COMP-302: SOFTWARE ENGINEERING

NEED FOR SPEAR

Phase2:
REQUIREMENTS AND MODELING
&
DESIGN

SyntacticSugar:

İrem Demir

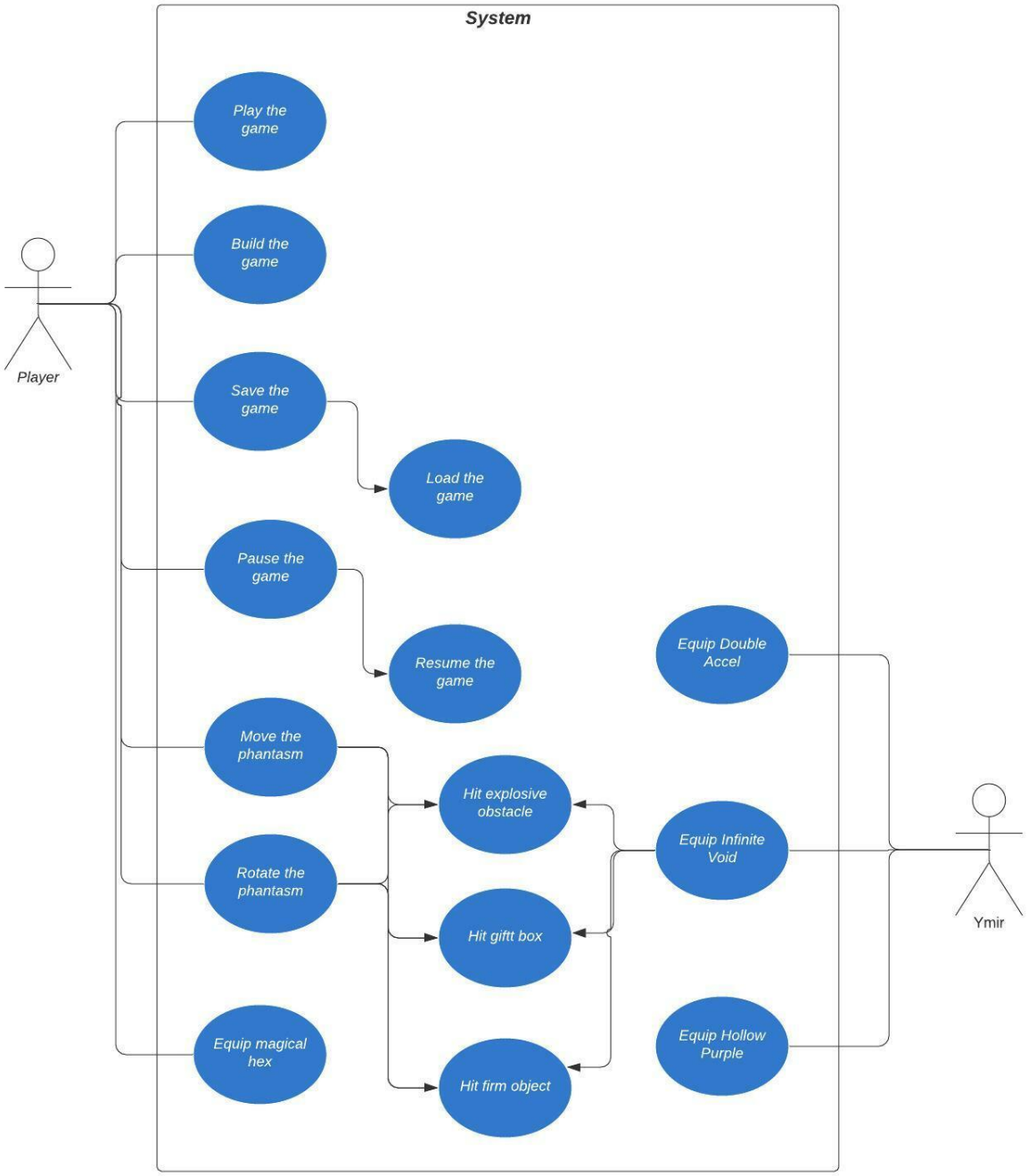
Atakan Kara

Asu Tutku Gökçek

Farrin Marouf Sofian

Eren Yenigül

UML Use Case Diagram



Use Case Narratives:

Use Case 14: Equip Infinite Void

Use Case Name: Equip Infinite Void
Scope: Game
Level: Subfunction
Primary Actor: Ymir
Stakeholders and Interests: Ymir / wants to make game difficult for player
Preconditions: The game is running and Ymir had a successful toss and infinite void is the randomly chosen magical ability.
Success Guarantee: Chosen obstacles are freezed for 15 seconds and cannot be affected by an enchanted sphere nor the magical hex ability. They can be only destroyed by the unstoppable enchanted sphere.
Main Success Scenario:

1. The infinite void equipped by the game.
2. Randomly chosen 8 obstacles are freezed
3. Ball hits the frozen obstacles it reflected but obstacles are not destroyed
4. Magical hex used and no effect on the obstacles
5. Infinite void stays active for 15 seconds.

Extensions:

*a At anytime, player performs Pause the Game
1.Player performs Resume the Game
2.Player performs Save the Game
3.Player goes back to start
4.Player exits the game
2.a. There are less than 8 obstacles in the game
2.b. All of the obstacles are chosen
3.a. Ball is in the unstoppable mode
3.b. Ball acts like a normal ball on the frozen obstacles

Special Requirements:Game is installed on the machine.
Technology and Data Variations List:
Frequency of Occurrence: On every 30 seconds Ymir wins coin toss(50%) and choses infinite void among 3 options.
Miscellaneous:

Use Case 15: Equip Double Accel

Use Case Name: Equip Double Accel
Scope: Game
Level: Subfunction
Primary Actor: Ymir
Stakeholders and Interests: Ymir / wants to make game difficult for player
Preconditions: The game is running and Ymir had a successful toss and double accel is the randomly chosen magical ability.
Success Guarantee: Speed of the enchanted sphere is reduced by half.
Main Success Scenario:

1. The double accel equipped by the game.
2. Speed of the enchanted sphere is reduced by half
3. Enchanted sphere moves slower
4. Double accel stays active for 15 seconds.

Extensions:

*a At anytime, player performs Pause the Game

1. Player performs Resume the Game
2. Player performs Save the Game
3. Player goes back to start
4. Player exits the game

Special Requirements:Game is installed on the machine.
Technology and Data Variations List:
Frequency of Occurrence: On every 30 seconds Ymir wins coin toss(50%) and choses double accel among 3 options.
Miscellaneous:

Use Case 16: Equip Hollow Purple

Use Case Name: Equip Hollow Purple
Scope: Game
Level:Subfunction
Primary Actor: Ymir
Stakeholders and Interests: Ymir / wants to make game difficult for player
Preconditions: The game is running and Ymir had a successful toss and hollow purple is the randomly chosen magical ability.
Success Guarantee: 8 hollow purple obstacles are added to the game
Main Success Scenario:

1. The hollow purple equipped by the game.
2. 8 hollow purple obstacles are added to the game randomly

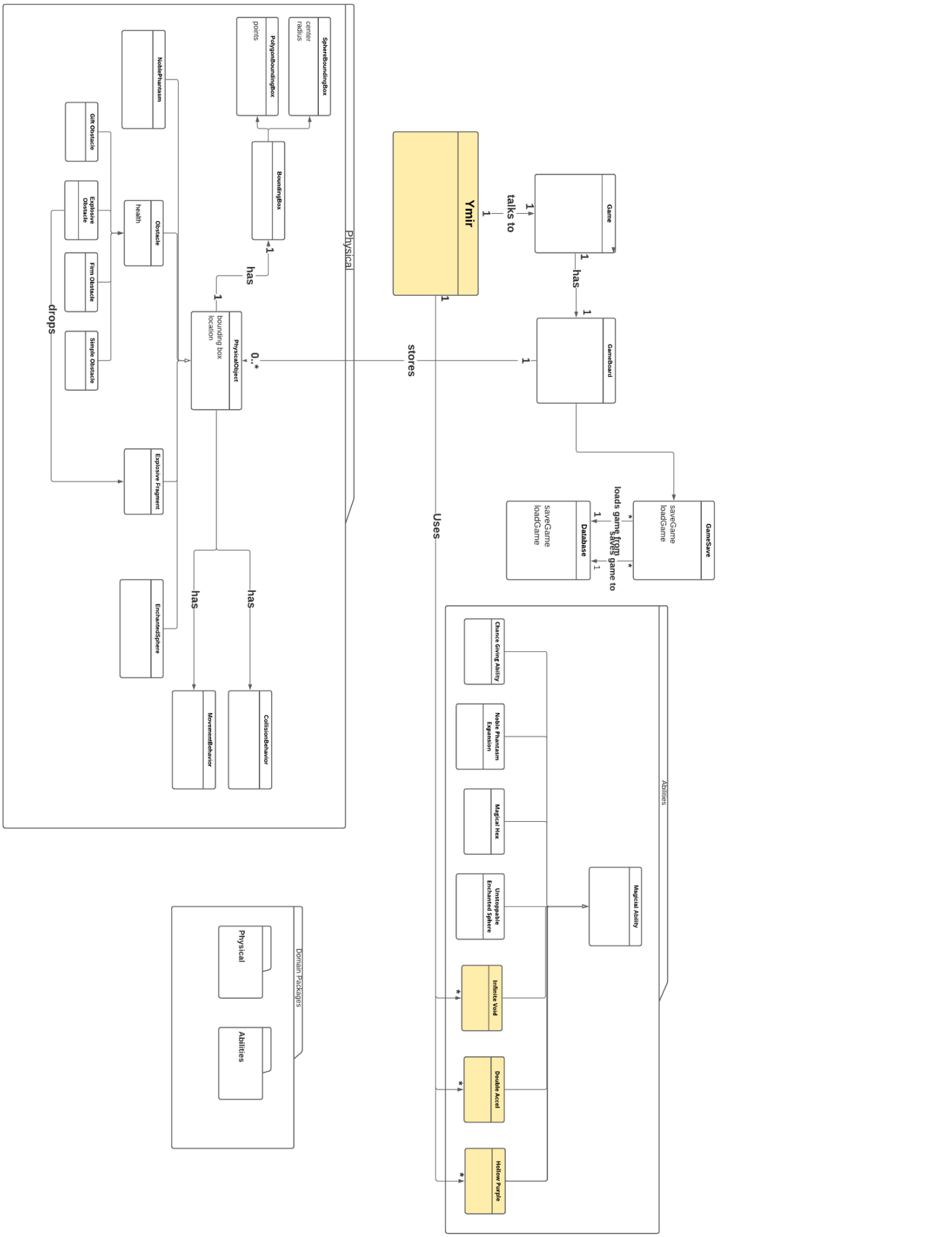
Extensions:

*a At anytime, player performs Pause the Game

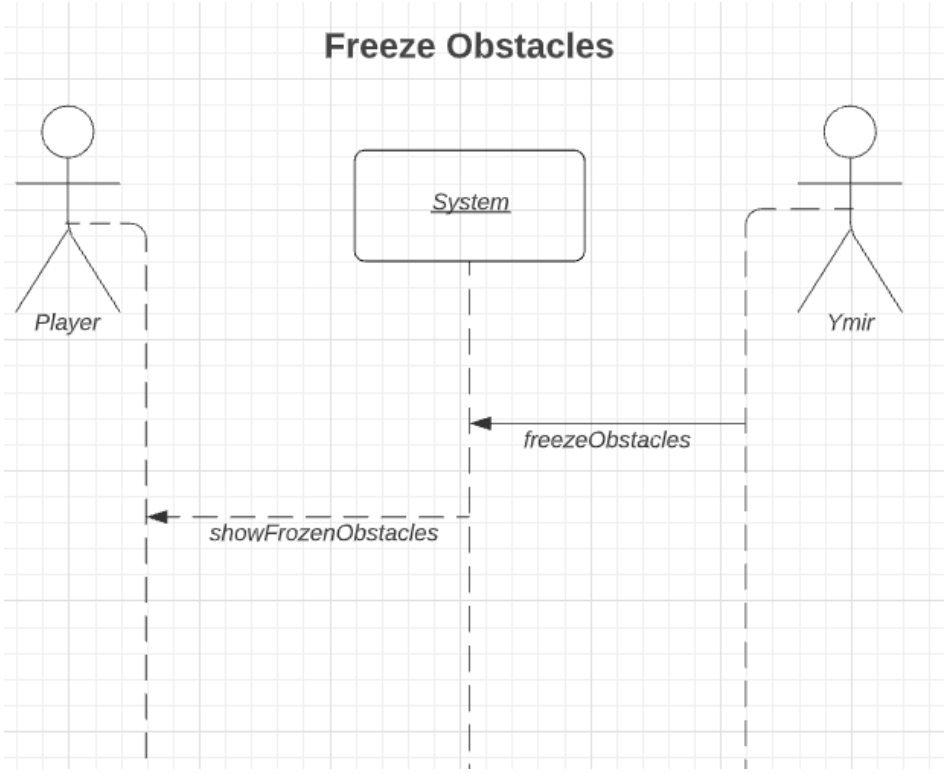
1. Player performs Resume the Game
2. Player performs Save the Game
3. Player goes back to start
4. Player exits the game

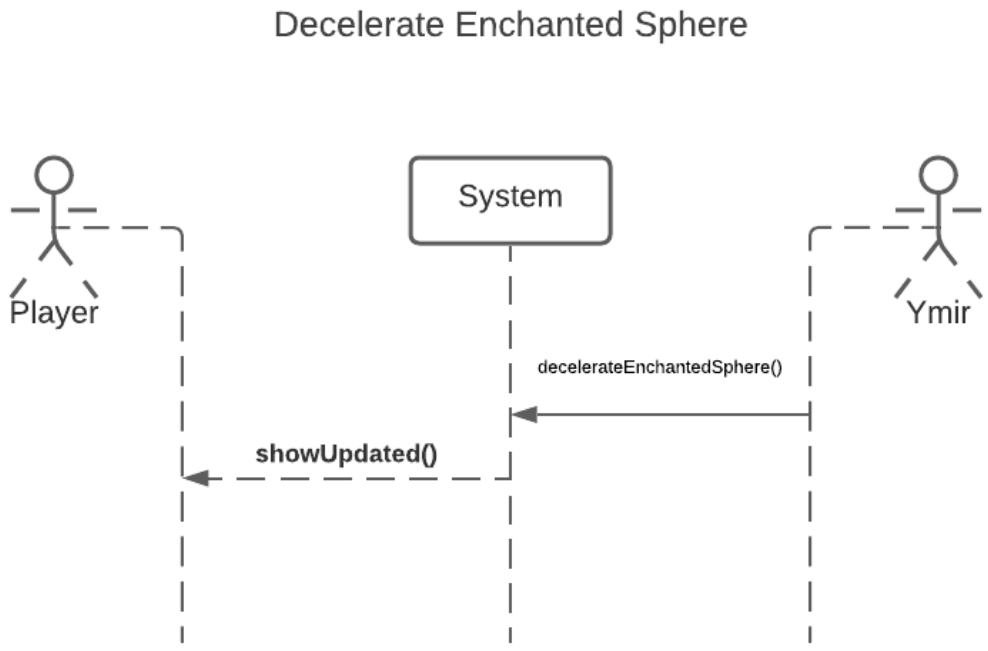
Special Requirements:Game is installed on the machine.
Technology and Data Variations List:
Frequency of Occurrence: On every 30 seconds Ymir wins coin toss(50%) and chooses hollow purple among 3 options.
Miscellaneous:

Domain Classes



System Sequence Diagram





Operation Contracts:

Contract CO12: Freeze Obstacles:

Operation:	freezeObstacles(freezedObstaclesList)
Cross References:	Use Case 14: Equip Infinite Void
Preconditions:	The obstacles in the obstaclesList exists in the game
Postconditions:	freeze method of the obstacles in the freezedObstaclesList was called. The obstacles were set to be unbreakable by the enchanted sphere. The freezed obstacles were set to be destroyable by the unstoppable ball. if hit by the unstoppable, the health was set to be decremented by one.

Contract CO13: Decelerate Enchanted Sphere:

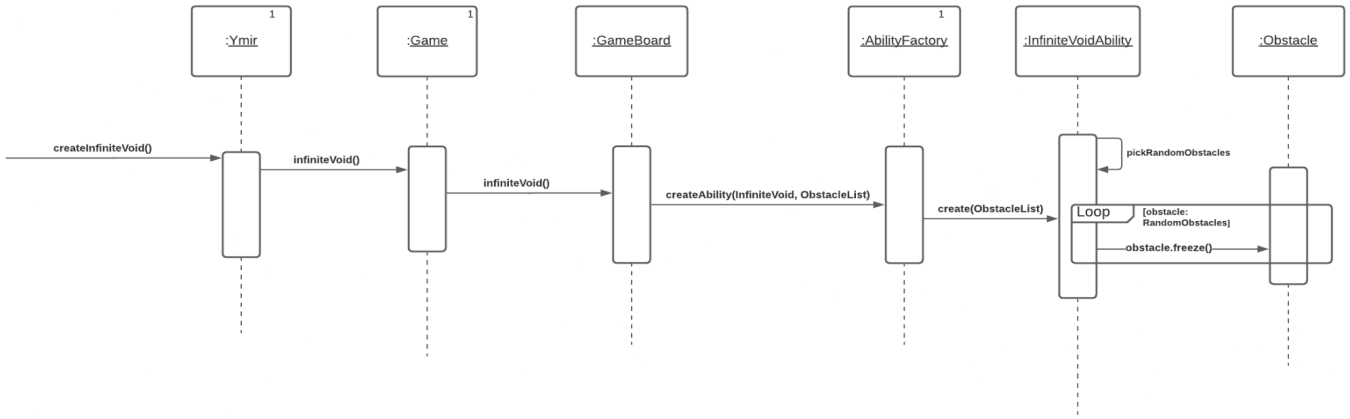
Operation:	decelerateEnchantedSphere()
Cross References:	Use Case 15: Equip Double Accel
Preconditions:	Double axel ability is activated
Postconditions:	The speed of the enchanted sphere was reduced by half.

Glossary

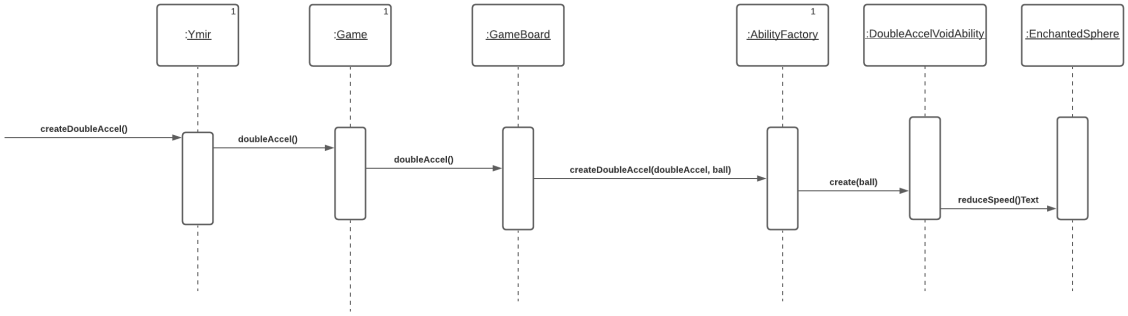
Term	Definition and Information	Format	Validation Rules	Aliases
Ymir	A sorcerer that will coin toss in every 30 seconds and if it is a success activate one of the 3 magical abilities (Infinite void, double accel, and hollow purple) to make game more difficult for player			
Infinite Void (Magical Ability)	A powerup which is activated by Ymir that choses 8 obstacles (if there are less obstacles in the game all of them) and freezes them.		Once activated, it only lasts for 15 seconds.	
Double Accel (Magical Ability)	A powerup which is activated by Ymir that speeds down the enchanted sphere by half.		Once activated, it only lasts for 15 seconds.	
Hollow Purple (Magical Ability)	A powerup which is activated by Ymir that puts 8 purple colored empty obstacles.		Once activated, it only lasts for 15 seconds.	

Interaction Diagrams

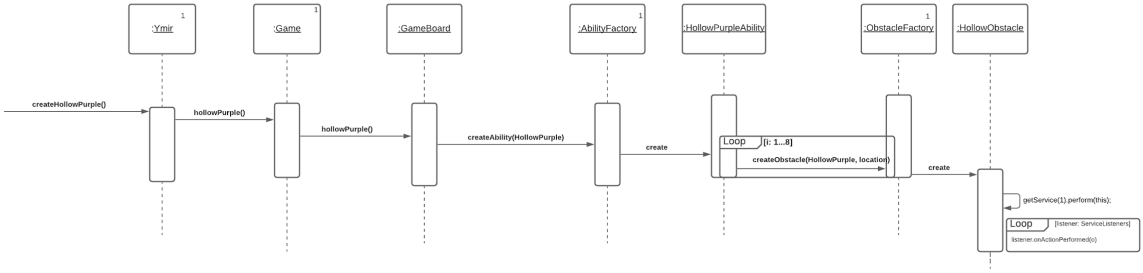
SD.1: Equip Infinite Void



SD.2: Equip Double Accel



SD.3: Equip Hollow Purple



Discussion of Design Patterns

Controller Pattern

In accordance with controller pattern (i.e Game class), Ymir only with the Game class to activate its abilities. This approach is in conformity with the Game class being the controller. In other words, when a request is made from the UI to activate a magical ability, this action is delegated to the Game class which further does the task division between the classes in domain. Hence, this way the model-view separation is maintained. A key point in using this pattern is to carefully assign the message received from the UI to responsible classes which is handled by the Game class.

Factory Pattern

For creating Ymir Abilities, we used factories. That made it possible to easily create those abilities with a given type. An advantage of this pattern is that if at some point we decide to improve the game and add new features to the classes which use factory design patterns, we can do so easily, since the desired features will only have to implement the interface class and its methods based on the features functionality. A minor disadvantage but worth mentioning in this approach may be the fact that while creating the instances if there are many different abilities, the code will consist of an inevitable amount of if and else statements.

Singleton Pattern

As a team we planned to adapt a singleton pattern for Ymir class since we only need one instance of it throughout the project. Therefore, for the Ymir class as well as our factory class (AbilityFactory) we used the Singleton Pattern. This helps us achieve high cohesion and low coupling. However, one of the disadvantages of this approach is that later in our project we are going to use unit testing to test our code, and the singleton pattern makes it difficult to recognize dependency chains which leads to a more difficult unit testing process.

Class Diagram

You can find as a second document on Blackboard submission and here: