SEM - ASSIGNMENT 3

Group 42

Exercise 1 - Design Patterns

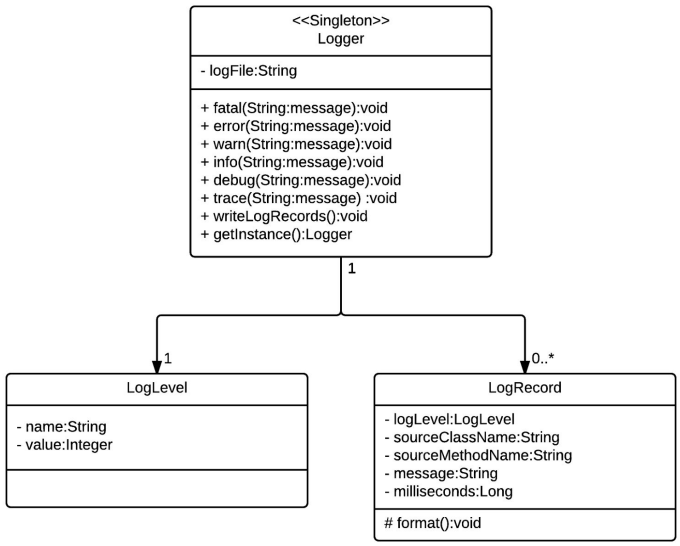
Exercise 1.1 - Singleton

1. **Write a natural language description of why and how the pattern is implemented in your code.**

We implemented the Singleton pattern for the Logger class.

Logger is a singleton, so that all LogRecords are stored in the same List, and all logging is done to the same file. This way all classes that want to log something only have to call `Logger.getInstance()` and are ready to go.

2. **Make a class diagram of how the pattern is structured statically in your code.**



Exercise 2 - Menu

The main menu allows players to choose between singleplayer, multiplayer,quit and settings.

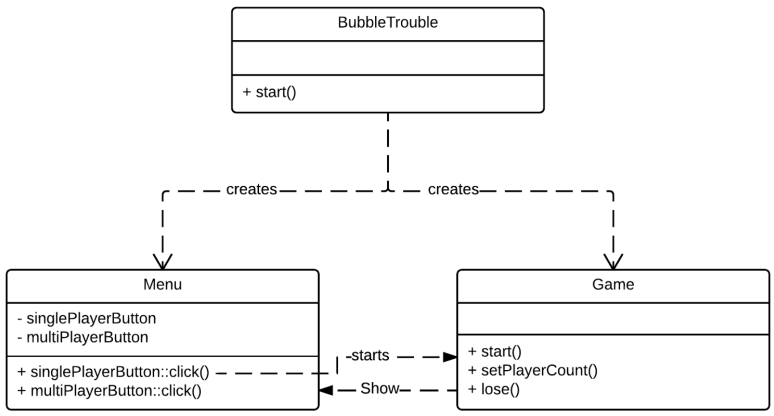
Requirements

The requirements (also found on https://github.com/MatthijsKok/TI2206/wiki/Requirement-documents) are:

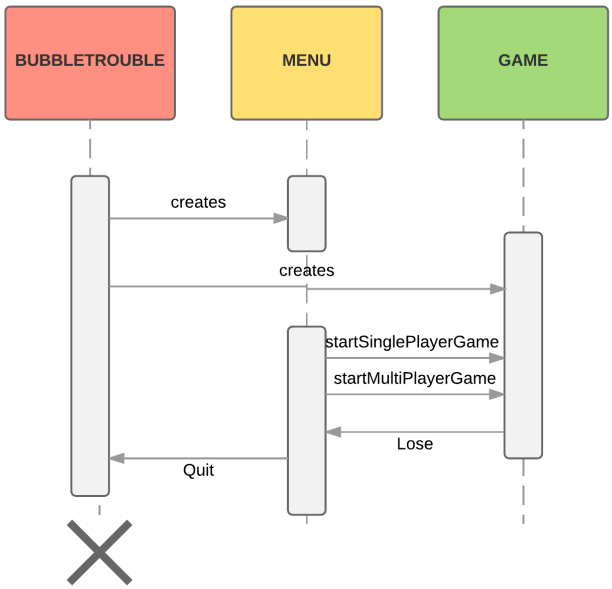
- The menu should contain a title with the name of the game. "bubble Trouble" (name can be changed).
- The menu should contain a background witch is representative for our game.
- The menu should contain four buttons:
    - Start single player game
    - Start multi player game
    - Quit
    - Settings
- If the player clicks on the "start single player game" button, a single player game should start.
- If the player clicks on the "start multi player game" button, a multi player game should start.
- If the player clicks on the "Quit" button, the window should be closed and the javaFX application should stop running.

UML Diagram

Class diagram



Sequence diagram


BUBBLETROUBLE
MENU
GAME
creates
creates
startSinglePlayerGame
startMultiPlayerGame
Lose
Quit

Exercise 3 - Power Ups

The powerup extension gives the game a lot more versatility. When a ball is hit by a harpoon, it has a chance to drop a power up. When a character collides with this powerup, an effect occurs that changes the behaviour of the player, the level.

Requirements

The requirements (also found on https://github.com/MatthijsKok/TI2206/wiki/Requirement-documents) are:

- The game shall have at least two power ups that act on different entities in the game.
- A power up can only be used after the character has grabbed it, that is, the character has collided with the pick up of the power up.
- If the character already had one power up and then grabs another one the power ups should stack, that is, the character has the benefits of both power ups at the same time, or the effect should be improved.
- If a player has an active power up of which the effect can not be improved by picking up another one of that type, a second power up of that type will not spawn until the effect of the previous one has expired.
- A power up has a time limit after which the effect of the power up expires.
- A pick up of the power up can randomly appear after the rope has collided with a ball.
- A pick up of the power up will be available for a limited amount of time.
- A pick up of the power up will disappear after the character has grabbed it.
- One of the power-ups is that the character receives an extra life.
- One of the power-ups is that the character received more time to finish the level.
- One of the power-ups is that the character can move around faster.
- One of the power-ups is that the character can shoot an additional rope.
- One of the power-ups is that the character is protected from the ball by a shield, that is, the player can collide with a ball once, after this the ball splits and the shield disappears.
- After the level is completed power-ups that are still active will deactivate.

UML Diagram

Class diagram