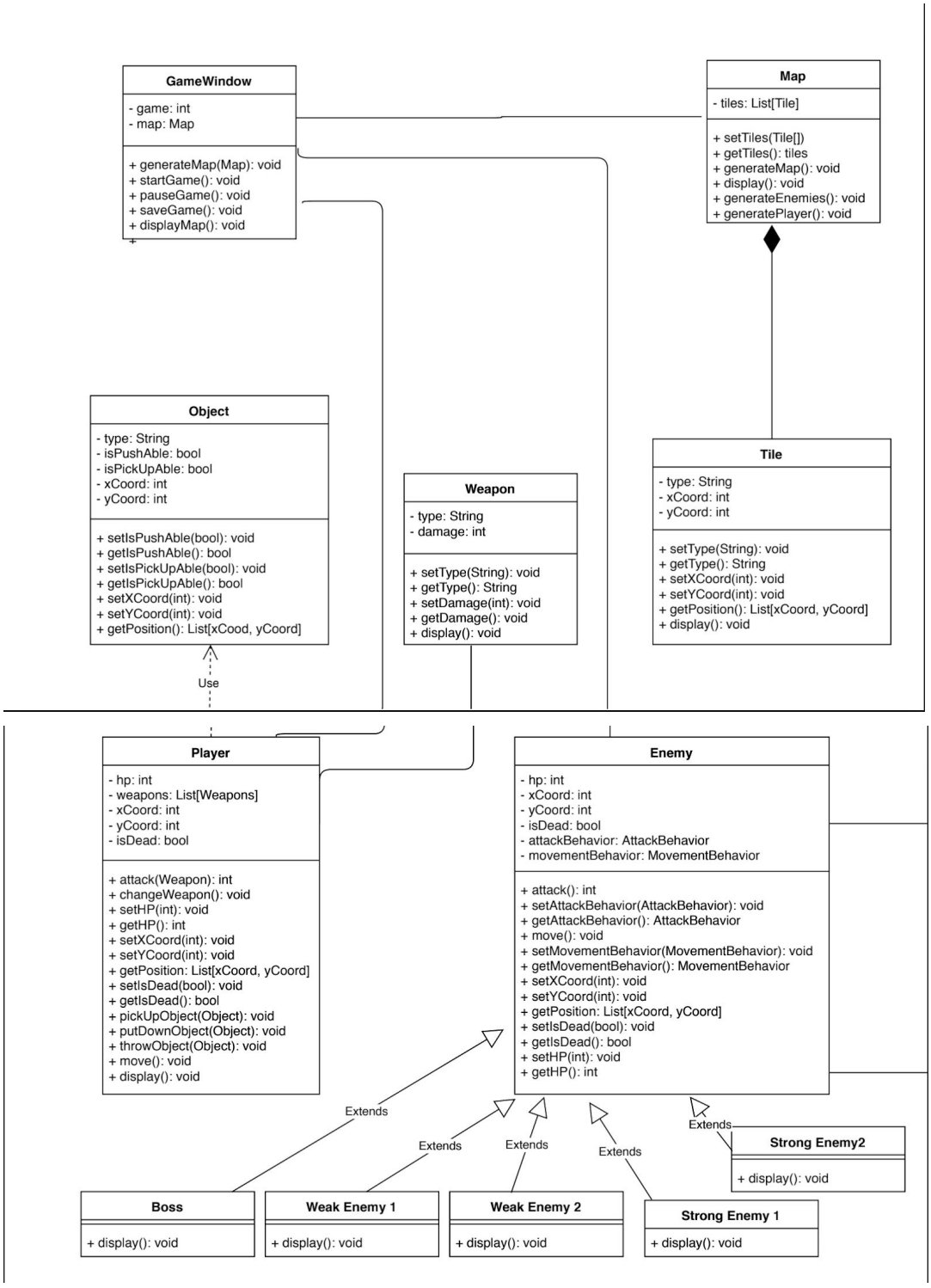
Name of project: OneDungeon

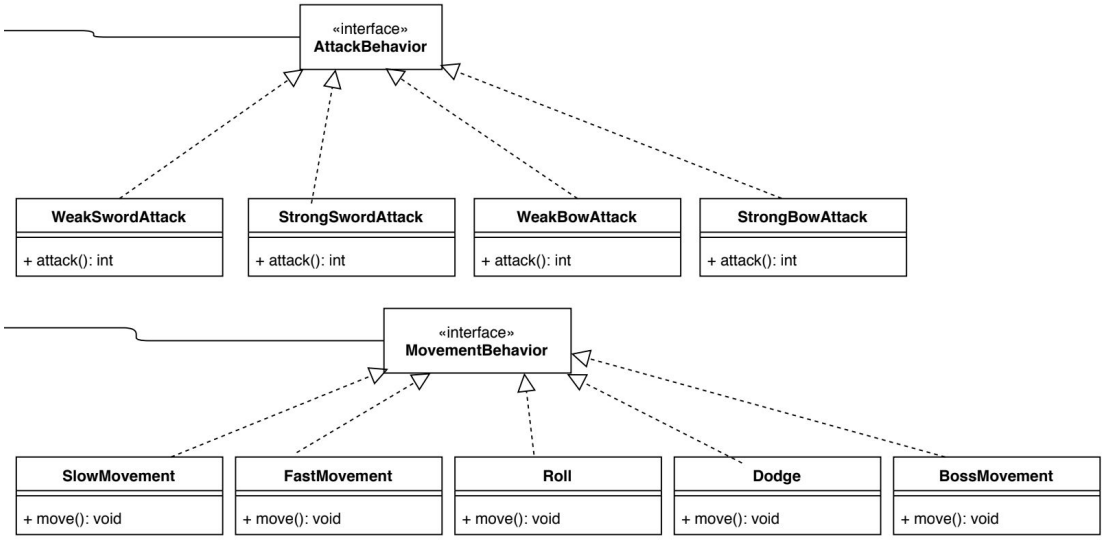
Name of team members: Yun-Ting Chen, Kyle McDevitt

Final state of the system (what were implemented, what were not?):
In our final system state, we have a start menu that lets you enter the game or load a game if a previous save exists. We also implemented controls for the player to move, attack, and take damage. We implemented controls for enemies to chase after the player, take damage and die. We also implemented controls that allow the player to take damage from certain objects in the game, such as spikes. We implemented a collectible health item as well. We were also able to implement a pause screen with the options to resume, save the game, or quit. We were not able to fully implement a strategy pattern for enemy behavior and we were also not able to implement multiple weapons to choose from because getting familiar with Unity and building a fully functional level took a tremendous amount of time. We also were not able to make all objects in the game fully interactive because of time constraints.

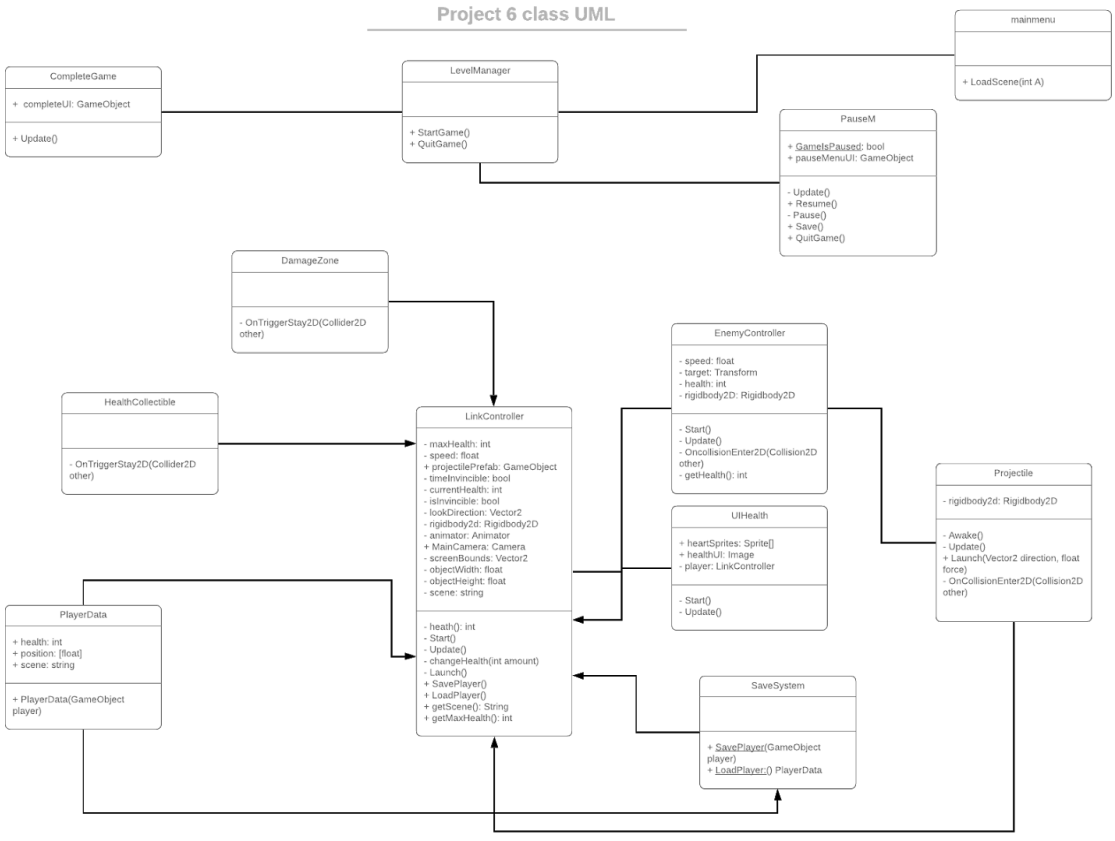
UML Digram:

- Project 4:


GameWindow
- game: int
- map: Map
+ generateMap(Map): void
+ startGame(): void
+ pauseGame(): void
+ saveGame(): void
+ displayMap(): void
Map
- tiles: List[Tile]
+ setTiles(Tile[])
+ getTiles(): tiles
+ generateMap(): void
+ display(): void
+ generateEnemies(): void
+ generatePlayer(): void
Object
- type: String
- isPushAble: bool
- isPickUpAble: bool
- xCoord: int
- yCoord: int
+ setIsPushAble(bool): void
+ getIsPushAble(): bool
+ setIsPickUpAble(bool): void
+ getIsPickUpAble(): bool
+ setXCoord(int): void
+ setYCoord(int): void
+ getPosition(): List[xCood, yCoord]
Weapon
- type: String
- damage: int
+ setType(String): void
+ getType(): String
+ setDamage(int): void
+ getDamage(): void
+ display(): void
Tile
- type: String
- xCoord: int
- yCoord: int
+ setType(String): void
+ getType(): String
+ setXCoord(int): void
+ setYCoord(int): void
+ getPosition(): List[xCoord, yCoord]
+ display(): void
Use
Player
- hp: int
- weapons: List[Weapons]
- xCoord: int
- yCoord: int
- isDead: bool
+ attack(Weapon): int
+ changeWeapon(): void
+ setHP(int): void
+ getHP(): int
+ setXCoord(int): void
+ setYCoord(int): void
+ getPosition: List[xCoord, yCoord]
+ setIsDead(bool): void
+ getIsDead(): bool
+ pickUpObject(Object): void
+ putDownObject(Object): void
+ throwObject(Object): void
+ move(): void
+ display(): void
Enemy
- hp: int
- xCoord: int
- yCoord: int
- isDead: bool
- attackBehavior: AttackBehavior
- movementBehavior: MovementBehavior
+ attack(): int
+ setAttackBehavior(AttackBehavior): void
+ getAttackBehavior(): AttackBehavior
+ move(): void
+ setMovementBehavior(MovementBehavior): void
+ getMovementBehavior(): MovementBehavior
+ setXCoord(int): void
+ setYCoord(int): void
+ getPosition: List[xCoord, yCoord]
+ setIsDead(bool): void
+ getIsDead(): bool
+ setHP(int): void
+ getHP(): int
Extends
Extends
Extends
Extends
Extends
Boss
+ display(): void
Weak Enemy 1
+ display(): void
Weak Enemy 2
+ display(): void
Strong Enemy 1
+ display(): void
Strong Enemy2
+ display(): void


«interface»
AttackBehavior
WeakSwordAttack
+ attack(): int
StrongSwordAttack
+ attack(): int
WeakBowAttack
+ attack(): int
StrongBowAttack
+ attack(): int
«interface»
MovementBehavior
SlowMovement
+ move(): void
FastMovement
+ move(): void
Roll
+ move(): void
Dodge
+ move(): void
BossMovement
+ move(): void

- Project 6:


Project 6 class UML
mainmenu
+ LoadScene(int A)
CompleteGame
+ completeUI: GameObject
+ Update()
LevelManager
+ StartGame()
+ QuitGame()
PauseM
+ GameIsPaused: bool
+ pauseMenuUI: GameObject
- Update()
+ Resume()
- Pause()
+ Save()
+ QuitGame()
DamageZone
- OnTriggerStay2D(Collider2D other)
EnemyController
- speed: float
- target: Transform
- health: int
- rigidbody2D: Rigidbody2D
- Start()
- Update()
- OncollisionEnter2D(Collision2D other)
- getHealth(): int
HealthCollectible
- OnTriggerStay2D(Collider2D other)
LinkController
- maxHealth: int
- speed: float
+ projectilePrefab: GameObject
- timeInvincible: bool
- currentHealth: int
- isInvincible: bool
- lookDirection: Vector2
- rigidbody2d: Rigidbody2D
- animator: Animator
+ MainCamera: Camera
- screenBounds: Vector2
- objectWidth: float
- objectHeight: float
- scene: string
- heath(): int
- Start()
- Update()
- changeHealth(int amount)
- Launch()
+ SavePlayer()
+ LoadPlayer()
+ getScene(): String
+ getMaxHealth(): int
UIHealth
+ heartSprites: Sprite[]
+ healthUI: Image
- player: LinkController
- Start()
- Update()
Projectile
- rigidbody2d: Rigidbody2D
- Awake()
- Update()
+ Launch(Vector2 direction, float force)
- OnCollisionEnter2D(Collision2D other)
PlayerData
+ health: int
+ position: [float]
+ scene: string
+ PlayerData(GameObject player)
SaveSystem
+ SavePlayer(GameObject player)
+ LoadPlayer:() PlayerData

Since our player only has one way to attack enemies, one movement behavior, one weapon. Therefore, the new diagram does not have weapon selection, different attack behaviors, and movement behaviors. The diagram on project 4 does not include quit, resume, restart, and save game options in our scenes, we included these functions as we proceed. Pattern we used in this game design is MVC. The program sends data to the view so that the player the player gets to control the UI to attack, save game, and quit.

Third-Party code:

1. This is a tutorial of how to create pause, resume and quit game functions. PauseM script are from this tutorial

https://www.youtube.com/watch?v=JivuXdrIHK0

2. This is a Gameover page tutorial. Used this tutorial to check for player's health and load the game over scene

https://www.youtube.com/watch?v=YdxYdHidCkE

   a. Implementation details for checking if the player is dead is original code.

3. Scripting API to look for object with specific tag

https://docs.unity3d.com/ScriptReference/GameObject.FindGameObjectsWithTag.html

4. Some code for the EnemyController was take from https://learn.unity.com/tutorial/world-interactions-damage-zones-and-enemies?courseId=5c5c1e08edbc2a5465c7ec01&projectId=5c6166dbedbc2a0021b1bc7c
   a. Enemy AI and damage constraints were original.
5. Code for health collectibles taken from https://learn.unity.com/tutorial/world-interactions-collectibles?courseId=5c5c1e08edbc2a5465c7ec01&projectId=5c6166dbedbc2a0021b1bc7c
6. Code for save system taken from https://www.youtube.com/watch?v=XOjd_qU2Ido
   a. Scene saving mechanism was original

Statement on OOAD Process

1. One issue was underestimating how difficult it would be to stick to the original design that was laid out in Project 4. Because we had to learn a new framework (Unity) and a new language to write scripts for our game (C#), we had to steer away from a lot of our original design plans.
2. Putting effort into making our code extensible and modifiable limited our ability to make a lot of process in the finished product, but this would allow us to make easier changes down the road if we should so choose.
3. Planning in advance did allow us to reach a semi-concrete idea of what we wanted our game to do and what use cases were essential to the game itself,

and if we had a more general, abstract idea, it would have been much more difficult to make the progress that we did.