Assignment 1.2 – UML in practice

1.2.1

The composition and aggregation are forms of relationships between objects.

A composition is a bound relationship between two objects, the parent and the child, the parent can exist without the child but the child can not exist without the parent.

An aggregation is a relationship between two objects which can be split at any time. They are in no way dependent on each other.

Composition

Classes	Explanation
GameController GameLoop	The GameController controls all UI within the game, including the GameLoop it initializes the scene in which the game can be played. The GameLoop is a class which loops over every frame and updates all locations/properties of the objects spawned within the GameController. It is possible for the GameController to exist without a GameLoop but the GameLoop is dependent on the GameController, thus a composition.
Vector IntersectionPoint	The Vector class is used for calculations on the movement of certain GameObjects. The IntersectionPoint class is used to check for an intersection between the vector and a position. The intersectionPoint is useless without a Vector because the Vector gives a line on which intersectionPoint can look for an intersection between the Vector and an object. IntersectionPoint is dependent on the Vector,thus a composition
Bubble Point	The Bubble class is used for spawning a bubble which the player has to pop in the game. The Point class is used to instantiate a point in a field. The Bubble can’t be created without the use of a Point because the bubble needs a point at which it should spawn. Bubble is dependent on the point, thus a composition
Bubble Vector	The Bubble class is used for spawning a bubble which the player has to pop in the game.

	The Vector class is used for calculations on the movement of certain GameObjects. The bubble is a GameObject which needs direction, without a Vector it is just a ball. The bubble is dependent on a vector, thus a composition.
GameLoop(IDraw) GameObjects	The GameLoop updates all objects within the game and draws them with the help of IDraw, IDraw is a drawing interface which gives a limited set of methods to the game container classes to indirectly draw things on the screen GameObjects contains all GameObjects which it should use and draw within the game. Without IDraw the GameObjects class is just a big container of objects, with IDraw the GameObjects class can draw the objects onto the scene to be used in the game. GameObjects is a container of all objects which should draw them onto the stage when they are needed, they can't be drawn without IDraw which is why GameObjects is dependent on GameLoop(IDraw). This makes GameObjects and GameLoop(IDraw) a composition.
Scene Keyboard	The scene class is the container for all content in a scene graph. It makes it possible to show everything on the screen The keyboard class is used to listen to key-events which control the player. The keyboard has to listen to a certain scene, without a scene the keyboard class is useless. This makes Scene and Keyboard a composition.
Point Line	A Line is used for a collision functionality Points are locations on the screen The line is created using the Point class, without two points you can't make a line. This makes Line dependent on Points, a composition.
Keyboard Player	The keyboard is used to control the player The player is the character in the game which should be manipulated by the user of the game. If there is no keyboard the player can't be moved by the user thus the game is unbeatable. This makes Player dependent on Keyboard, a composition.

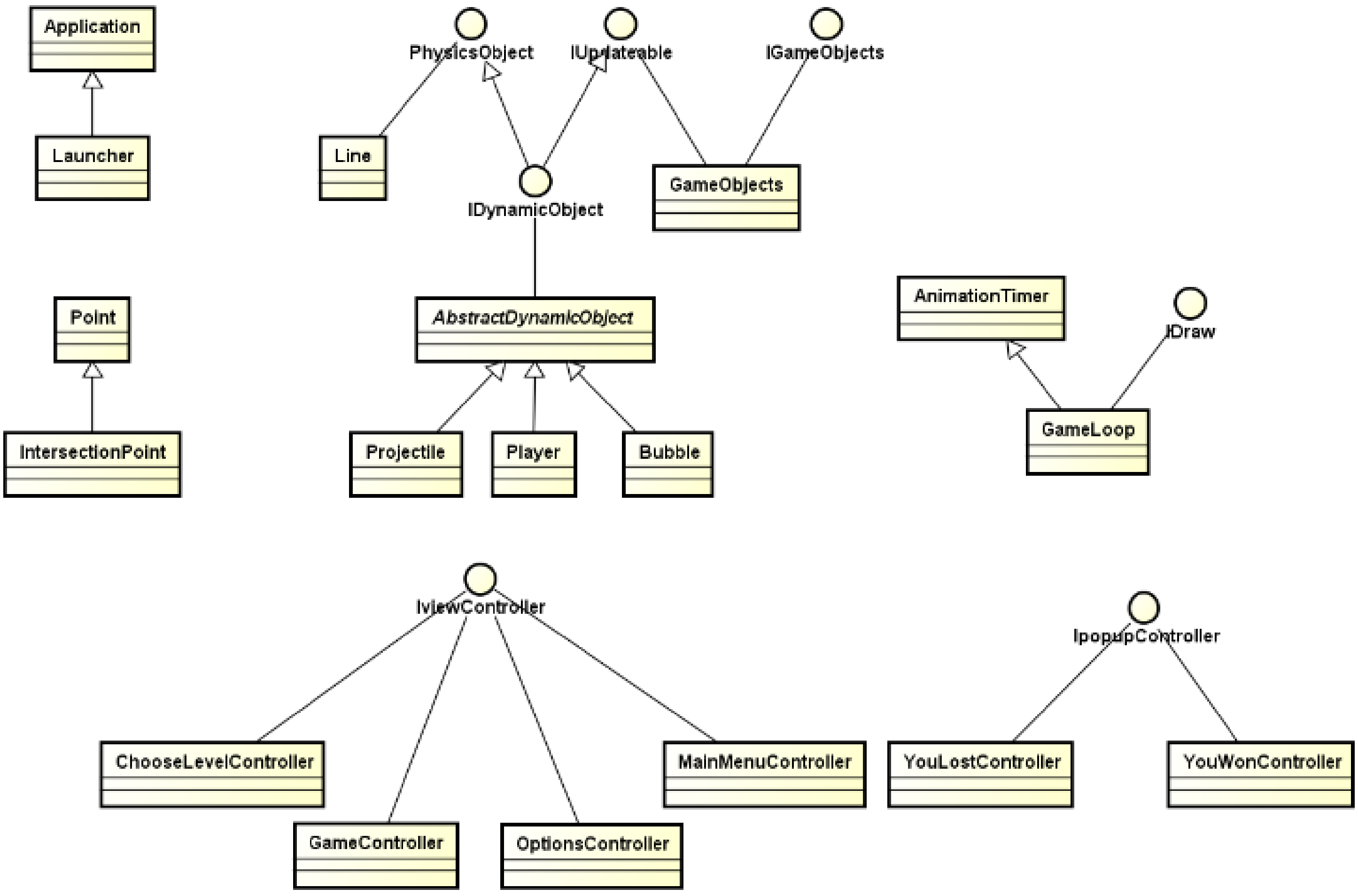
Aggregation

GameObjects Bubble	GameObjects contains all GameObjects which it should use and draw within the game. Bubbles are circles used as bubbles in the game with position speed and direction. The GameObjects class can store the bubbles to use in the game but it isn't necessary for bubbles to be there. bubbles are not bound to GameObjects in any way so you can create a bubble without using GameObjects. This makes the relation between GameObjects and Bubble an aggregation.
GameObjects Player	GameObjects contains all GameObjects which it should use and draw within the game. The player is the character controlled by the user to play the game. The GameObjects class can store the player to deploy him into the game but it is not necessary that there is a player for GameObjects to work. The player can be created without any connection to GameObjects. This makes the relation between GameObjects and Player an aggregation.
GameObjects Projectile	GameObjects contains all GameObjects which it should use and draw within the game. The Projectile is used to create a line which has the property that it can be shot by a player and can burst bubbles. The GameObjects class can store a projectile to deploy it into the game but It's not necessary. The Projectile can be created without any connection to GameObjects. This makes the relation between GameObjects and Projectile an aggregation

1.2.2

There are no Parameterized classes in our source code.
Parameterized classes are very useful when used in testing, otherwise it

1.2.3



Launcher extends Application to create a JavaFX Life-Cycle
Type: IS-a

GameLoop extends AnimationTimer to call a timer which is called in each frame.
Type: IS-a

GameLoop implements IDraw make it possible to indirectly draw objects onto the screen
Type: IS-a

Projectile, player and Bubble extend AbstractDynamicObject to avoid duplicate code between all three dynamic objects.
Type: Polymorphism

AbstractDynamicObject implements IDynamicObject to make it possible for objects to move
Type:IS-a

IDynamicObject extends PhysicsObject to use objects with physics
Type:IS-a

Line implements PhysicsObject to use objects with physics
Type: IS-a

GameObjects implements IUpdateable to make it able for GameLoop to update.
Type: IS-a

IDynamicObject extends IUpdateable to make it for GameLoop to update
Type: IS-a

GameObjects implements IGameObjects to define which methods may access objects within the GameObjects class
Type: IS-a

ChooseLevelController, GameController, OptionsController and MainMenuController implement IviewController to make it possible to use them as scenes.
Type: PolyMorphism

YouLostController and YouWonController implement IpopupController to make it possible to use popupwindows.
Type: PolyMorphism