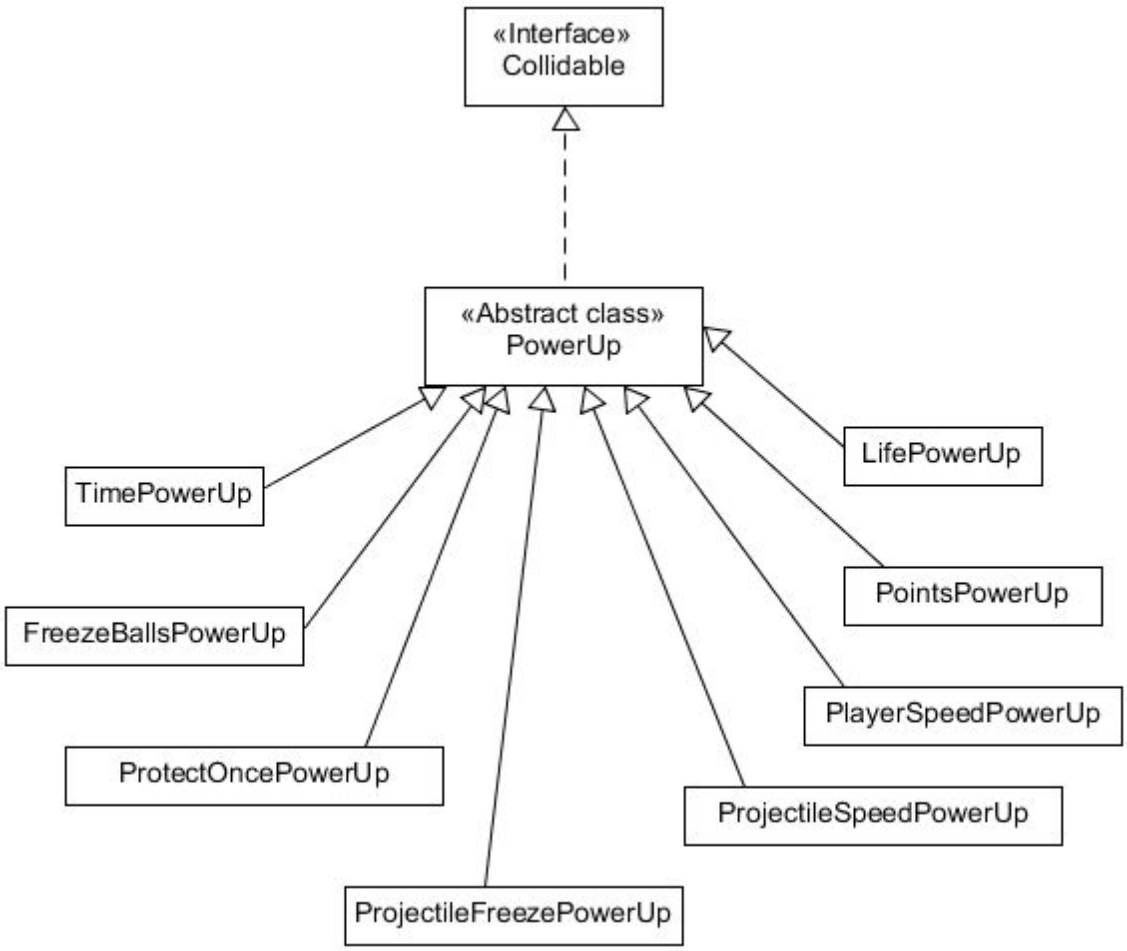
Exercise 2.1 Responsibility driven design

1. For this assignment, we wanted walls that disappeared and walls that partly stay, so we thought of adding functionality to the Wall class and adding a new MoveableWall class to be able to see the difference
2. The main class for this assignment is Wall, which is responsible for the creation of walls. Level is also important for logic and LevelFactory had to be adjusted to be able to load these walls.
3. We decided not to make use of the MoveableWall class because it was unnecessary complex and more difficult than adding that functionality to the Wall class itself.
4. There are no non-important classes for this feature.
5. The UML and diagrams are not different from earlier ones.

Exercise 2.2 Responsibility driven design

1. To extend the game with powerup functionality, as described in the requirements, the existing framework and design the game had could stay the same. However, a few changes in the Player and Level class were necessary. These changes include the addition of some flags to determine the game state and some methods in Level to initialize and activate powerups.
2. All PowerUps extend the PowerUp class, which holds basic powerup functionality such as powerup duration and some callback classes for powerup.
3. The available powerups are generated upon Level initialization, by searching for all classes that extend the PowerUp class, via reflection. This makes adding new powerups very easy and efficient. It is only necessary to add an extra class. Nowhere does the powerup have to be registered.
4. The powerup class is only constructed when the actual powerup is dropped. The dropping chance is determined by an annotation, PowerUpChance.
5. The important classes are PowerUp and PowerUpAnnotation, as they provide the basic, extendable functionality.


«Interface»
Collidable
«Abstract class»
PowerUp
LifePowerUp
TimePowerUp
PointsPowerUp
FreezeBallsPowerUp
PlayerSpeedPowerUp
ProtectOncePowerUp
ProjectileSpeedPowerUp
ProjectileFreezePowerUp

Exercise 2.1 CRC

Wall	
Superclass: -	
Subclasses: MoveableWall	
Load the wall	LevelFactory
Logic when the wall disappears	Level

MoveableWall	
Superclass: Wall	
Subclasses: -	
Let the wall partly disappear	Level

PowerUp	
Superclass: -	
Subclasses: TimePowerUp, ProtectOncePowerUp, ProjectileSpeedPowerUp ProjectileFreezePowerUp, PointsPowerUp, PlayerSpeedPowerUp, LifePowerUp, FreezeBallsPowerUp	
Influence level state	Level
Influence state of player that picked up the powerup	Player

TimePowerUp	
Superclass: PowerUp	
Subclasses: -	
Add time to level	Level

ProjectileSpeedPowerUp	
Superclass: PowerUp	
Subclasses: -	
Increase projectile start speed	Level

ProjectileFreezePowerUp	
Superclass: PowerUp	
Subclasses: -	
Add projectile freeze flag to level	Level

PointsPowerUp	
Superclass: PowerUp	
Subclasses: -	
Add points to player	Player

PlayerSpeedPowerUp	
Superclass: PowerUp	
Subclasses: -	
Increase player speed	Player

LifePowerUp	
Superclass: PowerUp	
Subclasses: -	
Add lives to player	Player

FreezeBallsPowerUp	
Superclass: PowerUp	
Subclasses: -	
Stop ball moving on level	Level

ProtectOncePowerUp	
Superclass: PowerUp	
Subclasses: -	
Add collision ignore to level	Level

ProtectOncePowerUp	
Superclass: PowerUp	
Subclasses: -	
Add collision ignore to level	Level

TimePowerUp	
Superclass: PowerUp	
Subclasses: TimePowerUp, ProtectOncePowerUp, ProjectileSpeedPowerUp ProjectileFreezePowerUp, PointsPowerUp, PlayerSpeedPowerUp, LifePowerUp, FreezeBallsPowerUp	
Add time to level	Level
Influence player stat e	Player