Assignment 3 Pang! :

Group: -1

Jurriaan den Toonder: 4431324
Erik Wiegel: 4476328
Govert de Gans: 4491955
Jaap-Jan van der Steeg: 4456130
Tim Rietveld: 4472926

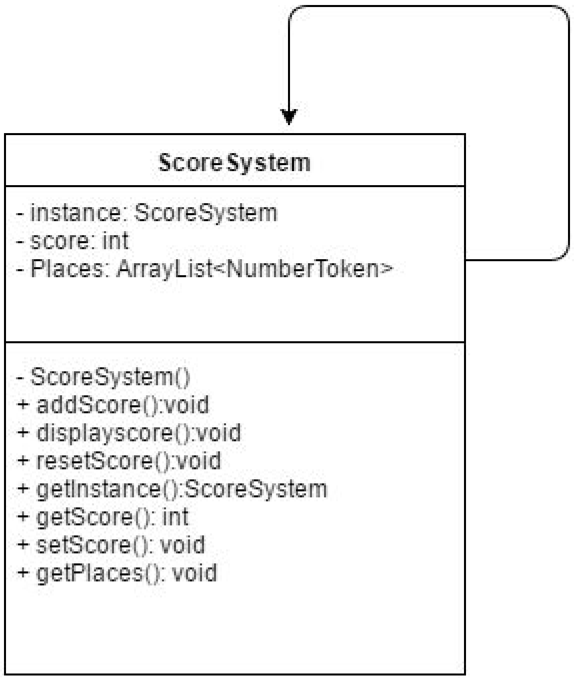
Exercise 1a - Design patterns (15 pts)
Choose two design patterns among those that we saw in class3 and that you did not use in a previous assignment. For each chosen design pattern, you must have a corresponding implementation in your code. If not, refactor your code to include it. Then, per each chosen design pattern, complete the following points:

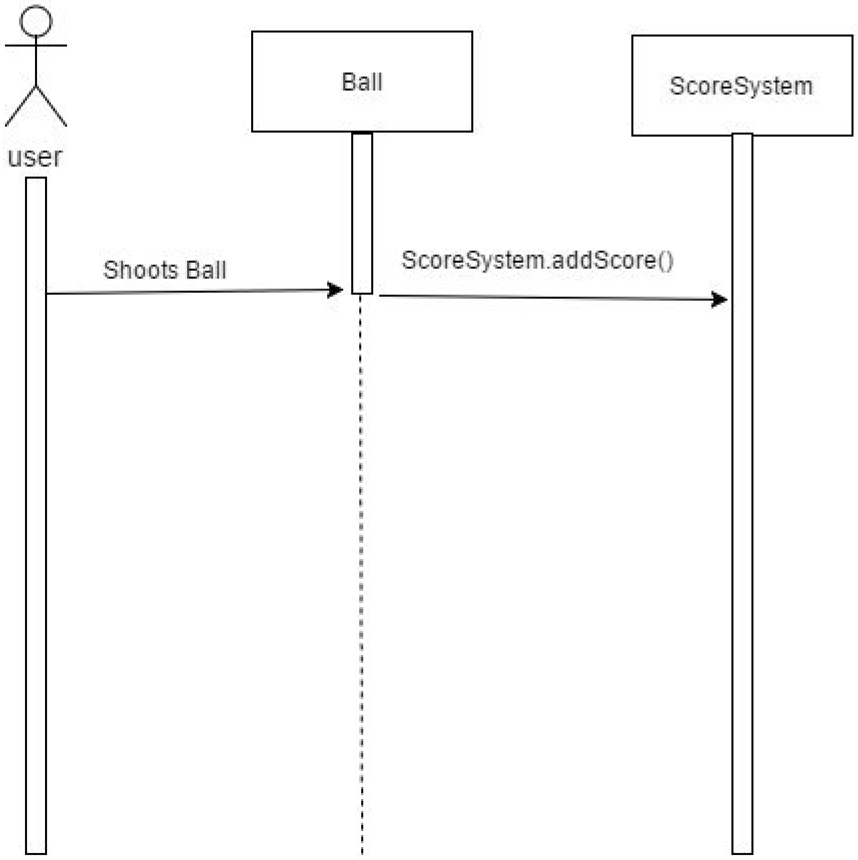
1. Write a natural language description of why and how the pattern is implemented in your code (5 pts).

We use the singleton strategy for the score system to avoid accidentally making multiple instances of the score system, because we only ever need one at the same time and having multiple instances could lead to improper score changes.

The singleton is implemented by making the constructor private and creating an instance in the variable field declaration.

2. Make a class diagram of how the pattern is structured statically in your code (5 pts).
See next page.
3. Make a sequence diagram of how the pattern works dynamically in your code (5 pts)
See next page.


ScoreSystem
- instance: ScoreSystem
- score: int
- Places: ArrayList<NumberToken>
- ScoreSystem()
+ addScore():void
+ displayscore():void
+ resetScore():void
+ getInstance():ScoreSystem
+ getScore(): int
+ setScore(): void
+ getPlaces(): void


user
Ball
ScoreSystem
Shoots Ball
ScoreSystem.addScore()

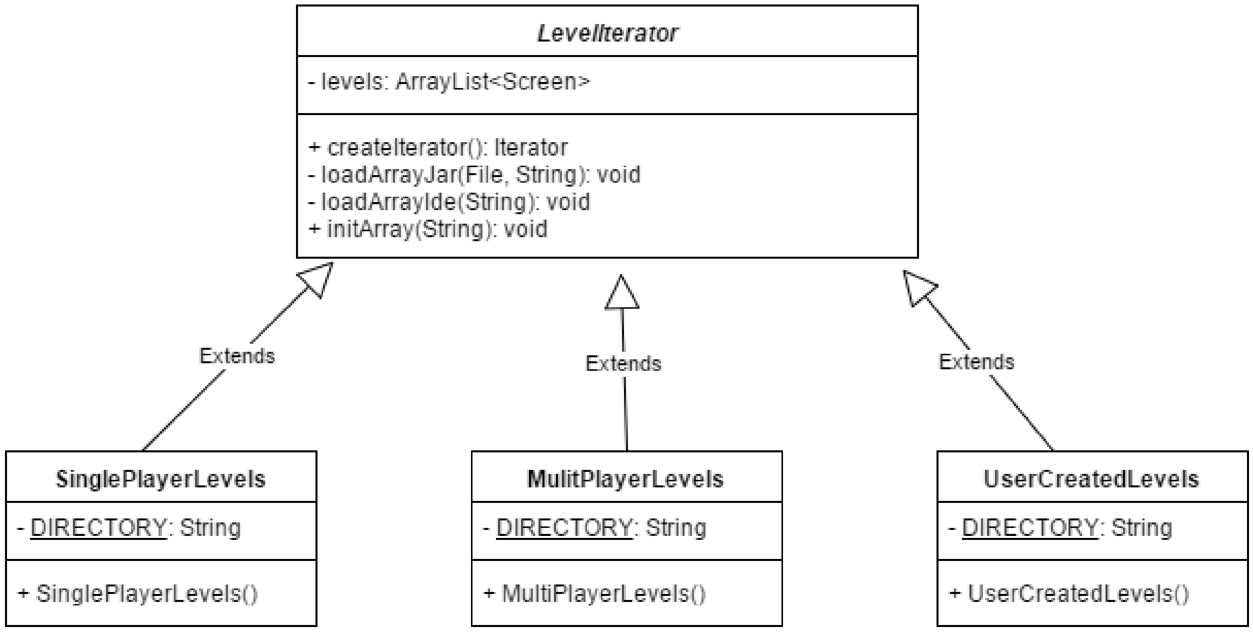
Exercise 1b - Design patterns (15 pts)
Choose two design patterns among those that we saw in class3 and that you did not use in a previous assignment. For each chosen design pattern, you must have a corresponding implementation in your code. If not, refactor your code to include it. Then, per each chosen design pattern, complete the following points:

1. Write a natural language description of why and how the pattern is implemented in your code (5 pts).

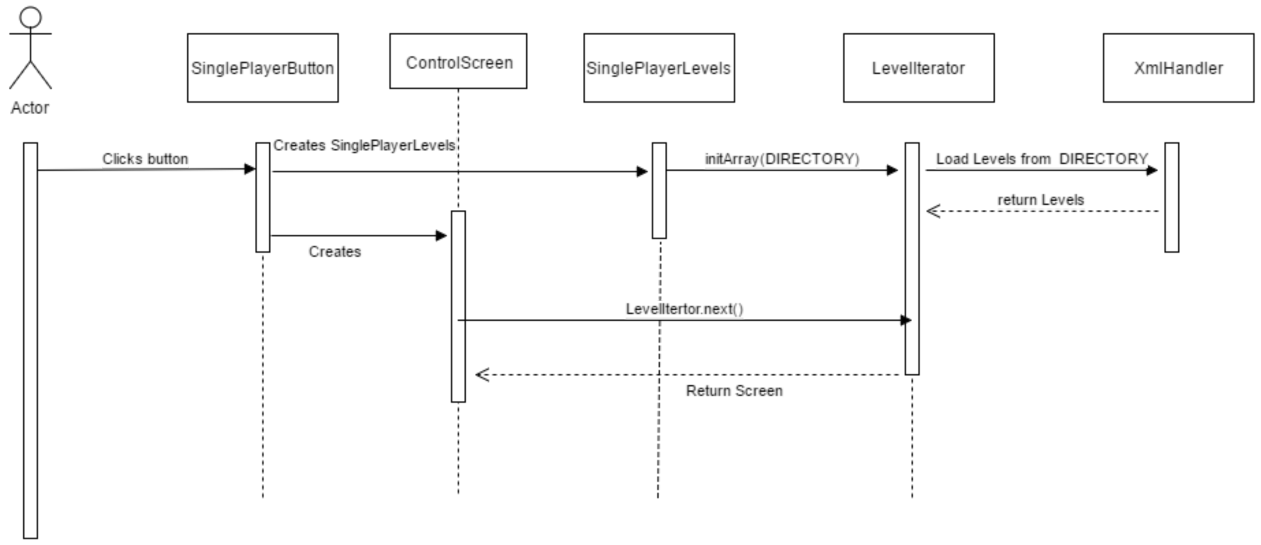
We use the iterator design pattern to be able to load lists of levels easily. We want to be able to distinguish between singleplayer, multiplayer and custom (user made) levels, but treat them in the same way when we are loading the next level.

The pattern is implemented by an abstract class, which is inherited by the classes which specify the different types of levels. The abstract class contains an ArrayList which is instantiated differently by the different subclasses.

2. Make a class diagram of how the pattern is structured statically in your code (5 pts).



3. Make a sequence diagram of how the pattern works dynamically in your code (5 pts)



Exercise 2:

Requirements Pang!: Level Editor:

Group: -1

Jurriaan den Toonder: 4431324
Erik Wiegel: 4476328
Govert de Gans: 4491955
Jaap-Jan van der Steeg: 4456130
Tim Rietveld: 4472926

Functional Requirements:

Must have:

- It must be possible to add balls in the level editor.
- It must be possible to add a player in the level editor.
- It must be possible to remove objects from the level editor.
- It must be clear to the user what buttons can be used in the editor.
- It must be possible to drag and drop objects in the level editor.
- It must be possible to play the user created level.

Should have:

- In the level editor, It should be possible to choose the size of the balls.
- It should be possible to write the level to an xml file so that the level can be played more often.

Could have:

- The level editor could support the creation of a multiplayer game.
- The level editor could have a check to ensure a level cannot be saved if it doesnt contain:
    - 1 or 2 players
    - And at least 1 ball
- After playing the level the screen could return to the level editor and show the last created level.

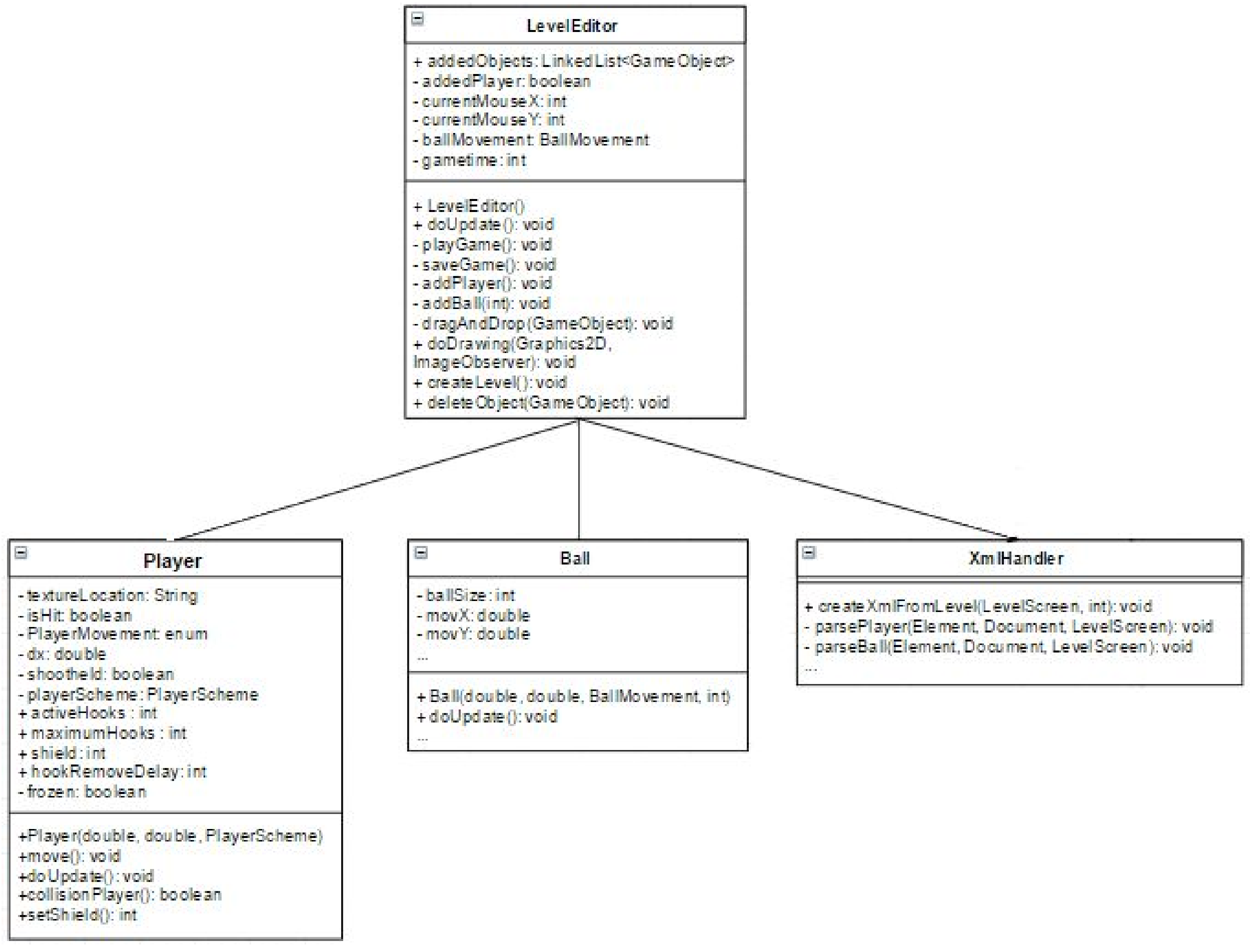
Won't have:

- The level editor won't have the functionality to choose which direction the ball goes to.
- You won't be able to select a specific power-up to be dropped by the balls.
- You will not be able to set a time for the level.

Non-functional Requirements

- The user should be able to use his keyboard to add new objects to the scene.

2. During the analysis and design phases of this extension use responsibility driven design and UML (push to the repository the single PDF file including all the documents produced) (8 pts).



Exercise 3 - 20-Time (30 pts)

1. Google asks its employees to spend 20% of their time at Google to a project that their job description does not cover. As a result of the 20% Project at Google, we now have Gmail, AdSense, and Google News, among the others. This is your occasion to have similar freedom. You can decide what to do next to your game:
It can be an extension/improvement from any perspective, such as improved code quality, or novel features. Define your own requirements and get them approved by your teaching assistant. Afterwards you must implement the requirements. (22 pts).

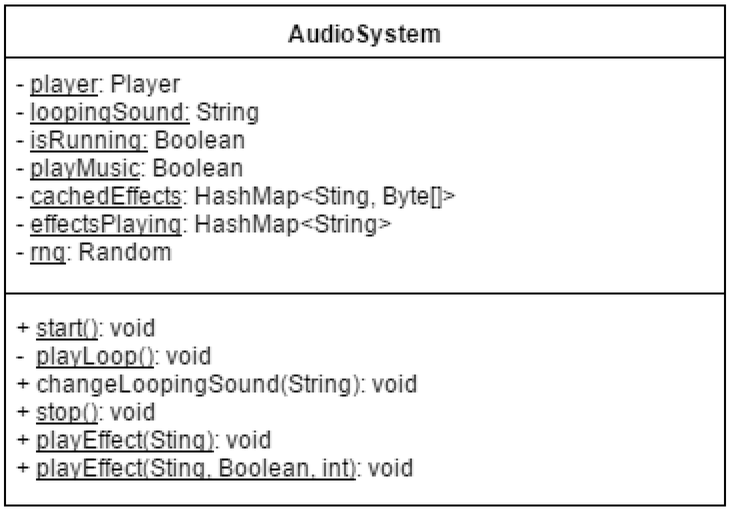
Sound effects.

As a player, I want to hear sound effects when I walk, shoot ropes and hit balls to enjoy the game even more.

The game should play a sound effect for the following situations:

- Player moves
- Player shoots a rope
- Player hits a ball
- Player picks up a powerup
- Player picks up a coin

2. During the analysis and design phases of this extension use responsibility driven design and UML (push to the repository the single PDF file including all the documents produced) (8 pts).


AudioSystem
- player: Player
- loopingSound: String
- isRunning: Boolean
- playMusic: Boolean
- cachedEffects: HashMap<Sting, Byte[]>
- effectsPlaying: HashMap<String>
- rng: Random
+ start(): void
- playLoop(): void
+ changeLoopingSound(String): void
+ stop(): void
+ playEffect(Sting): void
+ playEffect(Sting, Boolean, int): void

We already have an AudioSystem, so the most logical solution would be to add a method for playing sound effects in that.
Then, on collision/key press, each class can play their own sounds.
The player object should play a shooting sound when shooting a hook, the drop object should play an item pickup sound, and the ball should play a pop sound, etc.