Notes:
All abstract methods in the parent class are inherited and overridden in the child classes

Wave Event

- waveNumber: int <<get>>
- waveProgress: String <<get>>
- eventType: String
- time: int

+ WaveEvent(WaveData String)
+ updateWaveProgress(): void
+ spawnWave(): void
+ updateWave(path: List): void
+ endWave(): void
+ updateTimescale(timescale: int): void

1..*

Level

- levelNum: int <<get>>
- map: TiledMap <<get>>
- path: List <<get>>
- timescale: int <<get>>

+ Level(levelNum: int)
+ newLevel(): void
+ readWaveData(): void
+ startWave(): void
+ updateWave(): void
+ drawMap(): void
+ increaseTimescale(): void
+ decreaseTimescale(): void

1..*

ShadowDefend

- level: int

+ ShadowDefend()
+ main(args: String[]): void
# update(input: Input) void;
- drawStatusPanel()
- drawBuyPanel()

0..*

Tank

- tankProjGraphic: Image
- tankGraphic: Image
- RADIUS: int = 100
- DAMAGE: int = 1
- CD: int = 1000

+ Tank()

Spawn Event

- spawnNumber: int
- enemyType: String
- delayTime: int

+ SpawnEvent(WaveData: String)

Delay Event

- delayTime: int

+ DelayEvent(WaveData: String)

1

PlayerData

- playerdata: PlayerData
- money: int <<get>>
- life: int <<get>>

- PlayerData()
+getInstance(): PlayerData
+resetPlayerData(): void
+loseLife(): void
+addMoney(reward: int): void
+loseMoney(cost: int): void

could also potentially put the timescale here instead of passing it through methods

Singleton Should only have one instance of a player and should be accessible anywhere

0..*

Airplane

- SPEED: int = 5
- time: int
- count: int
- cooldown: int

+ launch(): void
+ updatePos(towards: Point): void

0..*

Explosive

- RADIUS: int = 150
- CD: int = 2000
- DAMAGE: int = 500
- time: int

+ detonate(): void
+ drop(): void

AbstractTank

- radius: int
- cooldown: int
- damage: int
- time: int
- pPos: Point

+ AbstractTank()
+ getPos(): Point
# setAttributes(radius: int, damage: int, cooldown: int)
+ checkBounds(map Map): void
+ render(): void
+renderIndicator(mousePos: Point): void
+ place(input: Input): void
+ launch(): void

0..*

SuperTank

- superTGraphic: Image
- superProjGraphic: Image
- CD: int = 500
- DAMAGE: int = 3
- RADIUS: int = 150

+ SuperTank()

AbstractEnemy

- vPos: Vector2 <<get>>
- speed: double
- currHealth: int <<get\set>>
- penalty: int <<set>>
- defaultSpeed: double <<set>>
- angle: double
- currentIndex: int

+ AbstractEnemy(point: Point, speed: double)
# setAttributes(defaultSpeed: double, reward: int, penalty: int, health: int) : void
+ updatePos(towards:Point, speed: double): void
+ render(graphic: Image): void
+ decreaseSpeed(): void
+ increaseSpeed(): void
+ recieveDamage(damage: int): void
+ slicerPenalty(penalty: int): void
+ slicerDeath(): void

1..*

ApexSlicer

- apexGraphic: Image
- HEALTH: int = 25
- REWARD: int = 150
- SPEED: double = 0.75
- PENALTY: int = 16

+ ApexSlicer(Point: point, speed: int)

MegaSlicer

- megaGraphic: Image
- HEALTH: int = 2
- REWARD: int = 10
- SPEED: double = 1.5
- PENALTY: int = 4

+ MegaSlicer(Point: point, speed: double)

Slicer

- regGraphic: Image
- HEALTH: int = 1
- REWARD: int = 2
- SPEED: double = 2
- PENALTY: int = 1

+ Slicer(Point: point, speed: double)

SuperSlicer

- superGraphic: Image
- HEALTH: int = 1
- REWARD: int = 15
- SPEED: double = 1.5
- PENALTY: int = 2

+ SuperSlicer(Point: point, speed: double)

the health calculations will be done in the super class, the health values in the child classes are used for instantiating

0..*

Projectile

- vPos: Vector 2
- SPEED: int = 10
- damage: int

+ updatePos(towards: Point): void
+ targetPos(enemy: AbstractEnemy): void
+ render(projectile: Image): void
+ damage(damage: int): void

If projectile speed is constant is it affected by the timescale?