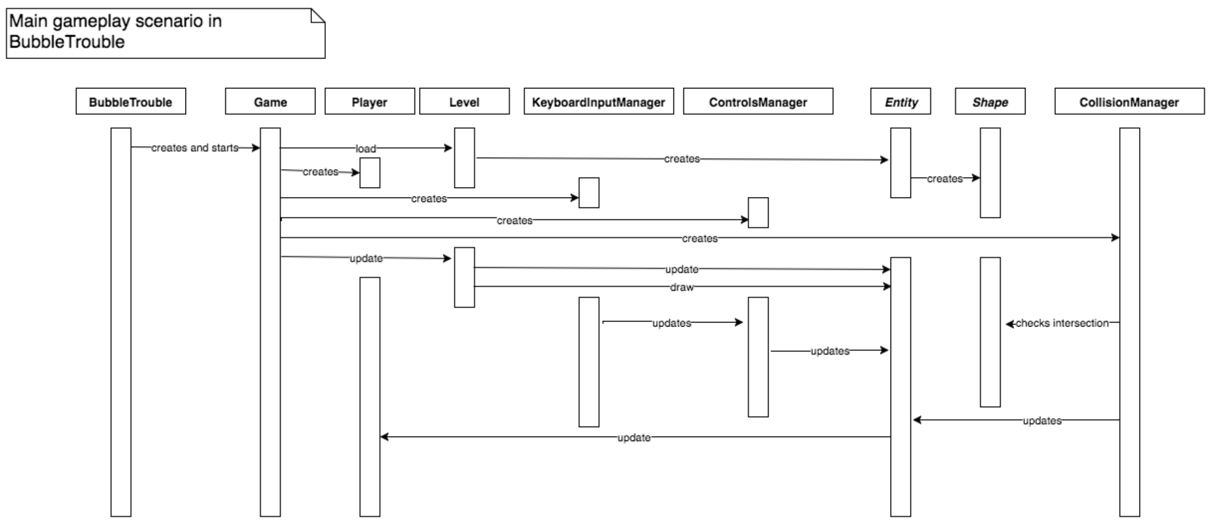
Sprite

We do not consider the Sprite class to be a main class. Namely, every sprite can be replaced by an image to make sure the player can still observe the position of the objects. Therefore, the class is not crucial to play the game. However, it does make the game look a lot better and should not be removed.

Exercise 1.4



Exercise 1.5



Exercise 2 - UML in practice

Exercise 2.1

The current source code uses both composition and aggregation.
Aggregation between A and B means that B may also exist independent of A.
Composition between A and B means that B cannot exist without A.

Occurrences:

- Character and Rope - The Rope can exist without a Character, but a Character creates a Rope on instantiation: Aggregation
- Level and Entity - Entities are useless if they are not placed in a level: Composition
- Entity and Shape - An entity can have a shape for collision, but shapes can also exist separately: Aggregation
- Game and Level - Levels do not do anything without a Game: Composition

Exercise 2.2

Currently the source code does not contain any parameterized classes. Instead of an abstract class Powerup, we could make it a parametrized class: PowerupContainer<T extends Powerup>. This would separate the falling behaviour of the PowerupContainer and the skill improving behaviour of a Character.

Exercise 2.3

Exercise 3 - Simple logging

Exercise 3.1

Requirements for logging:

- The logger shall be able to log log records to a any output stream.
- The logger shall have a log level, which indicates which messages to log and which ones to ignore.
- The following log levels shall exist (ordered by importance):
    - **OFF:** The OFF LogLevel has the highest possible rank and is intended to turn off logging. Never write logs with this LogLevel.
    - **FATAL:** The FATAL LogLevel designates very severe error events that will presumably lead the application to abort.
    - **ERROR:** The ERROR LogLevel designates error events that might still allow the application to continue running.
    - **WARN:** The WARN LogLevel designates potentially harmful situations.
    - **INFO:** The INFO LogLevel designates informational messages that highlight the progress of the application at coarse-grained level.
    - **DEBUG:** The DEBUG LogLevel designates fine-grained informational events that are most useful to debug an application.
    - **TRACE:** The TRACE LogLevel designates more fine-grained informational events than the DEBUG LogLevel.
    - **ALL:** The ALL Level has the lowest possible rank and is intended to turn on all logging. Never write logs with this LogLevel.
- The logger shall be available everywhere through static calls.
- The logger shall be able to find out which class called it.
- The logger shall include a timestamp in each log record.

Exercise 3.2