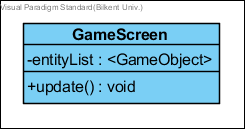
## C:\Users\Asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Entity Subsystem Class Diagram Upgraded2.jpg**Game Entities Package**

In the Game Entities Subsystem, our all game entity classes are grouped together with their relations. We have GameScreen, Vector2f, GameObject(abstract), DynamicGameObject(abstract), Stats, Enemy, Inventory, Player, Bullet, BouncyBullet, Final, Midterm, Quiz, Assignment, Lab, Bug,Chest, SeniorChest, JuniorChest, SaphomoreChest, FreshmenChest, Coin, Item, StandardItem, UltraRareItem, RareItem, HackerItem, Bonus, Key, PowerUp, ExtraTime, SlowTime, BouncyBullets, BulletBlast, DamageIncrease classes and Drawable, Shooter, Collectable interfaces. These classes and interfaces will be explained in detail, in this section.

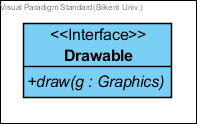
**GameScreen Class**



GameScreen class holds the game entities and upgrades them on the screen.  ***Attributes:***  
**private ArrayList<GameObject>entityList:** Holds the entity(GameObjects) objects of the game.

***Methods:*public void update():** calls the update method of every entity object.

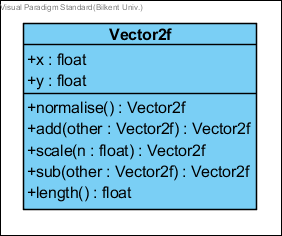
**Drawable (interface)**

****

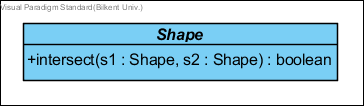
Drawable interface is provided for all game objects …

***Methods:*public void draw():** overridden by gameObjects a simple draw method to visualize objects.

**Vector2f1**

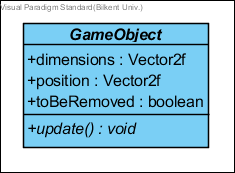
  
A library provided representation of two dimensional vectors[[1]](#footnote-1)

**Shape2**

****

A library provided class. The description of any 2D shape that can be transformed. The points provided approximate the intent of the shape.  
***Attributes*public boolean intersect(Shape s1, Shape s2):** Check if this shape intersects with the other shape.

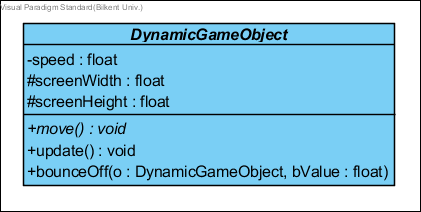
**GameObject Class (abstract)**

****

GameObject class is a base class for all entities in our game. ***Attributes:*public Vector2 dimensions:** rectangle sizes of a GameObject.**public Vector2 position:** thecenter position of a GameObject.  
**public boolean toBeRemoved:** it is a flag for GameMaster to check the object is to be removed or not before the next game cycle.

***Methods:***  
**public void update():** updates the information accordingly to the internal commands.

**DynamicGameObject (Abstract)**

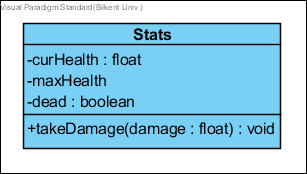
****

DynamicGameObject class is an abstract class for all movable game objects.

***Attributes:***  
**private float speed:** is the current speed of the dynamic game object.  
**protected float screenWidth:** is the width of the screen window in pixels.  
**protected float screenHeight:** is the height of the screen window in pixels.

***Methods:*private void move():** updates the location by speed.  
**private void update():** It updates the position and the state of the object in every game loop. This method does not return anything.  
**private bounceOff(DynamicGaneObject object, flaot bounceValue):** This method is to separate to colliding the dynamic game object by setting their positions bounce Value away from each other. This simply gives them visualisation of two game object bouncing from each other. This is usually called when two objects collides.

**Stats Class**

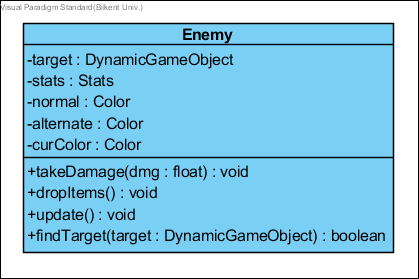
******

Stats class holds the variables abaout the health status of Player and Enemy objects.

***Attributes:*  
private float curHealth:** is the current health of game object(Player or Enemy). **private float maxHealth:** is the maximum health of game object(Player or Enemy).  
**private boolean dead():** is a flag value for removal from the gameScreen.

***Methods:*  
public void takeDamage(float damage):** decrements the curHealth by damage value. It controls the dead flag is true or not by checking curHealt is less than zero.

**Enemy Class**

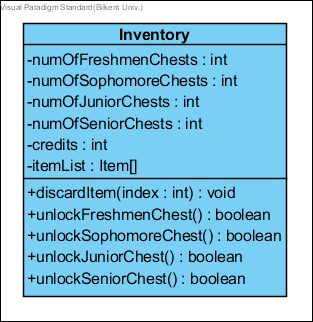
******

This class constructs the Enemy in our game. Enemy class is a parent class of all enemy types which are Final, Quiz, Midterm, Assignment, Lab, and Bug.

***Attributes:*  
private DynamicGameObject target:** the target of the enemy which will be attacked and possibly damaged. **private Stats stats:** stats objects for the enemy.  
**private Color normal:** it is a color when the object is initially created.  **private Color alternate:** it is a temporary color when the game object takes damage. **private Color curColor:**it is a current color of game object.

***Methods:*  
public void takeDamage(float dmg):** calls the enemy’s stat objects, the amount of Damage that will be applied to Enemy if enemy is killed it will be marked for removal.(calls stats.takeDamage(dmg)) **public void dropItems():** Enemy drops items before getting removed because of death.  
**public void update():** It updates the position and the state of the object in every game loop. This method does not return anything. **public boolean findTarget(DynamicGameObject target):** Enemy seeks player as target.

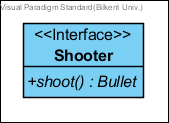
**Inventory Class**

****

Inventory class is to hold the credits, the number of chests, and most importantly the list of items.

***Attributes:*  
private intnumOfFreshmenChests:** the number of chests of type ‘freshmen’ which are collected and not opened. **private intnumOfSophomoreChests:** the number of chests of type ‘sophomore’ which are collected and not opened. **private intnumOfJuniorChests:** the number of chests of type ‘junior’ which are collected and not opened. **private intnumOfSeniorChests:** the number of chests of type ‘senior’ which are collected and not opened. **private intcredits:** number of coins collected. **private Item[] itemList:** 5 items that are owned by the player.

***Methods:*  
public void discardItem(intindex):** removes the item at the given index. **public booleanFreshmenChest():** instantiates and opens a freshmenChest. **public booleanSophomoreChest():** instantiates and opens a SophomoreChest. **public booleanJuniorChest():** instantiates and opens a JuniorChest. **public booleanSeniorChest():** instantiates and opens a SeniorChest.

**Shooter(interface)  
**

Shooter is an interface for every shooting game object.

***Methods:*  
public Bullet shoot():** An abstract shoot method for shooting player and enemies.

1. In the feedback It was advised us to name our classes “**wisely**” although we switched libraries in between and this particular class is no more our implementation the name is roughly the same(It used to be Vector2 now Vector2f). We also would like to inform you that representing 2D points/Vectors in programming with the name “**Vector2**” is common convention.

   Here are some links to popular Game Engines and Java Game Libraries:

   Unity Engine **Vector2**: <https://docs.unity3d.com/ScriptReference/Vector2.html>

   Unreal Engine F**Vector2D**: <https://docs.unrealengine.com/latest/INT/API/Runtime/Core/Math/FVector2D/>

   Hero Engine(has only 3D vectors) **Vector**3: <http://hewiki.heroengine.com/wiki/Vector>

   Java Light Weight Game Library’s AI**Vector2D**: <https://javadoc.lwjgl.org/>

   Slick 2D’s **Vector2**f: <http://slick.ninjacave.com/javadoc/org/newdawn/slick/geom/Vector2f.html> (our library)

   2 <http://slick.ninjacave.com/javadoc/org/newdawn/slick/geom/Shape.html> [↑](#footnote-ref-1)