The Tower Defense Game Documentation

Created By:

Tarm Kalavantavanich #6230252121
Thanat Phrudprisan #6230235521

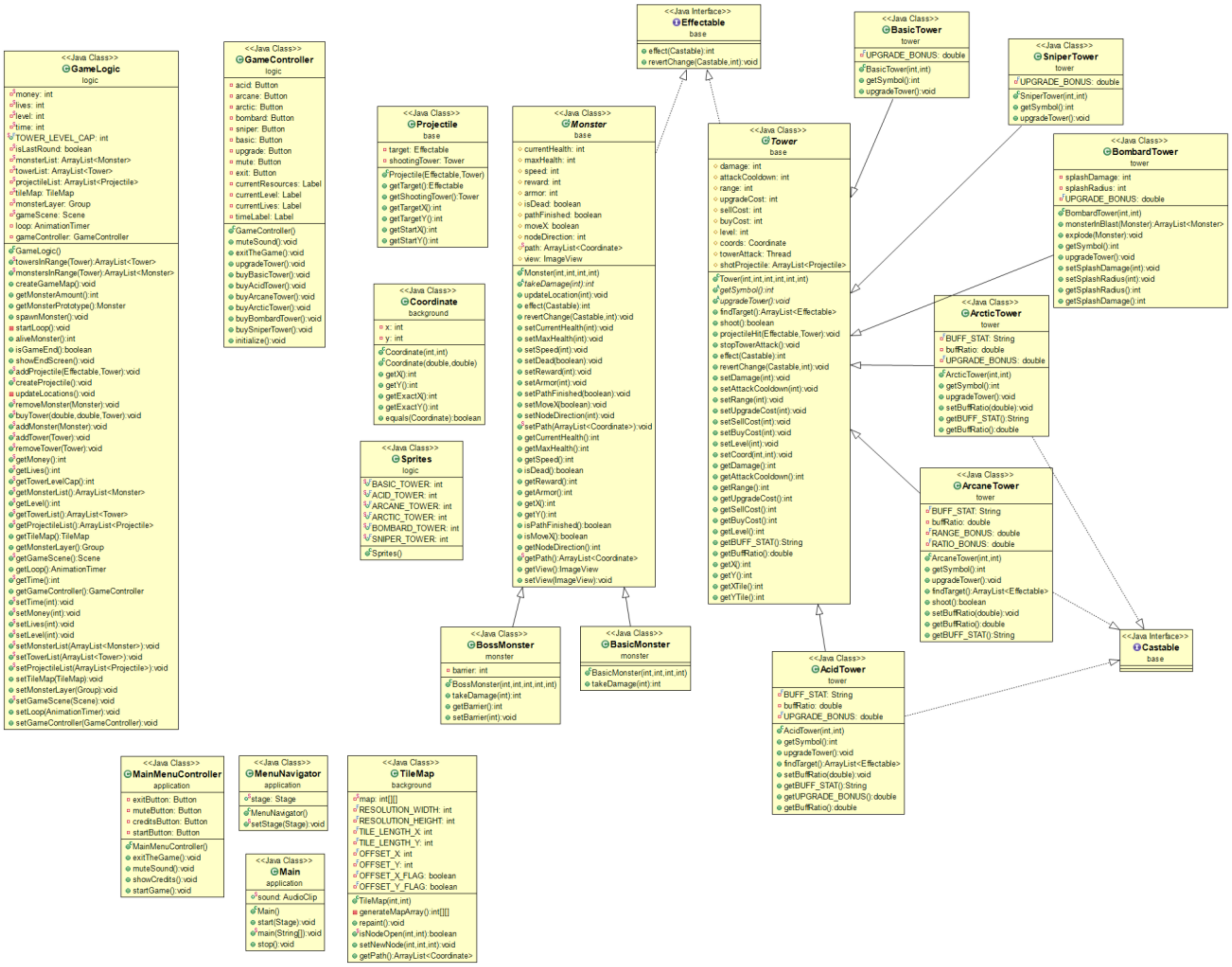
2110215 Programming Methodology
Semester 1 Year 2020
Chulalongkorn University

The Tower Defense Game

Introduction

The Tower Defense Game is your classic medieval-themed tower defense game, a single-player strategy game. The objective of the game is to survive the multiple waves of monster by building towers to kill them off.

UML diagram



Rule

The player starts the game of with 200 coin (i.e., “money”) and 20 lives. Money is dropped from monster and will be use on building and upgrading towers. Lives will drop by 1 if a monster survives to the end of the path (drop by 5 if it is a boss monster). Lives will not regenerate.

The player wins if he/she reach the last round (round 7) and still has lives that is more than zero. The player immediately loses if their lives drop to zero (or below).

Each tower upgrade differently and cost differently, however they will look the same and a tower can be upgraded at most 3 times.

Characters

Monster:

Basic Monsters:

Basic monsters are what will be spawned in usually. Its stats can vary, depending on which level (or wave) the player is currently in. They drop fewer coins than boss monsters but are weaker. The incoming damage will be decreased by its armor before decreasing its health (currentHealth -= (incomingDamage – armor)).

Boss Monsters:

Boss monsters are spawned in level 5. It has ‘barrier’ stat that is unique from basic monsters. Every tower hit takes 1 of the barrier. Once it drops to 0, the boss monsters will then calculate damage in the same way as basic monsters do (currentHealth -= (incomingDamage – armor)). Once they are slain, they drop large number of coins. They cannot be slowed by arctic towers.

Towers:

Basic Towers:

Basic tower has no special ability and is the cheapest. Once upgraded, they shoot farther and deal more damage.

Acid Towers:

Acid tower can decrease monsters’ armor that are shot by them. They shoot at all monsters in range at once. The armor decreasing effect lasts for 2 seconds. Once upgraded, they decrease more armor per shot.

Arcane Towers:

Unlike any other tower, the arcane tower shoots at other tower. Arcane Towers will shoot at all towers within range at once. Towers that are shot by them will be granted a buff that makes their shot deal more damage for 1 seconds. Once upgraded, the tower they buffed by them will deal even more damage.

Arctic Towers:

Arctic Tower decrease the movement speed of the monsters struck by them. Like basic towers, they shot at one monster at a time. The slowing debuff cuts the movement speed in half and lasts forever. However, boss monsters cannot be slowed. Once upgraded, it will shoot farther and faster.

Bombard Towers:

Bombard tower shots will explode on impact, dealing damage to all monsters within blast radius. They shot at one monster at a time and the initially hit monster will not take damage twice. Once upgraded, its damage will be increased.

Sniper Towers:

Sniper tower acts like an actual sniper. It has enormous range, shoots slowly and deal massive damage that can one-shot every, if not all, monster. Once upgraded, it will hit even harder and faster.

1 package base

This package contains base classes for Monster sub-classes and Tower sub-classes including 2 interfaces (Effectable and Castable) for other classes to use.

1.1 Abstract Class Monster

Implements : Effectable

This abstract class acts as a template for all monsters. It contains all necessary fields and methods that all monsters must have. All int fields must be non-negative integer at all times.

1.1.1 Fields

# int currentHealth	The current health of the monster.
# int maxHealth	The maximum health of the monster.
# int speed	Represents how fast the monster moves. The bigger the speed, the faster it moves.
# int reward	The money the player will gain when the player has slain the monster.
# int armor	The armor will decrease the incoming damage before calculating the remaining health.
# boolean isDead	Represents whether the monster is dead or not. (It is considered dead when currentHealth <= 0)
# boolean pathFinished	Check monster that finish the path.
# boolean moveX	Check monster move along in X axis.
# int nodeDirection	Times that the monster had changed axis
# ArrayList<Coordinate> path	Path that monster will move along.
# ImageView view	Outlook of monster

1.1.2 Methods

+ Monster(int maxHealth, int armor, int speed, int reward)	This is the Constructor. isDead is set to false and currentHealth is set to maxHealth by default.
+ abstract int	Returns the damage dealt. Boss monsters calculate

takeDamage(int incomingDamage)	differently from Basic monsters.
+ void updateLocation(int distance)	Moving monster along the path.
+ int effect(Castable e)	This method is called when this monster is buffed or debuffed by Castable caster. Returns the resulting stat.
+ int revertChange(Castable e)	This method is called after the buff/debuff ran out to make this monster’s stat return to normal. Returns the resulting stat.
+ getters and setters for all fields.	Getter/Setter for all fields. Note that all int fields must be a non-negative integer.
+ getX() and getY()	Return x and y coordinates, respectively. The returned x and y are the point which will be used to represent the monster.

1.2 Class `Tower`

Implements : Effectable

This abstract class acts as a template for all towers. It contains all necessary fields and methods that all towers must have.

1.2.1 Fields

# int damage	The damage this tower dealt to monsters.
# int attackCooldown	The amount of time (in seconds) the tower has to wait until it can shoot again.
# int range	How far a monster can be away from the tower that it can shoots.
# int upgradeCost	How much it costs to upgrade the tower.
# int sellCost	The money received when the player sold this tower.
# int buyCost	The money spent to buy this tower.
# int level	The level will be increase when the tower is upgraded, higher levels means higher stats.

# Coordinate coods	The place that this tower set on the map.
# Thread towerAttack	Thread that run the projectile of attacking from tower to monster.
# ArrayList<Projectile> shotProjectile	Animation of the shot of tower.

1.2.2 Methods

+ Tower(int damage, int attackCooldown, int range, int buyCost, int sellCost, int upgradeCost)	This is the Constructor. The level is set to 1 and coords is set to (0,0) , by default.
+ abstract int getSymbol()	Sprite that use to print entity on the map.
+ abstract void upgradeTower()	This method is called when the tower is being upgraded. Each tower's upgrade bonuses differ from each other.
+ ArrayList<Effectable> findTarget()	The default findTarget() method. Add only the front-most monster to the ArrayList.
+ void projectileHit (Effectable target , Tower shootingTower)	Effect of all tower attack to monster and tower.
+ void stopTowerAttack()	Exception to stop the tower attack.
+ void shoot()	The default shoot() method. Shoot the target from findTarget() and apply effects (if any).
+ public int effect(Castable caster)	This method is called when this tower is buffed by Castable caster. Returns the resulting stat
+ public int revertChange(Castable caster)	This method is called after the buff ran out to make this tower's stat return to normal. Returns the resulting stat.
+ getter and setter for all fields	Do note that all int fields must not be negative.
+getX() and getY()	Returns the coordinate of the center of the tower.

1.3 Interface `Effectable`

This interface represents a character (from this point onwards, ‘character’ is referred to both monsters and towers) whose stat (i.e. field) can be altered by other “castable” character by means of buffing and/or debuffing.

1.3.1 Methods

+ abstract int effect(Castable caster)	To be overridden by implementing classes. Called to change the stat.
+ abstract int revertChange(Castable caster	To be overridden by implementing classes. Caled to change the stat back to normal.

1.4 Interface `Castable`

This is a marker interface for grouping those character who can cast buffing and/or debuffing effects onto other characters. Because it is a marker interface, it has no methods.

1.5 Class `Projectile`

This class represents projectiles that towers shoot.

1.5.1 Fields

- Effectable target	The character which the projectile is aimed at.
- Tower shootingTower	Tower that shoot this projrctile.

1.5.2 Methods

+ Projectile(Effectable target, int towerX, int towerY)	This is the constructor. Set each field accordingly.
+ int getTargetX()	Return x coordinate of target.
+ int getTargetY()	Return y coordinate of target.
Getter and Setter all field	

2 package `monster`

This package contains the concrete monster classes. Basic monster and Boss monster calculate damage differently.

2.1 Class BasicMonster

Extends : Monster

This class is the monsters that will walk on the map. Each decreases the player’s live by 1 if it survives to the end of the path.

2.1.1 Methods

+ BasicMonster(int health, int armor, int speed, int reward)	This is the Constructor for making a template (i.e prototype) monster. Calls the super constructor to set each field.
+ int takeDamage(int incomingDamage)	The incomingDamage is reduced by armor and is dealt to the monster currentHealth. If the monster is slain, set its isDead to true, add the reward money to the player’s and remove this monster. Returns the damage taken.

2.2 Class BossMonster

This class represents the BossMonster who will be significantly harder to defeat and drop bigger rewards.

2.2.1 Field

- int barrier	This represents the amount of hit the boss must take before the boss takes actual damage.

2.2.2 Methods

+ BossMonster(int health, int armor, int speed, int reward, int barrier)	This is the Constructor for making a template (i.e prototype) boss monster. Calls the super constructor to set each field.
+ int takeDamage(int incomingDamage)	Each hit decreases the barrier by 1. Once the barrier is broken, the boss calculate takeDamage the same way basic monster does.

3 package tower

This package contains the concrete tower classes. Each tower class has different ability and stats.

3.1 Class AcidTower

Implements : Castable

Extends : Tower

This class represent acid tower. It attacks all target in range, deals little damage but decreases the struck monster’s armor.

3.1.1 Fields

- final String BUFF_STAT	The specific stat for this tower (“armor”).
- private double buffRatio	The ratio of the remaining armor of the struck monsters. Will be higher with higher level.
- private final double UPGRADE_BONUS	The ratio of resulting stat after this tower is upgraded.

3.1.2 Methods

+ public AcidTower(int x, int y)	Set: damage = 10, attackCooldown = 3, range = 8, buyCost = 120, sellCost = 100, upgradeCost = 300, BUFF_STAT = “armor”, buffRatio = 0.8, UPGRADE_BONUS = 2. Set the coordinate of the tower to (x,y).
+ int getSymbol()	Sprite that use to print entity on the map.
+ void upgradeTower()	The resulting buffRatio = buffRatio * UPGRADE_BONUS, rounded down to 2 dp.
+ ArrayList<Effectable> findTarget()	Overrides the default findTarget(). Returns all monsters that is in tower’s range.
+ void shoot()	Overrides the default shoot(). The acid tower shoots every monsters that is in tower’s range.
+ getter and setter for all fields	Excluding setters for final fields.

3.2 Class `ArcaneTower`

Implements: Castable

Extends: Tower

This class represent arcane Tower. It doesn’t attack any monsters but it buffs allied towers’ damage. It shoots every tower in range at once.

3.2.1 Fields

- final String BUFF_STAT	The specific stat for this tower (“damage”).
- private double buffRatio	The ratio of the resulting damage of the tower. Will be higher with higher level.
- private final double RANGE_BONUS	Represents how much its range will be increased if the tower is upgraded.
- private final double RATIO_BONUS	Represents how much its buffRatio will be increased if the tower is upgraded.

3.2.2 Methods

+ public ArcaneTower(int x, int y)	Set: damage = 0, attackCooldown = 3, range = 10, buyCost = 150, sellCost = 50, upgradeCost = 340, BUFF_STAT = “damage”, buffRatio = 1.2, RANGE_BONUS = 2, RATIO_BONUS = 0.15. Set the coordinate of the tower to (x,y).
+ int getSymbol()	Sprite that use to print entity on the map.
+ void upgradeTower()	The resulting buffRatio = buffRatio + RATIO_BONUS, The resulting range = range * RANGE_BONUS.
+ ArrayList<Effectable> findTarget()	Overrides the default findTarget(). Returns all towers that is in this tower’s range.
+ void shoot()	Overrides the default shoot(). The arcane tower buffs every towers that is in its range.
+ getter and setter for all fields	Excluding setters for final fields.

3.3 Class ArcticTower

Implements: Castable

Extends : Tower

This class represent arctic tower. The monsters shot by the arctic tower is slowed.

3.3.1 Fields

- final String BUFF_STAT	The specific stat for this tower (“speed”).
- private double buffRatio	The ratio of the resulting speed of the monster. Will be higher with higher level.
- private final double RANGE_BONUS	Represents how much its range will be increased if the tower is upgraded.
- private final double RATIO_BONUS	Represents how much its buffRatio will be increased if the tower is upgraded.

3.3.2 Methods

+ public ArcticTower(int x, int y)	Set: damage = 30, attackCooldown = 7, range = 5, buyCost = 70, sellCost = 40, upgradeCost = 150, BUFF_STAT = “speed”, buffRatio = 0.5, RANGE_BONUS = 1.5, RATIO_BONUS = 0.1. Set the coordinate of the tower to (x,y).
+ int getSymbol()	Sprite that use to print entity on the map.
+ void upgradeTower()	The resulting buffRatio = buffRatio + RATIO_BONUS, The resulting range = range * RANGE_BONUS.
+ getter and setter for all fields	Excluding setters for final fields.

3.4 Class `BasicTower`

Extends : Tower

This class represent basic tower. It is the cheapest and doesn’t have anything special.

3.4.1 Fields

- private final double UPGRADE_BONUS	Represents how much its range and damage will be increased if the tower is upgraded.

3.4.2 Methods

+ public BasicTower(int x, int y)	Set: damage = 20, attackCooldown = 2, range = 5, buyCost = 50, sellCost = 30, upgradeCost = 100, UPGRADE_BONUS = 1.5. Set the coordinate of the tower to (x,y).
+ int getSymbol()	Sprite that use to print entity on the map.
+ void upgradeTower()	The resulting damage = damage + RATIO_BONUS, The resulting range = range * RANGE_BONUS.

3.5 Class `BombardTower`

Extends : Tower

This class represent bombard tower. Its shot will explode on impact with monster, dealing damage in a circle around the struck monster. The initially struck monster will take damage twice.

3.5.1 Fields

- int splashDamage	The damage it deals to monster in the explosion. It is set to be damage / 3 (rounded down to int).
- int splashRadius	The radius from the initially struck monster.
- private final double UPGRADE_BONUS	Represents how much its damage will be increased if the tower is upgraded.

3.5.2 Methods

+ public BombardTower(int x, int y)	Set: damage = 40, attackCooldown = 3, range = 10, buyCost = 120, sellCost = 50, upgradeCost = 250, splashDamage = damage / 3, splashRadius = 1, UPGRADE_BONUS = 1.5. Set the coordinate of the tower to (x,y).
+ ArrayList<Monster> monsterInBlast (Monster aimmedMonster)	Find all of the monsters that is in blast range. Add to ArrayList and return it. Note: aimmedMonster will be added too if it survives.
+ void upgradeTower()	The resulting damage = damage *UPGRADE_BONUS, The splashDamage changes accordingly.

+ int getSymbol()	Sprite that use to print entity on the map.
+ getter and setter for all fields	Excluding setters for final fields. splashDamage must not be lesser than 1.

3.6 Class `SniperTower`

Extends : Tower

This class represent sniper tower. It strikes hard but slow – like an actual sniper.

3.6.1 Fields

- private final double UPGRADE_BONUS	Represents how much its damage will be increased and its attackCooldown will be decreased if the tower is upgraded.

3.6.2 Methods

+ public BasicTower(int x, int y)	Set: damage = 100, attackCooldown = 8, range = 15, buyCost = 200, sellCost = 100, upgradeCost = 300, UPGRADE_BONUS = 2.0. Set the coordinate of the tower to (x,y).
+ int getSymbol()	Sprite that use to print entity on the map.
+ void upgradeTower()	The resulting damage = damage *UPGRADE_BONUS, The resulting attackCooldown = attackCooldown - 1.

4 package `logic`

This package contains mostly static methods and fields. It handles game logic that is core to the game.

4.1 Class `GameLogic`

This class is the class that will handle all of logics and calculation.

4.1.1 Fields

- private int money	Represents money the player has. It is used to buy and upgrade towers.
- private int lives	Represents the player lives. Game is lost when lives drops to 0 (or below). BasicMonster decreases 1 and BossMonster decreases 5.
- private int level	Represents number of waves.
- private int time	Represent time of the loop that monster going to spawn.
- private boolean isGameOver	Represents whether the game has ended or not (either win or lose). It is set to false by default.
- private final int TOWER_LEVEL_CAP	Represents the maximum level the tower can reach. It is set to 3.
- private ArrayList<Monster> monsterList	Contains alive monsters in the round.
- private ArrayList<Tower> towerList	Contains towers built in the game.
- private ArrayList<Projectile> projectileList	Contains projectiles that must be drawn.
-TileMap tileMap	Map of the game.
-Group monsterLayer	Layer of the monster that show in the map.
-Scren gameScene	Scene that contain map and monsterLayer
-AnimationTimer loop	Loop of the game.
-GameContrlloer gameController	Class that control the game logic.

4.1.2 Methods

+ArrayList<Tower>towersInRange(Tower attackingTower)	List of the tower that in range of the buffer tower to applied effect.
+ArrayList<Monster> monstersInRange(Tower attackingTower)	List of the tower that in range of the attacker tower to attack and buff.
+void createGameMap()	Create the game map with 1280*800 and add tower shop and data.

+int getMonsterAmount()	Number of monster that going to spawn in that waves
+Moster getMonsterPrototype()	Stat and type of monster in the waves.
+ void spawnMonster()	Create the monster to the map and run along the path.
-void startLoop()	Looping the wave and spawn monster and move monster.
+ int aliveMonster()	Number of monster that alive.
+ boolean isGameEnd()	Check lives <= 0 and all the monster has been defeated.
+ void showEndScreen()	Show end screen.
+ void addProjectile(Effectable target, Tower shootingTower)	Set projectile when the tower attack or buff.
+ void createProjectile()	Create the projectile to attack the monster and buff tower.
- void updateLocation()	Move monster along the path.
+synchronized void removeMonster(Monster monster)	Remove monster from the map which reach the final path or dead.
+ void buytower(double x, double y, Tower t)	Add tower to the list and settle tower on the map
+ void dropCoin(Monster monster)	Add reward when defeated the monster.
Getter and Setter all field	

4.2 Class GameController

4.2.1 Fields

- Button acid	Button that use to buy acid tower
- Button arcane	Button that use to buy arcane tower
- Button bombard	Button that use to buy bombard tower
- Button sniper	Button that use to buy sniper tower

- Button basic	Button that use to buy basic tower
- Button upgrade	Button that use to upgrade tower
- Button arctic	Button that use to arctic tower
- Button mute	Mute sound
- Button exit	Exit game
- Label currentResources	Show current money
- Label currentLevel	Show current wave
- Label currentLives	Show current lives
- Label timeLabel	Show current time

4.2.2 Methods

+ void muteSound()	Stop the sound if it play and play if it stopped.
+ void exitTheGame()	Exit Game.
+ void upgradeTower()	Upgrade the tower that click on.
+ void buyTower() (every tower)	Set the tower on the tile map.
+void initialize()	Set all data to the Label.

4.3 Class Sprites

4.3.1 Fields

+ final int BASIC_TOWER	The number that identified the tower use to put tower on the map
+ final int ACID_TOWER	The number that identified the tower use to put tower on the map
+ final int ARCANE_TOWER	The number that identified the tower use to put tower on the map
+ final int BOMBARD_TOWER	The number that identified the tower use to put tower on the map

+ final int ARCTIC_TOWER	The number that identified the tower use to put tower on the map
+ final int SNIPER_TOWER	The number that identified the tower use to put tower on the map

5 package Appication

This Package contains all classes that concerned with main menu and screen switching.

5.1 Class Main

Extends : Application

5.1.1 Fields

- AudioClip sound	Represents song of this game.

5.1.2 Methods

+ void start(Stage stage)	Set up main menu on screen and auto play sound. The screen is 1280 * 800 and has 4 buttons Start, Credits, Mute and Exit.
+ void main(String[] args)	Run start.
+ void stop()	Stop all things.

5.2 Class MainMenuController

5.2.1 Fields

- Button exitButton	Represents exit button.
- Button muteButton	Represents mute button.
- Button creditsButton	Represents credits button.
- Button startButton	Represents start button.

5.2.2 Methods

+ void exitTheGame()	Close the main menu window.
+ void muteSound()	Stop the song if it is playing and play it if it stopped.
+ void showCredits()	Pop-up new window that contain name of the creator of this game.
+ void startGame()	Set up GameLogic and call createGameMap()

5.3 Class `MenuNavigator`

5.3.1 Fields

<u>+ Stage stage</u>	Represents stage that is showing

5.3.2 Methods

+ void setStage(Stage stage2)	Make stage2 show instead of the current one.

6 package `background`

This Package contains all classes that concerned with game map.

6.1 Class Coordinate

6.1.1 Fields

- int x	Represents coordinate x
- int y	Represents coordinate y

6.1.2 Constructor

+ Coordinate(int x , int y)	Set up coordinate x and y which x and y are integer.
+ Coordinate(double x , double y)	Set up coordinate x and y.

6.1.3 Methods

Getter and setter of all field	
+ boolean equal(Coordinate lhs)	Compare 2 coordinate that has the same x and y.

6.2 Class TileMap

Extends : ImageView

6.2.1 Fields

- int[][] map	Represents map of the game.
- final int RESOLUTION_WIDTH	Represents resolution of width.
- final int RESOLUTION_HEIGHT	Represents resolution of height.
- final int TILE_LENGTH_X	Represents width of tile map.
- final int TILE_LENGTH_Y	Represents height of tile map.
- final int OFFSET_X	Represents off set of x in map.
- final int OFFSET_Y	Represents off set of y in map.
- final boolean OFFSET_X_FLAG	Represents is having off set x.
- final boolean OFFSET_Y_FLAG	Represents is having off set y.

6.2.2 Methods

- int[][] generateMapArray()	Set map.
+ void repaint()	Paint all entity in map.
+ boolean isNodeOpen(int x , int y)	Check is that node don’t have any entity.
+ void setNewNode(int x, int y ,int value)	Set the entity on the map.
+ ArrayList<Coordinate> getPath()	Find path of monster.