The Tower Defense Game

Documentation

Created By:

Tarm Kalavantavanich #6230252121

Thanat Phrudprisan #6230235521

**2110215 Programming Methodology**

**Semester 1 Year 2020**

**Chulalongkorn University**

## **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.

# **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) |
| # Coordinate coords | TO BE FILLED |

### 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  takeDamage(int incomingDamage) | Returns the damage dealt. Boss monsters calculate differently from Basic monsters. |
| + 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. |

# **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. |

### 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 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 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. |
| - final int startX | The X coordinate of where the projectile is shot from. |
| - final int startY | The Y coordinate of where the projectile is shot from. |

### 1.5.2 Methods

|  |  |
| --- | --- |
| + Projectile(Effectable target, int towerX, int towerY) | This is the constructor.  Set each field accordingly. |

# 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. |
| + BasicMonster(int x, int y, BasicMonster prototype) | This is the Constructor for making the actual monster at coordinate (x,y), using the fields from prototype. |
| + 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.

### 2.1.1 Field

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

### 2.1.2 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. |
| + BossMonster(int x, int y, BossMonster prototype) | This is the Constructor for making the actual boss monster at coordinate (x,y), using the fields from prototype. |
| + 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). |
| + 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). |
| + 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). |
| + 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.

### 3.3.1 Fields

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

### 3.3.2 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). |
| + 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.

### 33.3.1 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. |

### 3.3.2 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. |
| + getter and setter for all fields | Excluding setters for final fields. |

### **3.6 Class ColorBox**

Implements: Interactable

This class represent Colored Box type Entity. It is a solid object that can be pushed around like Box, but only when the game’s global switch matched their internal value.

### 3.6.1 Fields

|  |  |
| --- | --- |
| - boolean activeBool | A boolean contains the game switch state that the box can be pushed. |

### 3.6.2 Constructor

|  |  |
| --- | --- |
| + ColorBox (boolean activeBox) | Initialize activeBool |

### 3.6.3 Methods

|  |  |
| --- | --- |
| + int getSymbol() | return the value Sprites.BOX\_RED if the box active on the switch value true or Sprites.SWITCH\_BLUE otherwise.  These contains the correct value of the subimage used to render the colored box. |
| + boolean interact(Entity e) | If the game’s global switch matches the activeBool, move the box in the same direction as Entity e and returns the result of the move.  Otherwise, returns false. |
| + boolean getActiveBool()  + void setActiveBool(boolean activeBool) | Getter/Setter for activeBool |

**3.7 Class TrashCompactor**

Implements: Interactable, Consumable, Updatable

This class represent Trash Compactor type Entity. It is a solid object. However, it can consume any type of Box (Box, ColoredBox) if pushed into it, and then entering cooldown mode. During the cooldown mode, it cannot consume any Box. The cooldown timer ticks down once the player made a movement.

### 3.7.1 Fields

|  |  |
| --- | --- |
| - int cooldown | A cooldown time, the Trash Compactor is only functional if the cooldown time is 0. |

### 3.7.2 Constructor

|  |  |
| --- | --- |
| + TrashCompactor () | Initialize cooldown with 0 |

### 3.7.3 Methods

|  |  |
| --- | --- |
| + int getSymbol() | return the value Sprites.COMPACTOR\_ON if the Trash Compactor is usable. Returns Sprites.COMPACTOR\_OFF otherwise  These contains the correct value of the subimage used to render the Trash Compactor. |
| + boolean interact(Entity e) | Returns true if the Entity e is a type of box and the Compactor is active to allow them to pass. Returns false otherwise to block them.  **Tip:** You can use Entity.isBox(Entity e) to check if Entity e is a type of box or not. |
| + boolean consume(Entity e) | Returns true if the Entity e is a type of box and the Compactor is usable to consume the Entity, as well as setting the cooldown time to the value of GameController.MAX\_COOLDOWN\_TIME.  Returns false otherwise.  **Tip:** You can use Entity.isBox(Entity e) to check if Entity e is a type of box or not. |
| + void update() throws IllegalValueException | Decrease its cooldown value by 1 if it’s more than 0  If cooldown is less than 0, throws IllegalValueException |
| + void valueCorrection() | Set cooldown to 0 |
| + int getCooldown()  + void setCooldown(int cooldown) | Getter/Setter for cooldown |