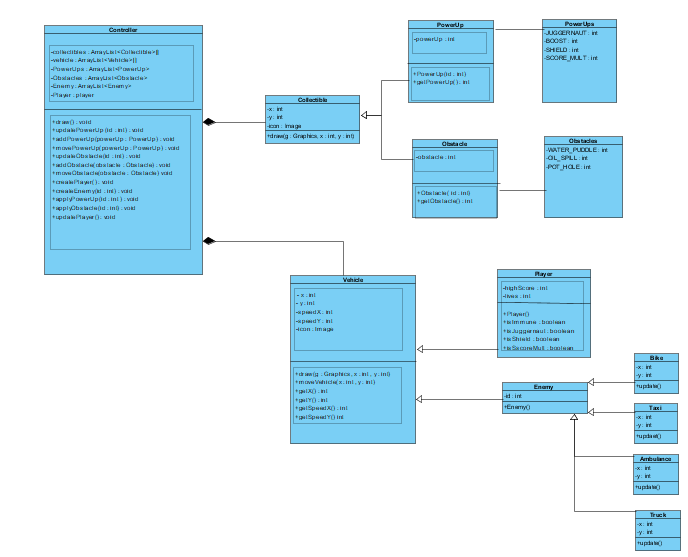
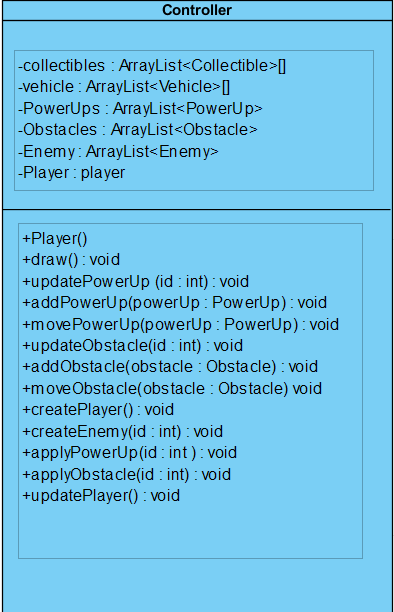
# Game Entities Subsystem Entities

Entities subsystem is the susbsystem that contains domain specific objects. This subsystem consists of 10 classes. Player car and enemy car inherit from Vehicle abstract class. Also additional features of game inherit from Collectible abstract class. Façade class of this subsystem is Controller class, which controls the objects of the system and modify them according to game. Because of our Façade design pattern, dividing systems into subsystems minimizes the association of these subsystems. In the figure Game Entities subsystem visualized. Each class of will be specifically explained in this section.



# Controller Class



* Controller class is the Façade class of this subsystem, thus, this class’s operations modifies and creates entities of the game. Also communication of the objects occurs here.

# Attributes

**private ArrayList<Collectible> collectibles:** this array list references to the all collectibles of the game.

**private ArrayList<Vehicle> vehicle:** this array list holds all vehicle objects of the game.

**private ArrayList<PowerUp> PowerUps:** this array represents all power ups on the screen.

**private ArrayList<Obstacle> Obstacles:** this array represents all obstacles on the screen.

**pivate ArrayList<Enemy> Enemy:** this array holds all enemy objects on the screen.

**private Player player:** Since there will be only one player, this attribute represents player itself.

# Methods

**public void draw():** draws all the vehicle and collectible objects on the main panel and updates them.

**public void updatePowerUp(int id):** this methods updatespower up’s coordinates which are on the screen according to their ID’s.

**public void addPowerUp(PowerUp powerUp):** this method adds created power up to PowerUp array list in order to be on the screen soon.

**public void createPowerUp(PowerUp powerUp):** creates new power up

**public void updateObstacle(int id):** this method updates coordinates of the obstacles which are on the screen

**public void addObstacle(Obstacele obstacle):**this method adds created obstacle to the obstacle list in order to be presented on the screen.

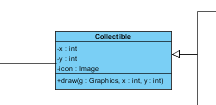
**Public void createObstacle(Obstacle obstacle):** creates new obstacle.

**public void createEnemy(int id):** creates new enemy.

**public void createPlayer():** constructs new player.

**Public void updatePlayer():** updates players controls as user input taken in terms of moving.

# Collectible Class



* This class is an abstract class for image and 2D coordinates for collectibles.

# Attributes

**private int x:** this attribute is x-axis value for collectibles.

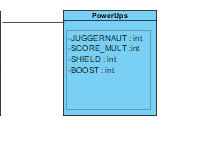
**private int y:** this attribute is y-axis value for collectibles.

**private Image icon:** this attribute is for how a collectible represented on the screen.

# Methods

**public void draw(Graphics g, int x, int y):** This method draws the collectible object to proper location on the main screen. For instance, generating water puddle on the screen but any random place.

# PowerUps



# Attributes

**private int JUGGERNAUT**

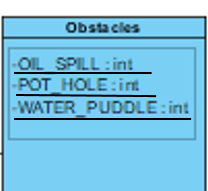
**private int SCORE\_MULT**

**private int SHIELD**

**private int BOOST**

* These attributes define the kinds of power ups.

# Obstacles



# Attributes

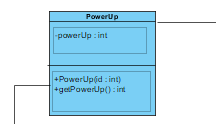
**private int OIL\_SPILL**

**private int POT\_HOLE**

**private int WATER\_PUDDLE**

* These attributes define the kinds of obstacles.

# PowerUp class



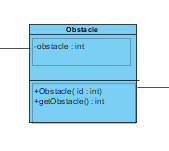
# Attributes

**private int powerUp:** this attribute holds the id as the kind of power up.

# Constructors

**public PowerUp(int id):** constructs a power up object according to its kind.

# Obstacle Class



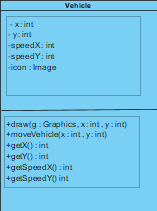
# Attributes

**private int obstacle:** this attribute holds for the kind of obstacle.

# Constructors

**Public Obstacle(int id):** constructs an obstacle object accordingly its kind.

# Vehicle Class



* This abstract class changes the objects attributes, in other words, as objects states changed, this class represents those states features in terms of game.

# Attributes

**private int x:** this attribute holds the x-axis value for the vehicles.

**private int y:** this attribute holds the y-axis value for the vehicles.

**private int speedX:** how fast the vehicle’s x-axis value changes.

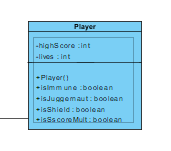
**private int speedY:** how fat the vehicle’s y-axis value changes.

# Methods

**public void draw(Graphics g, int x, int y):** This method draws the vehicle object to proper location on the main screen. For instance, generating enemy cars on the screen but any random place. Also locates players location according to user.

**public void moveVehicle(int x, int y):** updates coordinates of the vehicles according to game.

# Player Class



# Attributes

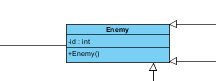
**private int highScore:** score of the player that he makes during the game.

**private int lives:** remainin lives of the player during the game.

# Constructor

**public Player():** constructs player object and initializes its attributes.

# Enemy Class



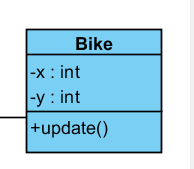
# Attributes

**private int id:** defines kind of enemy.

# Constructors

**public Enemy():**  constructs a proper enemy object according to game.

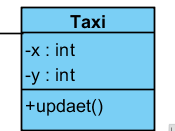
# Bike Class



# Methods

**Public void update():** updates x and y axis values of the bike object according to game

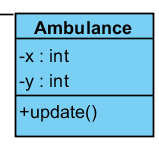
# Taxi Class



# Methods

**Public void update():** updates x and y axis values of the taxi object according to game

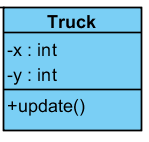
# Ambulance Class



# Methods

**Public void update():** updates x and y axis values of the ambulance object according to game

# Truck Class



# Methods

**Public void update():** updates x and y axis values of the truck object according to game