

# **CP Story Documentation**

## **Created by**

Surawich Laprattanatrai 5930551621

Natchaporn Ponkitsaran 5931018721

**2110215 Programming Methodology**  
**Semester 1 Year 2017**  
**Chulalongkorn University**

# CP Story Documentation

## Introduction

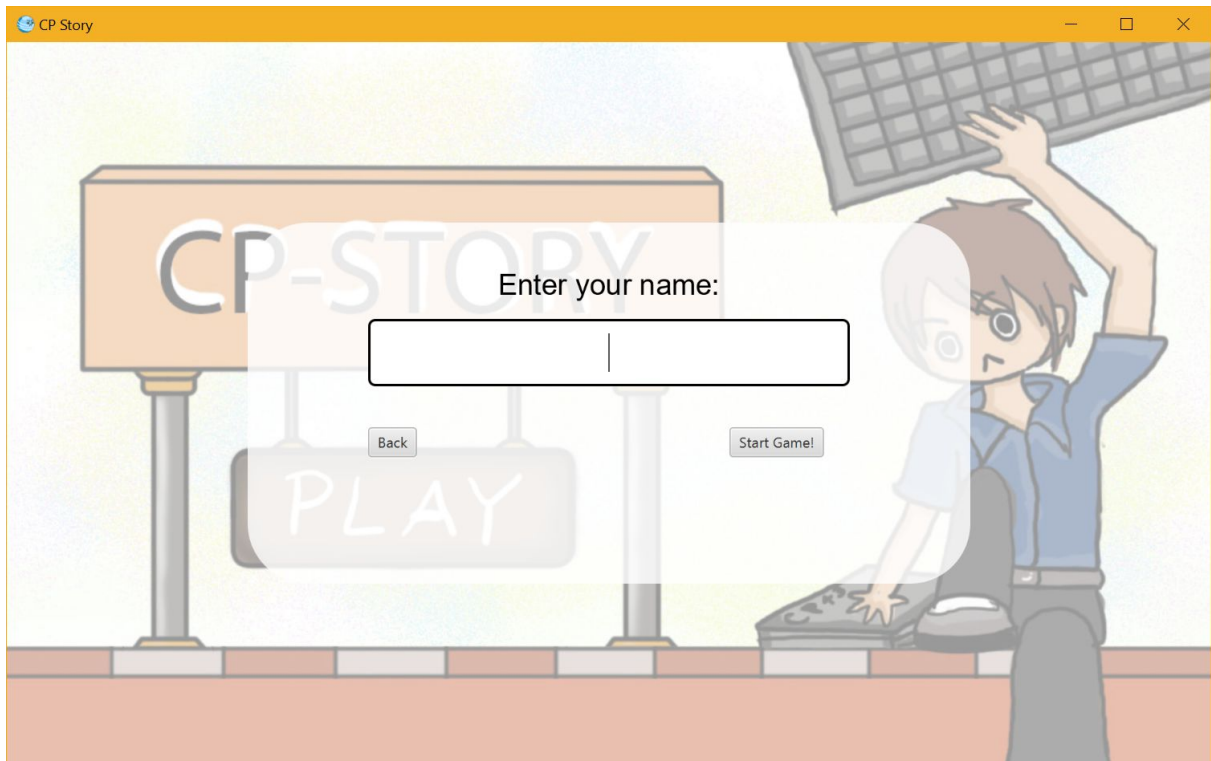
CP Story is an RPG game. The goal of this game is to reach level 20 as fast as possible. To gain a level, a player has to fight monsters to collect experience. Enough experience on each level will make a player go to the next level. Fighting monsters may seem very easy by just pushing an attack button on the keyboard. That is boring! While fighting monsters is the backbone of the game, the game provides other interesting features to make playing more fun such as the funny look of monsters, helpful skills (sometimes funny) and items to help you win the game easier.

## User Manual

### Start Scene



To start the game, click on the play button.



You will be prompted to enter your name. This is just for showing under the character. It's okay to leave it blank. If you leave it blank, the default name, Joetoken, will be used. After you click start game button, you will be presented in the game scene.

## Game Scene

The game scene consists of 5 major components.



## 1. Main view

Main view is the view that shows everything in the current map. Background image, monsters, players and particles are shown in this area. Main view's area occupies the whole screen (including status bar area)

## 2. Status bar

Status bar is the place where the player's properties are shown. The following list are the properties that status bar shows.

- Current level
- Current HP and max HP
- Current MP and max MP
- Current experience points and remaining experience points to reach next level
- Skills, its cooldown and its description
- Inventory

## 3. Message

There are some cases that the player does something that is not allowed to do, for example, using skill without enough MP. The message will notify the player that way.

## 4. Timer

The score of this game depends on the time that you play. So, there is a time indicator that shows total time that you have played the current session. Timer is shown on the top-center of the screen.

# Game Controls

## 1. Movement

Move left: Arrow left

Move right: Arrow right

Jump: Space

Jump down through the platform: Arrow down + Space

Use portal: Arrow up

## 2. Other controls

Normal attack: A




Collect item: C

Use skill: Q, W, E, R

Use item: Number 1 to 0

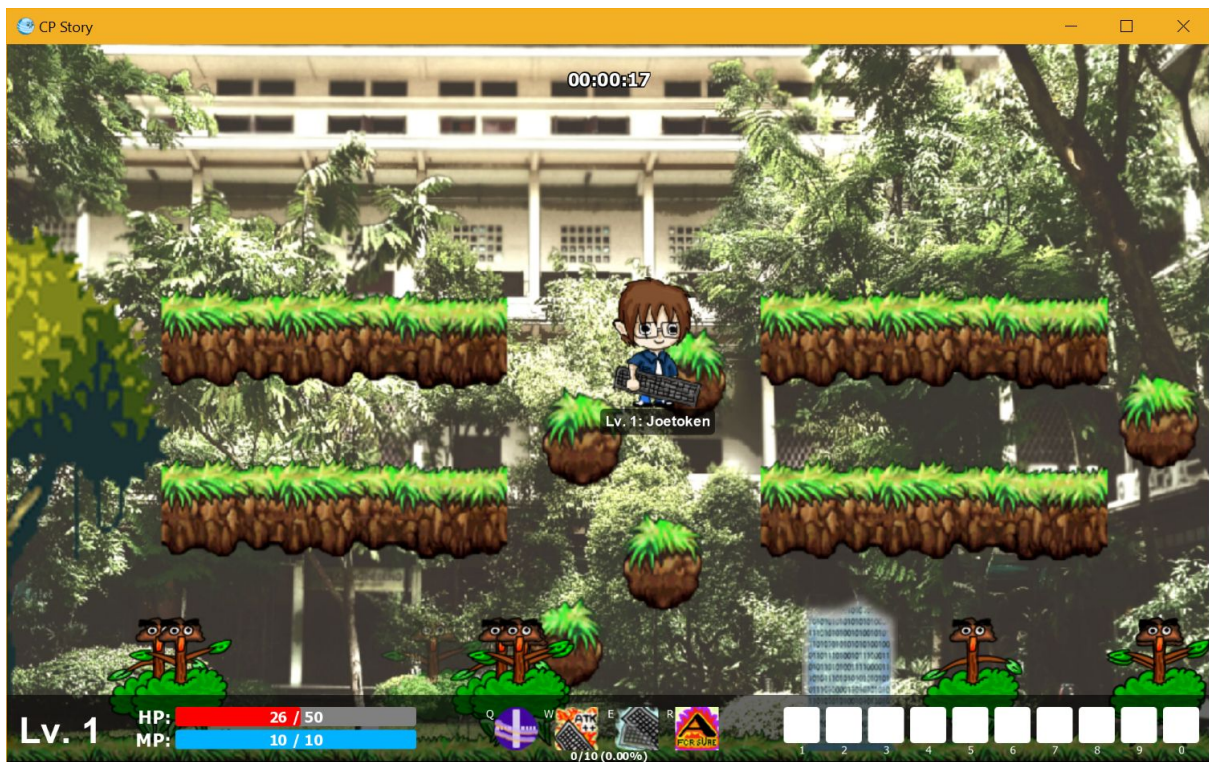


## Monsters

	Name	Level	HP	Attack Damage	Experience	Drops
	Mini Tree	2	20	3 ~ 8	4	Red Potion, Blue Potion
	Prog Meth	6	65	6 ~ 12	10	Red Potion, Blue Potion
	Node	10	120	10 ~ 20	17	Red Potion, Fast Learn Scroll

## Maps

### 1. Garden



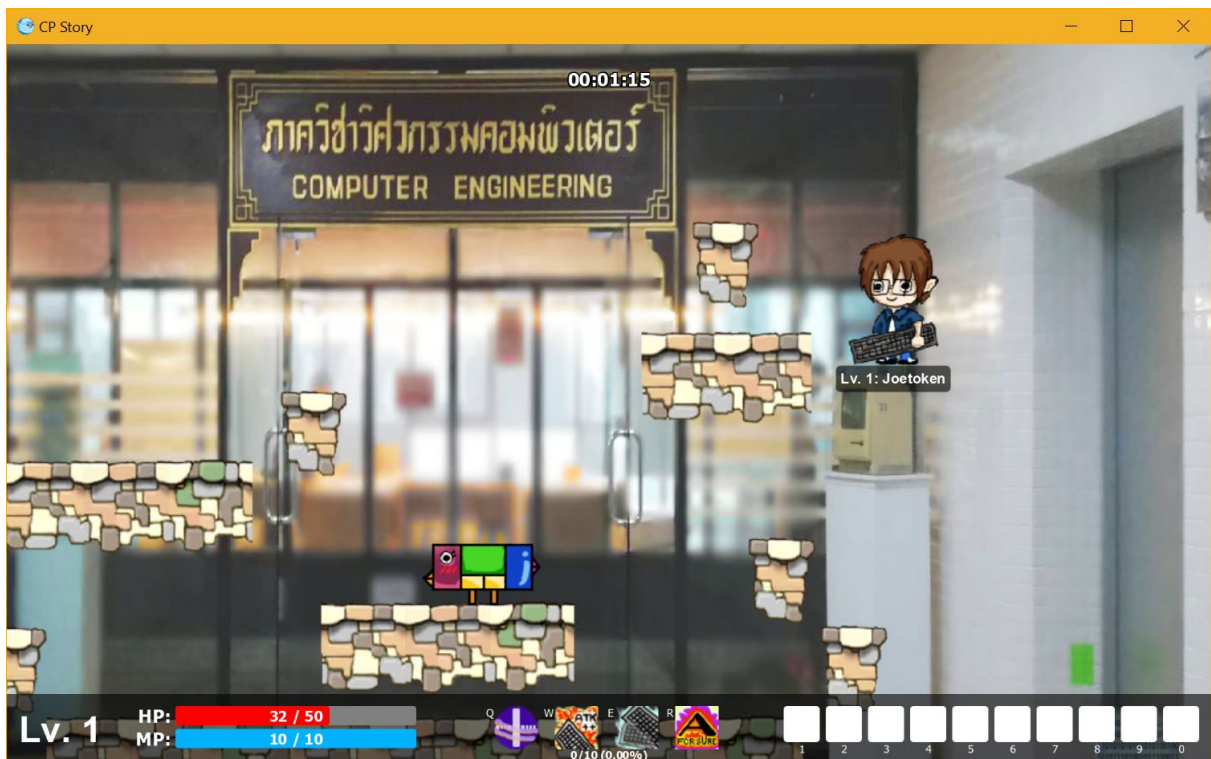
**Monster type:** Mini Tree  
**Connection:** Building 4

## 2. Building 4



**Monster type:** Prog Meth  
**Connection:** Garden, Sky Cafe

## 3. Sky Cafe



**Monster type:** Node  
**Connection:** Building 4

## Skills

	Name	MP used	Cooldown time	Behavior
	Multi Kill	4	0.5 second	Like normal attack, but attack damage is increased by 50% and can hit up to 4 monsters
	Power Up	12	-	Increase attack damage by 50% for 90 seconds
	Power of Joe	2 per hit	-	Hit monster continuously with attack damage decreased by 50%
	Grade A for sure	40	2 minutes	Kill all monsters in the map

## Items

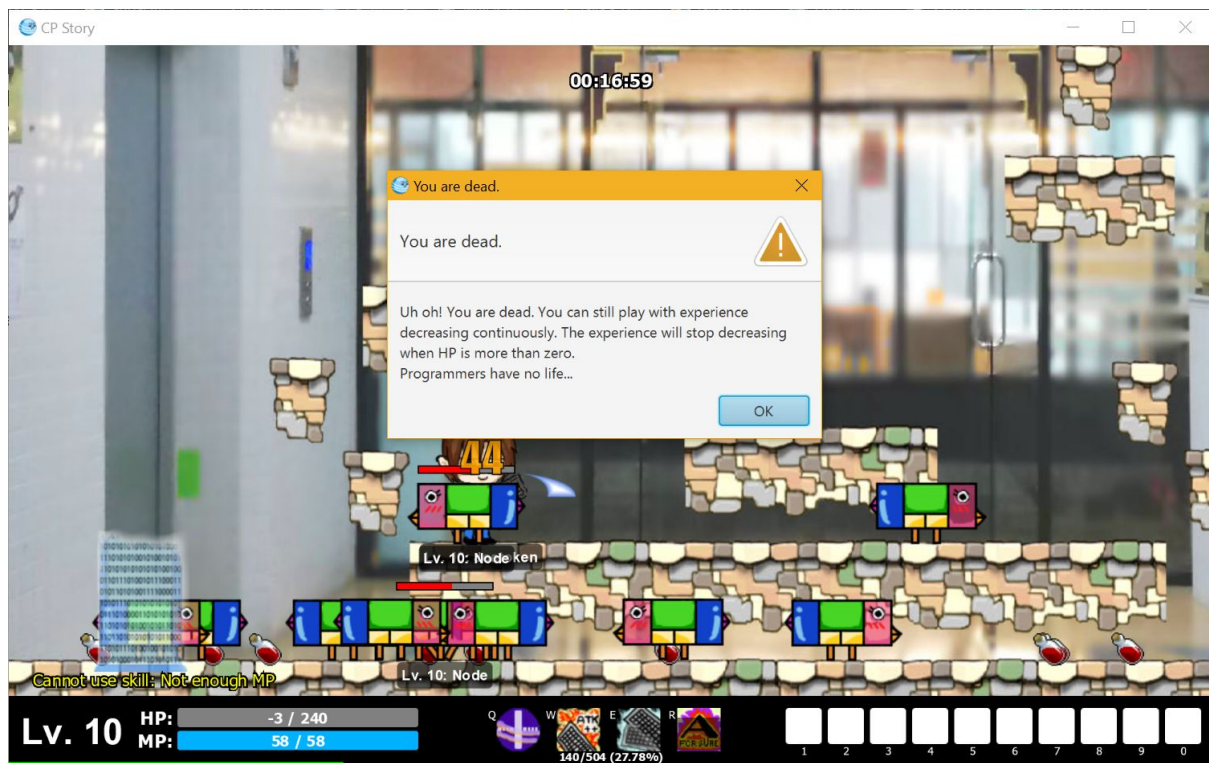
	Name	Maximum items per stack	Behavior
	Red Potion	50	Increase HP for 50 points
	Blue Potion	50	Increase MP for 20 points
	Fast Learn Scroll	1	Increase the amount of experience gained by 50% for 60 seconds

## Gameplay

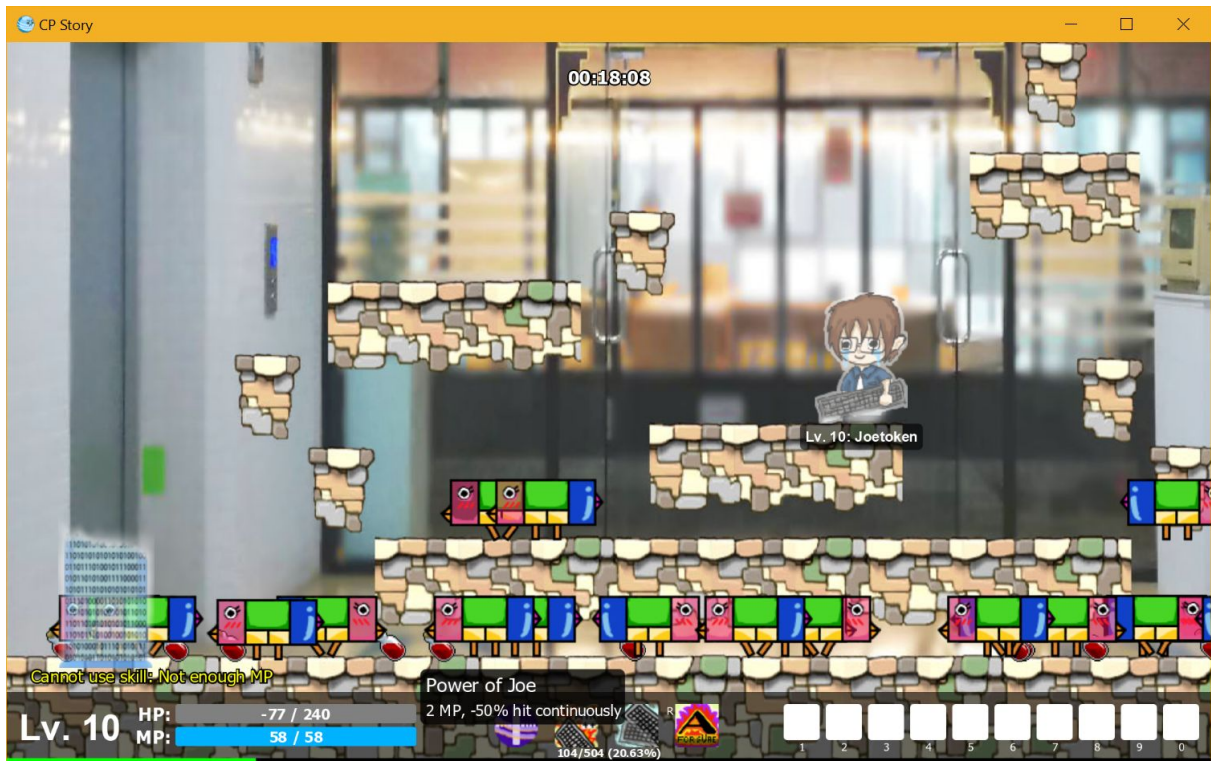
As soon as you have clicked “Start Game” button on the start scene, you will be presented in the map called Garden. You can control a player by using the guide above. As time passing, you will see monsters spawning. The monsters you see in this map is called Mini Tree. It is the easiest monster in the game with the lowest health and attack damage. Be cautious! When it touches you, you hurt. A simple way to attack monster is to press button A on the keyboard. Once the monster is attacked, it will be aggressive, follow you and attack you for some amount of time. If you attack it repeatedly until it dies, you will get experience which can be observed in the status bar. Once the experience bar is full, your level will be increased, increasing your maximum health, maximum magical points and attack damage. As the monster dies, it sometimes drops item. Each item effects a player differently. The behavior of each item can be found in the guide above. To collect item, press



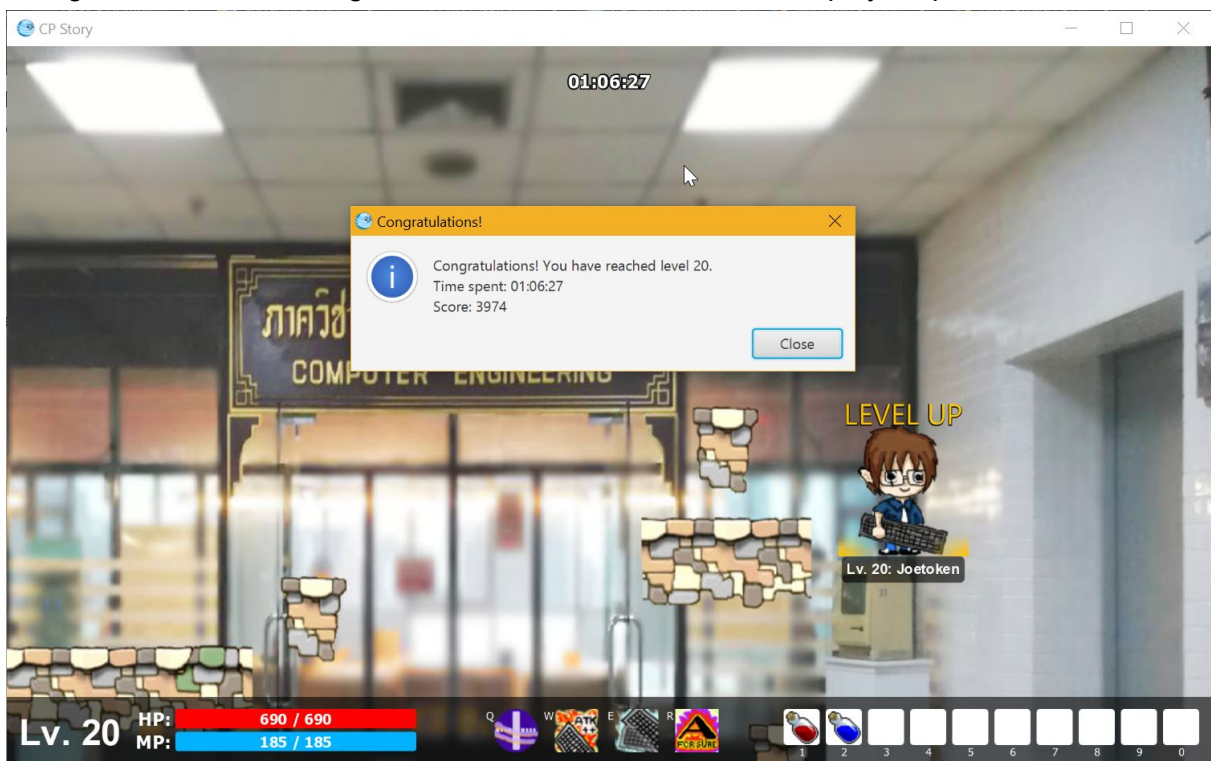
button C on the keyboard. The potion will be shown in the right of the status bar. To use item, press the number key on your keyboard following to the number shown below the item icon. As you level, you may want to fight more difficult monster to get more experience. You can find more difficult monster by using the portal in the bottom right of the map. To teleport, stand on the center of the portal and press arrow key UP. High level monsters have more health and attack damage than low level monsters, so you may want to use some type of skills to make you stronger. You can find skill in the center of status bar. For the detail of each skill is in the guide above. When you reach level 10, the next monster is waiting for you in the next map. Warp to the next map to fight the most difficult monster in the game. This monster is special for one reason. Since when your level is high, you need more experience to level up. You can get Learn Fast Scroll as a rare drop from this monster. This item let you earn 50% more experience than usual for 60 seconds.







Normally when the player hp reach 0, your character will die but that does not happen in this game. Game will be paused, alert the pop-up and your character will turn to be a ghost to continue the game. It will turn to normal when the player hp is more than 0.



And finally, when you reach level 20, you will win the game and the score will be shown.

## Implementation Detail



# 1. Package main

## 1.1 Class Main

### 1.1.1 Field

- Stage stage	Store stage that we get when the program starts.
- Scene startScene	Store the start scene
- Scene gameScene	Store the game scene

### 1.1.2 Method

+ void main(String[] args)	Method that always runs when the program is started.
+ void start(Stage inputStage)	The main entry point for all JavaFX applications. The start method is called after the init method has returned, and after the system is ready for the application to begin running. [Ref. <a href="https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html">https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html</a> ]
+ void stop()	This method is called when the application should stop.
<u>Getters of stage, startScene and gameScene and setter of startScene</u>	

# 2. Package input

## 2.1 Class KeyInput

This class handles keyboard input during the game.

### 2.1.1 Field

- <u>Set&lt;KeyCode&gt; activeKeys</u>	A set containing all the keys that the user is pressing
- <u>Queue&lt;KeyCode&gt; triggerKeys</u>	A queue of keys which will be polled and processed by Player class
- <u>Set&lt;KeyCode&gt; UNPOLLABLE_KEYS</u>	A set of keys that don't need to be in triggerKeys variable



### 2.1.2 Static Block

Add some keyboard keys (Left, Right, Down, Space and E) to UNPOLLABLE\_KEYS

### 2.1.3 Method

- void addKey(KeyCode key)	Add key to activeKeys and sometimes triggerKeys
- void removeKey(KeyCode key)	Remove key from activeKeys
+ boolean pressingKey(KeyCode key)	Check whether the user is pressing the specific key
+ void clear()	Remove all keys in the activeKeys set
+ boolean isPollAvailable()	Check whether there is a content inside triggerKeys queue
+ KeyCode pollKey()	Poll key from triggerKeys
+ void bindScene(Scene scene)	Bind KeyInput methods to the scene

## 3. Package utility

### 3.1 Class AudioPlayer

This class provides the ability to fade in and out the audio.

#### 3.1.1 Method

+ void fadeIn(MediaPlayer mediaPlayer, long fadeTimeMillis)	Fade in the audio for the given fade time
+ void fadeOut(MediaPlayer mediaPlayer, long fadeTimeMillis)	Fade out the audio for the given fade time

### 3.2 Class Random

This class provides the randoming methods used in this game.

#### 3.2.1 Method

+ int randInt(int n)	Random an integer from 0 to n-1
+ Pair<T,Double> weightedRandomInList(List<Pair<T,Double>> list)	Weighted random an object in the list with output being an object and a random number between 0 and the given double value
+ List<T>	Random in the list with multiple objects

<u>multipleRandomInList(List&lt;Pair&lt;T,Double&gt; &gt; list)</u>	allowed
---	---------

## 4. Package particle

### 4.1 Interface IParticle

This interface is a template of the graphics that only be shown and cannot be interacted.

#### 4.1.1 Method

+ boolean isVisible()	Abstract method by default, returns true if the particle is visible.
+ boolean isExpired()	Abstract method by default, returns true if the particle should be removed from the map
+ void render(GraphicsContext gc)	Abstract method by default, render the particle

### 4.2 Class Damage

This class represents orange and purple number that is shown when attacking occurs.

#### 4.2.1 Field

- <u>Font TEXT_FONT</u>	Font of the damage number
- int visibleTick	Initialized with 0, count the number of ticks that the particle is shown.
- int maxVisibleTick	Initialized with 60, maximum number of ticks that the particle will be shown.
- double x	X coordinate that the particle will be shown
- double y	Y coordinate that the particle will be shown
- int hp	The number that will be shown
- boolean isOfMonster	Show whether this damage is of monster

#### 4.2.2 Constructor

+ Damage(int hp, double x, double y, boolean isOfMonster)	Set the corresponding fields to the given parameters
---	--

#### 4.2.3 Method

+ void render(GraphicsContext gc)	Set color to orange if the damage is of monster and orchid otherwise. Set opacity according to visibleTick Draw number is hp is more than 0 and "MISS" otherwise Decrease the value of y by 0.5 Increase visibleTick by 1
+ boolean isVisible()	Return true if visibleTick is less than maxVisibleTick
+ boolean isExpired()	Return true if visibleTick is more or equal than maxVisibleTick

### 4.3 Class DisplayName

This class represents display name and level under damageable entities.

#### 4.3.1 Field

+ Font <u>NAME_FONT</u>	Font of display name
- DamageableEntity entity	Store the entity
- int visibleTick	Set to 180
- int maxVisibleTick	Set to 180

#### 4.3.2 Constructor

+ DisplayName(DamageableEntity entity)	Set the corresponding fields to the given parameters
--	--

#### 4.3.3 Method

+ void render(GraphicsContext gc)	If isVisible() is false and entity is not a player, return Otherwise, show name of the entity and increase visibleTick by 1.
+ boolean isVisible()	Return true if visibleTick is less than maxVisibleTick.
+ boolean isExpired()	Return true if entity is dead.
+ void resetVisible()	Set visibleTick to 0



## 4.4 Class HpBar

This class represents HP bar above the monster.

### 4.4.1 Field

+ <u>int WIDTH</u>	Set width to 80
+ <u>int HEIGHT</u>	Set height to 6
- DamageableEntity entity	Store the entity
- int visibleTick	Set to 180
- int maxVisibleTick	Set to 180

### 4.4.2 Constructor

+ HpBar(DamageableEntity entity)	Set the corresponding fields to the given parameters
----------------------------------	--

### 4.4.3 Method

+ void render(GraphicsContext gc)	If isVisible() is false, return Otherwise, show hp bar above the entity
+ boolean isVisible()	Return true if visibleTick is less than maxVisibleTick.
+ boolean isExpired()	Return true if entity is dead.
+ void resetVisible()	Set visibleTick to 0

## 4.5 Class LevelUp

This class does level up animation.

### 4.5.1 Field

+ <u>Font LEVEL_UP_FONT</u>	Font of display level up
+ Color TEXT_COLOR	Set color of level up font
- int visibleTick	Initialized with 0, count the number of ticks that the particle is shown.
- int maxVisibleTick	Initialized with 120, maximum number of ticks that the particle will be shown.

#### 4.5.2 Method

+ void render(GraphicsContext gc)	If visibleTick is less than 40, draw image of levelUpEffect at index of a half of visibleTick. Else if visibleTick is more than 80, draw image of levelUpEffect at index of a half of (maxVisibleTick - visibleTick). Otherwise, draw image of the last image in levelUpEffect arrays. If visibleTick more or equal than 60, set opacity to (maxVisibleTick - visibleTick)/60 Fill text with "LEVEL UP"
+ boolean isVisible()	Return true if visibleTick is less than maxVisibleTick.
+ boolean isExpired()	Return true if visibleTick is more or equal than maxVisibleTick

#### 4.6 Class NormalAttackEffect

This class does normal attack animation.

##### 4.6.1 Field

- int age	Initialized with 0, count the number of ticks that the particle is shown.
- int maxAge	Initialized with 30, maximum number of ticks that the particle will be shown.

##### 4.6.2 Method

+ void render(GraphicsContext gc)	Render attack images in front of the player, flip the image if the player is facing right.
+ boolean isVisible()	Return true if age is less than maxAge.
+ boolean isExpired()	Return true if age is more or equal than maxAge

#### 4.7 Class MultiKillEffect

This class does Multi Kill skill animation.

##### 4.7.1 Field

- int age	Initialized with 0, count the number of ticks that the particle is shown.
-----------	---

- int maxAge	Initialized with 30, maximum number of ticks that the particle will be shown.
--------------	---

#### 4.7.2 Method

+ boolean isVisible()	Return true if age is less than maxAge.
+ boolean isExpired()	Return true if the particle is not visible.
+ void render(GraphicsContext gc)	Set opacity to $(-1/\text{maxAge}) \times \text{age} + 1$ and draw skillMultiKill image in front of and same direction of a player .

### 4.8 Class PowerUpEffect

This class does Power Up skill animation.

#### 4.8.1 Field

- int age	Initialized with 0, count the number of ticks that the particle is shown.
- int maxAge	Initialized with 90, maximum number of ticks that the particle will be shown.

#### 4.8.2 Method

+ boolean isVisible()	Return true if age is less than maxAge.
+ boolean isExpired()	Return true if the particle is not visible.
+ void render(GraphicsContext gc)	Set opacity to 0.8 if age is less than 60. Otherwise, decrease opacity according to age variable to create fading animation. Draw fire image.

### 4.9 Class PowerOfJoeEffect

This class does Power of Joe skill animation.

#### 4.9.1 Field

- int age	Initialized with 0, count the number of ticks that the particle is shown.
- int maxAge	Initialized with 30, maximum number of ticks that the particle will be shown.



#### 4.9.2 Method

+ boolean isVisible()	Return true if age is less than maxAge.
+ boolean isExpired()	Return true if the particle is not visible.
+ void render(GraphicsContext gc)	Set opacity to $(-1/\text{maxAge}) * \text{age} + 1$ and draw skillPowerOfJoe image in front of and same direction of a player.

### 4.10 Class GradeAEffect

This class does Grade A for sure skill animation.

#### 4.10.1 Field

- Font CENTER_FONT	Font of letter A in the center of the screen
- int visibleTick	Initialized with 0, count the number of ticks that the particle is shown.
- int maxVisibleTick	Initialized with 240, maximum number of ticks that the particle will be shown.

#### 4.10.2 Method

+ boolean isVisible()	Return true if visibleTick is less than maxVisibleTick
+ boolean isExpired()	Return true if visibleTick is more than or equal to maxVisibleTick
+ void render(GraphicsContext gc)	Set opacity if maxVisibleTick-visibleTick is less than or equal to 30 to make fade out animation Draw image of the effect Draw letter A Play "Tada" sound when visibleTick is 30 Increase visibleTick by 1

## 5. Package model

### 5.1 Class Rectangle

#### 5.1.1 Field

# double x	X coordinate of the rectangle
# double y	Y coordinate of the rectangle

# double width	Width of the rectangle
# double height	Height of the rectangle

### 5.1.2 Constructor

+ Rectangle(double x, double y, double width, double height)	Set the corresponding fields to the given parameters
--	--

### 5.1.3 Method

+ boolean collideWith(Rectangle r)	Return true if rectangle r intersects with the current rectangle
+ boolean collideWith(double x, double y, double width, double height)	Return true if the rectangle is in the area of the rectangle with the specified property.
+ String toString()	Return x, y, width and height property of the rectangle in the term of string
Getters and setters of all fields	

## 5.2 Class Entity

This abstract class is a subclass of Rectangle that represents interactable things in the game main view.

### 5.2.1 Field

+ <u>int LEFT</u>	Constant value -1 that represents left direction
+ <u>int RIGHT</u>	Constant value 1 that represents right direction
# double velocityX	Velocity of the X-axis
# double velocityY	Velocity of the Y-axis
- Image image	Image of the entity
- Image imageL	Image of the entity facing left
- Image imageR	Image of the entity facing right
# double maxVelocityX	Maximum X-axis velocity that the entity can reach
# int facing	Current facing direction of the entity
# String name	Name of the entity

- Map map	Map that the entity currently in
-----------	----------------------------------

### 5.2.2 Constructor

+ Entity(String name, Image img, Map map, double x, double y)	Initialize Rectangle with the specified x, y and set width and height to the image's property Set name, image and map
+ Entity(String name, Image imgL, Image imgR, Map map)	Same as the third constructor with x and y be 0
+ Entity(String name, Image imgL, Image imgR, Map map, double x, double y)	Initialize Rectangle with the specifies x and y and set width and height to the image's property Set name, imageL, imageR and map to the given parameters Set facing direction to right

### 5.2.3 Method

+ void update()	Abstract method, will be updated every tick
+ void move()	Move entity with the specified velocity
+ void move(double x, double y)	Move entity with the specified x and y
+ void pushAccX(accX)	Add velocityX by accX Set back to maxVelocityX if the value exceeds the limit
+ void pushAccY(accY)	Add velocityY by accY
+ void render(GraphicsContext gc)	Draw entity image at the position x and y relative to the map's x and y
+ String toString()	Return simple class name
+ void setFacing(int facing)	Set facing and direction using another <code>setFacing()</code>
+ void setFacing(int facing, Image img)	Set both facing and image to the specified value
Getters of facing, velocityX, velocityY, name, image and map and setters of velocityX, velocityY and map	

## 5.3 Class DamageableEntity

This abstract class is a subclass of class Entity. It adds functionality to kill and to be damaged.

### 5.3.1 Field

# int level	Level of the entity
# int experience	Total experience gained/will be gained
- int maxHp	Maximum HP
- int hp	Current HP
- int maxMp	Maximum MP
- int mp	Current MP
- int attackDamageLow	Lower bound of attack damage
- int attackDamageHigh	Upper bound of attack damage
# List<Pair<Item,Double>> drops	List of pairs of item and chance of dropping item
- DisplayName displayName	Do the rendering of display name under the entity image job

### 5.3.2 Constructor

+ DamageableEntity(String name, Image imgL, Image imgR, model.map.Map map, int hp, int mp, int attackDamageLow, int attackDamageHigh)	Same as the second constructor but set x and y to 0
+ DamageableEntity(String name, Image imgL, Image imgR, model.map.Map map, double x, double y, int hp, int mp, int attackDamageLow, int attackDamageHigh)	Set corresponding fields to the given parameters Initialize displayName

### 5.3.3 Method

+ void damage(int hp)	Decrease HP of the entity Add damage particle reset displayName visibility
+ void forceKill()	Damage 99999 HP Remove from SharedEntity Add spawn loot to SharedEntity
+ void healHp(int hp)	Increase HP by the value of parameter Set to maxHp if hp is more than maxHp
+ void healMp(int mp)	Increase MP by the value of parameter Set to maxMp if mp is more than maxMp

+ void refillHp()	Increase HP to full
+ void refillMp()	Increase MP to fill
+ boolean isDead()	Return true if hp is equal or less than 0
+ List<ItemEntity> spawnLoot()	Random items to be spawned as loot using <code>Random.multipleRandomInList()</code> Create ItemEntity based on the random result
+ void useMp(int usedMp)	Decrease MP by usedMp Throw MpNotEnoughException if MP is not enough to use
+ void render(GraphicsContext gc)	Render entity image using superclass' one Render displayName
+ int getAttackDamage()	Return summation of attackDamageLow
+ String toString	Return string with the format " <u>Name</u> [HP: <u>hp/maxHp</u> , MP: <u>mp/maxMp</u> ]"
Getter of level, experience, maxHp, hp, maxMp, mp and drops and setter of setMaxHp, setMaxMp, setAttackDamageLow and setAttackDamageHigh	

## 5.4 Class ItemEntity

This class is a subclass of Entity that represents item that is in the map and dropped by monster, waiting for player to collect.

### 5.4.1 Field

- Item item	Item that will be in the player's inventory when collected
- int age	Initialized with 0, count total ticks since the item is dropped.
- int maxAge	Initialized with 3600, store maximum age of the item entity

### 5.4.2 Constructor

+ ItemEntity(Item item, Map map, double x, double y)	Set field to the corresponding parameters and set y-velocity to -6
--	--

### 5.4.3 Method

+ void update()	Increase age by 1
+ boolean isExpired()	Return true if age is more than or equal to maxAge
+ void render(GraphicsContext gc)	Draw image of the item and fade out when age is nearly expired
+ String toString()	Return string that shows property of the item entity
Getter of item	

## 5.5 Interface IUsable

This class is a template for skills and items.

### 5.5.1 Method

+ void use()	This method will be overridden by each item and skill.
+ void activate()	This method will be controlled by Skill class and Item class to create template of the using class.

## 6. Package model.player

### 6.1 Class Player

This class is a subclass of DamageableEntity.

#### 6.1.1 Field

# List<Image> imgWalking	Images of the player walking
# List<Image> imgCrying	Images of the player crying
# List<Image> imgWalkAndCry	Images of the player that is both walking and crying
# List<Image> imgAttack	Images of the player attacking
- boolean isWalking	Stores whether the player is walking
- boolean isJumping	Stores whether the player is jumping
- boolean isCrying	Stores whether the player is crying



# List<Skill> skills	Stores the collection of skills
- Rectangle attackArea	Stores the area of attacking of the player
- int walkTick	Initialized with 30, used to render walking animation
- int maxWalkTick	Initialized with 60, reset walkTick to 0 when it reaches this value
- int attackTick	Initialized with 0, the player can attack if this value is 0, this variable can be vary due to different skills
- int damageTick	Initialized with 60, the player which have this variable same as maxDamageTick can be damaged by monsters.
- int maxDamageTick	Initialized with 60
- int expDropTick	Count ticks of experience drops due to death
- Item[] inventory	Stores items in the form of array, null pointer means no item is in that inventory slot
- List<Buff> buffs	Collection of current buffs that is applied to the player
- boolean hasEverDead	Stores whether the player has ever dead or not

### 6.1.2 Constructor

+ Player(String name, Image imgL, Image imgR, List<Image> imgWalking, List<Image> imgCrying, List<Image> imgWalkAndCry, List<Image> imgAttack, Map map)	Same as the second constructor but initialize x and y with 0
+ Player(String name, Image imgL, Image imgR, List<Image> imgWalking, List<Image> imgCrying, List<Image> imgWalkAndCry, List<Image> imgAttack, Map map, double x, double y)	Set corresponding fields to the given parameters Set HP and MP using the value in Constants class Set attack damage using the method getAttackLow() and getAttackHigh() Initialized attackArea with the player's boundary Set maxVelocity to 4 Add NormalAttack to the skill

### 6.1.3 Method

+ void jump()	Set velocity to -10 and isJumping to true if the player is not jumping and is on floor
+ void jumpDown()	Move down 2 pixels and set isJumping to true if the player is not jumping and is on floor
+ boolean attack(AttackSkill skill)	<p>Return true if attacking success            If cannot attack now, attacking is failed            Get the list of monsters to be damaged by using map's collideDamageableEntity() with the parameter some property of this skill            For each monster in the list, damage monster with the damage value of product of normal damage value and attack multiplier            if the monster is dead, add experience to the player and remove monster from SharedEntity's list            Play sound of the attacked/dead monster</p>
+ void setMove(int direction)	<p>If player is crying,</p> <ul style="list-style-type: none"> <li>- If player is walking and walkTick is less than 5, setFacing with given direction and set image with imgWalkAndCry with the same given direction.</li> <li>- Otherwise, setFacing with given direction and set image with imgCrying with the same given direction.</li> </ul> <p>Else if player is walking,</p> <ul style="list-style-type: none"> <li>- If walkTick is less than 5, setFacing with given direction and set image with imgWalking with the same given direction.</li> </ul> <p>Otherwise, setFacing with given direction.            If player is attacking, setFacing with given direction and set image with imgAttack with the same given direction.</p>
+ void update()	<p>Set crying if HP is less than 20%            Update player by user's pressing keys            While polling key is available, update player by user's polled keys.            Increase walkTick if the player is walking.            Otherwise, set to 0.            If the player is being a ghost, decrease expDropTick and when it reaches zero, set</p>

	<p>to 15 and decrease experience value by 1. And if player's experience is less than 0, set back to 0.</p> <p>If the player's attackTick is more than 0, decrease by 1.</p> <p>If the player's damageTick is less than maxDamageTick, increase by 1. Otherwise, check for colliding monsters and damage player and set damageTick to 0 if there is a monster in range.</p> <p>Remove expired buff</p>
+ void updateByPressingKey()	<p>If the user is pressing left or right, set facing to that direction and push acceleration, the acceleration value is different by where the player is now.</p> <p>If the user is pressing space, check if the user is pressing down. If not, perform a normal jump, otherwise perform jumping down.</p> <p>If the user is pressing key C, collect the item that is colliding to the player and remove item entity that the item is collected.</p> <p>If the player is pressing up, perform a warp. That function will determine whether to warp by itself.</p>
+ void updateByPollKeys(KeyCode key)	<p>If the key is a number digit, activate item of that digit.</p> <p>If the key is A, activate skill 0.</p>
+ void collectItem(Item item)	<p>Find for existing stack of item that is the same type of the added one, meanwhile, find the empty space</p> <p>If there is a stack that is the same type of the new item and is not full, increase count of that stack.</p> <p>If no such stack available and there exists an empty space, add item to that space.</p> <p>If no space left, throw InventoryFullException</p>
+ void addExperience(int experience)	<p>If the player hasn't reached the maximum level, add experience with experience buff multiplier</p> <p>If the player's experience exceeds the maximum experience of that level</p> <ul style="list-style-type: none"> <li>- Increase the level</li> <li>- Decrease experience by the original maximum value</li> <li>- Set HP and MP to the higher one</li> </ul>

	using the value from Constants class - Refill HP and MP - Recalculate attack damage high and low - Play the level up sound - Add level up particle - If the player reaches maximum level, end game.
+ void addBuff(Buff buff)	If buff is found in buffs, refresh that buff Otherwise, add this buff to buffs
+ boolean canUseMp(int mp)	Return true if mp is equal or more than the given mp
+ boolean canAttack()	Return true if attackTick is equal or less than 0
+ int <u>getAttackLow(int level)</u>	Calculate the lower bound of attack damage of the specified level
+ int <u>getAttackHigh(int level)</u>	Calculate the upper bound of attack damage of the specified level
+ int getAttackDamage()	Return the product of random attack damage from superclass and buffs' multiplier
+ void damage(int hp)	Do the superclass's damage If the player has never dead before but is dead, clear KeyInput keys and show an alert.
+ Rectangle getAttackArea(AttackSkill skill)	Return attack area of the specified skill
+ Rectangle getAttackArea()	Return attack area of normal attack
+ void setName(String name)	If the given name does not null or equal "", set name with the given name.
Getter of isJumping, isWalking, isCrying, walkTick, skills, inventory and buffs and setter of isJumping	

## 6.2 Class CPEngineer

This is a subclass of Player

### 6.2.1 Field

- List<Image> imgWalkDead	Store the images of walking when
---------------------------	----------------------------------

	CPEnginner is dead.
- List<Image> imgFightDead	Store the images of fighting when CPEngineer is dead.

### 6.2.2 Constructor

+ CPEngineer(String name, Map map)	Call CPEngineer(name, map, 0, 0)
+ CPEngineer(String name, Map map, double x, double y)	Initialize player with the given name, ArrayList of walking, crying, walking and crying and fighting image. Set images of walking when player is dead with ArrayList type Set images of fighting when player is dead with ArrayList type Add MultiKill(), PowerUp(), PowerOfJoe() and GradeAForSure() skill to skills

### 6.2.3 Method

+ void setMove(int direction)	<p>If player is dead, setFacing with given direction and set image with imgWalkDead with the same given direction</p> <p>Else if player is crying,</p> <ul style="list-style-type: none"> <li>- If player is walking and walkTick is less than 5, setFacing with given direction and set image with imgWalkAndCry with the same given direction.</li> <li>- Otherwise, setFacing with given direction and set image with imgCrying with the same given direction.</li> </ul> <p>Else if player is walking,</p> <ul style="list-style-type: none"> <li>- If walkTick is less than 5, setFacing with given direction and set image with imgWalking with the same given direction.</li> </ul> <p>Otherwise, setFacing with given direction.</p> <p>If player is attacking,</p> <ul style="list-style-type: none"> <li>- If player is dead, setFacing with given direction and set image with imgFightDead with the same given direction.</li> <li>- Otherwise, setFacing with given direction and set image with imgAttack with the same given direction.</li> </ul>
-------------------------------	---

+ void updateByPressingKeys()	Update by pressing keys of the superclass If the user is pressing E, use skill 3.
+ void updateByPollKey(KeyCode key)	Update by poll key of the superclass If the key is Q, activate skill 1. If the key is W, activate skill 2. If the key is R, activate skill 4.

## 7. Package model.item

### 7.1 Class Item

This abstract class represents item in the game.

#### 7.1.1 Field

- String name	Name of the item
- String description	Description of the item
- Image image	Image of the item which will be shown in the status bar
# int count	Number of items
# int maxCount	Number of maximum items in the stack

#### 7.1.2 Constructor

+ Item(String name, Image img)	Set the corresponding fields to the given parameters, set count to 1 and set maxCount to 50
--------------------------------	---

#### 7.1.3 Method

+ boolean add(int n)	Increase count value by n if it doesn't exceed maxCount value and return true if adding successful.
+ boolean sameType(Item item)	Return true if the current item and the variable item is the same type.
+ void activate()	Use item by using method <code>use()</code> and decrease count variable if count is more than 0
Getters of name, description, image and count	



## 7.2 Class RedPotion

This class represents Red Potion item.

### 7.2.1 Constructor

+ RedPotion()	Initialize Item with the name “Red Potion” and image of red potion
---------------	--

### 7.2.2 Method

+ void use()	Increase player’s HP by 50 using Player’s <code>healHp()</code>
--------------	---

## 7.3 Class BluePotion

This class represents Blue Potion item.

### 7.2.1 Constructor

+ BluePotion()	Initialize Item with the name “Blue Potion” and image of blue potion
----------------	--

### 7.2.2 Method

+ void use()	Increase player’s MP by 20 using Player’s <code>healMp()</code>
--------------	---

## 7.4 Class FastLearnScroll

This class represents Blue Potion item.

### 7.2.1 Constructor

+ FastLearnScroll()	Initialize Item with the name “Fast Learn Scroll” and image of the scroll
---------------------	---

### 7.2.2 Method

+ void use()	Add <code>FastLearnBuff</code> to the player’s buff using the method <code>addBuff(Buff buff)</code> of Player object
--------------	---

# 8. Package model.monster

## 8.1 Class Monster

This abstract class is a subclass of `DamageableEntity` that is a template of all monster types in the game.

### 8.1.1 Field

- int age	Stores the number of ticks since the monster is spawned to render fading in animation of the monster (maximum 30)
- int aiDelay	Stores the number of ticks left that a monster stands still
- int walkTick	Stores the number of ticks that a monster will walk
- int futureFacing	Stores the facing direction of the monster since aiDelay drops to zero
- int aggressiveTick	Stores the number of ticks left that a monster follows player
- HpBar hpBar	Do the name and health bar rendering job
- List<Image> imgWalking	Stores the walking image

### 8.1.2 Constructor

+ Monster(String name, List<Image> imgWalking, Map map, double x, double y, int level, int hp, int mp, int atkLow, int atkHigh, int experience)	Set the corresponding fields to the given parameters and initialize hpBar
+ Monster(String name, List<Image> imgWalking, Map map, int level, int hp, int mp, int atkLow, int atkHigh, int experience)	Set the corresponding fields to the given parameters, set coordinate x and y to 0 and initialize hpBar

### 8.1.3 Method

+ int damage(int hp)	Decrease hp, show Damage particle and play damage sound using <code>super.damage (hp)</code> Reset time count of hpBar Set monster to be aggressive for 6 seconds (360 ticks) Set aiDelay and walkTick to 0 to force MonsterAI thread to recalculate walking path
+ boolean isAggressive()	Return true if aggressiveTick is more than 0
+ void update()	If age is less than 30, increase age by 1 (to make fade in animation) If aiDelay is more than 0, decrease aiDelay by 1

	If walkTick is more than 0 and aiDelay is equal to 0, decrease walkTick by 1 and then move monster in the direction of futureFacing using <code>pushAccX(0.5)</code> of map. Otherwise, set image to the standing position of the monster in the current facing direction.
+ void render(GraphicsContext gc)	Set opacity of the monster if age is less than 30 to make fade in animation Draw monster image Render HP bar
Getters and setters of aiDelay, walkTick and futureFacing	

## 8.2 Class MiniTree

This class represents Mini Tree monster.

### 8.2.1 Constructor

+ MiniTree(Map map, double x, double y)	Initialize monster with the name "Mini Tree", walking image of it Set level to 2 Set hp to 20 Set mp to 0 Set attack damage range from 3 to 8 Set experience to 4 Add two items to the drops list 1. Red Potion with drop chance 0.3 2. Blue Potion with drop chance 0.1
---	--

## 8.3 Class ProgMeth

This class represents Prog Meth monster

### 8.3.1 Constructor

+ ProgMeth(Map map, double x, double y)	Initialize monster with the name "Prog Meth", walking image of it Set level to 6 Set hp to 65 Set mp to 0 Set attack damage range from 6 to 12 Set experience to 10 Add two items to the drops list 1. Red Potion with drop chance 0.5 2. Blue Potion with drop chance 0.2
---	--

## 8.4 Class Node

This class represents Node monster

### 8.4.1 Constructor

+ Node(Map map, double x, double y)	Initialize monster with the name "Node", walking image of it Set level to 10 Set hp to 120 Set mp to 0 Set attack damage range from 10 to 20 Set experience to 17 Add two items to the drops list 1. Red Potion with drop chance 0.5 2. Fast Learn Scroll with drop chance 0.025
-------------------------------------	---

## 9. Package model.map

### 9.1 Class StructureItem

This class is a subclass of Rectangle class. It represents a block which all entities can stand on.

#### 9.1.1 Field

- boolean isPassable	If isPassable is true, this floor can go through down.
- boolean isSpawnable	If isSpawnable is true, monster can generate to this floor.

#### 9.1.2 Constructor

+ StructureItem(int x, int y, int width, int height, boolean isSpawnable, boolean isPassable)	Initialize the floor with position of the screen to put it (x,y), set width and height of the floor, set isSpawnable and isPassable.
+ StructureItem(int x, int y, int width, int height, boolean isSpawnable, boolean isPassable)	Initialize the floor with position of the screen to put it (x,y), set width and height of the floor, set isSpawnable and set isPassable to be true by default.

#### 9.1.3 Method

+ boolean collideWith(Rectangle r)	Return true when r is collide with this floor.
Getter of isPassable and isSpawnable	

## 9.2 Class MapStructure

This class is subclass of ArrayList<StructureItem> that adds some functionality to make it easy to get map structure info.

### 9.2.1 Field

- long serialVersionUID	For interface Serializable of super class
-------------------------	---

### 9.2.2 Constructor

+ MapStructure(StructureItem...items)	Initialize MapStructure and also add items to the ArrayList in the form of array
+ MapStructure(List<StructureItem> items)	Initialize MapStructure and also add items to the ArrayList in the form of List

### 9.2.3 Method

+ StructureItem collideWith(Entity entity)	Return structure item that the entity is collided with.
+ void addAll(StructureItem...items)	Add multiple structure items in the list

## 9.3 Class Portal

This class is a subclass of Rectangle class that represents portal in the game.

### 9.3.1 Field

- int counter	Initialized with 0, used to render portal image
- double xDest	X coordinate of the destination
- double yDest	Y coordinate of the destination
- Map destination	Destination map

### 9.3.2 Constructor

+ Portal(double xSrc, double ySrc, Map dest, double xDest, double yDest)	Set fields to the corresponding parameters
--	--

### 9.3.3 Method

+ void render(GraphicsContext gc)	Increase counter by 1 and set to 0 if counter is more than or equal to 30. Render portal images using count value
-----------------------------------	--

Getter of xDest, yDest and destination	
--	--

## 9.4 Class Map

### 9.4.1 Field

- Image img	Background image of a map
- double movementSpeed	Maximum map's x and y coordinate change per tick
- double gravity	Gravity of the map
- double groundFriction	Friction that is applied to the entities when is on ground
- double airFriction	Friction that is applied to the entities when is not on the ground
- double maxVelocityY	Maximum y velocity of entities when falling
- List<Class<? extends Monster>> monsterTypes	Collection of class of monsters that can be spawned in the map
- List<Portal> portals	Collection of portals in the map
- List<IParticle> particles	List of particles in the map
- MapStructure structure	Structure of the map
- MediaPlayer bgm	Background music of the map

### 9.4.2 Constructor

+ Map(Image img, AudioClip bgm, Class<? extends Monster>> monsterTypes)	Set x and y to 0 Set width and height to the width and height of the image Set img to the given img parameter Set bgm to a MediaPlayer that is the same source as given bgm parameter and set cycle count to infinite Add all monsters in parameter monsterTypes to the field
---	---

### 9.4.3 Method

+ void motion(Entity e)	Move entity Pull gravity to the entity Decelerate entity Move map Stop the entities if its y-velocity is more than
-------------------------	--



	zero and collides the map structure
+ void motionAll()	Motion all entities in the map
- void moveMap()	Change coordinate of the map if the player is going near the screen border
- void moveEntity(Entity e)	Move entity If entity is out of map border, set to the map border.
+ void pushAccX(Entity e, double accX)	Accelerate entity in the x-axis with the value of friction and accX
- void pullGravity(Entity e)	Accelerate entity in the y-axis with the value of gravity
- void decelerate(Entity e)	Decrease X velocity of the entity by the value of friction. Friction is groundFriction if the entity is on the floor and airFriction otherwise.
+ boolean isOnFloor(Entity e)	Return true if the player collides to the map structure or exceed height of the map
+ List<DamageableEntity> collideDamageableEntity(Rectangle r, int limit)	Return list of DamageableEntity which rectangle intersects. The list has the maximum value of limit parameter.
+ ItemEntity collideItemEntity(Rectangle r)	Return ItemEntity which rectangle intersects.
+ Portal collidePortal(Rectangle r)	Return portal which rectangle intersects.
+ void warpIn()	Fade in the background music for 500 milliseconds using <code>AudioPlayer.fadeIn()</code>
+ void warpOut()	Fade in the background music for 500 milliseconds using <code>AudioPlayer.fadeOut()</code>
+ void update()	Remove all particles that is expired Update all entities If item entity is expired, remove from the SharedEntity singleton.
+ void render(GraphicsContext gc)	Draw map background Draw player Draw all entities in the map Draw all portals in the map Draw all particles in the map

+ void spawnRandom()	Create a random monster from monsterTypes Weighted random in structures that is spawnable using the weight as the width of the structure Move monster to the structure that is randomed
+ double getFriction(Entity e)	Get friction of the entity (consider whether the entity is on the floor or not)
Getters of structure, maxVelocityY, particles and portals	

## 9.5 Class Garden

### 9.5.1 Constructor

+ Garden()	Initailize Garden map with gardenBackground, gardenBgm and MiniTree monster. Add structure items to the map in order to creating floor blocks.
------------	---

## 9.6 Class Building4

### 9.6.1 Constructor

+ Building4()	Initailize Building4 map with building4Background, building4Bgm and ProgMeth monster. Add structure items to the map in order to creating floor blocks.
---------------	--

## 9.7 Class SkyCafe

### 9.7.1 Constructor

+ SkyCafe()	Initailize SkyCafe map with skycafeBackground, skycafeBgm and Node monster. Add structure items to the map in order to creating floor blocks.
-------------	--

## 10. Package sharedObject

### 10.1 Class SharedEntity

Singleton of list of all entities in the game.

#### 10.1.1 Field

- <u>SharedEntity</u> instance	Singleton of SharedEntity class
- List<Entity> entities	Stores all entities of the game (excluding player)

#### 10.1.2 Method

+ List<Entity> getEntitiesOfMap(Map map)	Get list of entities that is in the specified map
+ List<Entity> getEntitiesOfCurrentMap()	Get list of entities in the current map
+ List<Entity> getMonsterOfCurrentMap()	Get list of monsters in the current map
+ void add(Entity e)	Add entity to the list
+ void addAll(Collection<? extends Entity> items)	Add all entities in the collection to the list
+ void remove(Entity e)	Remove entity from the list
+ void clear()	Clear all entities form the list
+ <u>SharedEntity</u> getInstance()	Getter of instance

## 11. Package skill

### 11.1 Class Skill

This abstract class implements IUsable interface. It is a template of all skill classes and attacking methods.

#### 11.1.1 Field

- long lastUsedTimeMillis	Stores last timestamp that the skill is used.
---------------------------	---

#### 11.1.2 Method

+ String getName()	Abstract method that returns name of the skill
--------------------	--

+ String getDescription()	Abstract method that returns description of the skill
+ int getMpUse()	Abstract method that returns MP usage of the skill
+ int getCooldownTimeMillis()	Abstract method that returns cooldown time
+ boolean shouldUse()	Return true if the skill is allowed to use (ignore MP and Cooldown) (default always true)
+ long getRemainingCooldownTimeMillis()	Return remaining cooldown time in milliseconds
- void checkCooldown()	Throws CooldownException if the skill cannot be used now due to cooldown
+ void activate()	Return if <code>shouldUse()</code> is true Check cooldown using <code>checkCooldown()</code> Use MP with the value of <code>getMpUse()</code> Use skill using the method <code>use()</code> Update <code>lastUsedTimeMillis</code>

## 11.2 Class AttackSkill

This is abstract class and it's a subclass of Skill

### 11.2.1 Method

+ double getDamageMultiplier()	Abstract method that returns multiplier of damage.
+ double getAttackRange()	Abstract method that returns the range of attacking.
+ int getMaxEntity()	Abstract method that returns the maximum number of entity to be affected by the skill.
+ int getCooldownTick()	Abstract method that returns cooldown for attacking
+ boolean shouldUse()	Return true if and player can attack
+ void use()	Call player to attack with this skill

## 11.3 Class NormalAttack

This class is a subclass of AttackSkill that represents attacking using key A on the keyboard.

### 11.3.1 Method

+ String getName()	Return string "Normal Attack"
+ String getDescription()	Return string ""
+ double getDamageMultiplier()	Return 1 since no extra attack damage
+ double getAttackRange()	Return 40 since this skill can attack with the monster that in range 40px from player
+ int getMaxEntity()	Return 1 since this skill can attack only 1 monster
+ int getMpUse()	Return 0 since this skill isn't use MP
+ int getCooldownTimeMillis()	Return 0
+ void use()	Utilize <code>use()</code> of its superclass Add particle NormalAttackEffect
+ int getCooldownTick	Return 30

## 11.4 Class MultiKill

This class is a subclass of AttackSkill that represents attacking using key Q on the keyboard.

### 11.4.1 Method

+ String getName()	Return string "Multi Kill"
+ String getDescription()	Return string "4 MP, Hit 4 monsters +50%"
+ double getDamageMultiplier()	Return 1.5
+ double getAttackRange()	Return 60 (This skill can hit monster further away than normal attack)
+ int getMaxEntity()	Return 4
+ int getCooldownTick()	Return 30
+ int getMpUse()	Return 4 since this skill uses 4 MP
+ int getCooldownTimeMillis()	Return 500
+ void use()	Utilize <code>use()</code> of its superclass Add particle MultiKill

## 11.5 Class PowerUp

This class is a subclass of Skill that represents Power Up buff skill.



### 11.5.1 Method

+ String getName()	Return string "Power Up"
+ String getDescription()	Return string "12 MP, +50% attack for 90s"
+ void use()	Add buff to the player Add particle PowerUpEffect Play fire sound
+ int getMpUse()	Return 12 since this skill uses 12 MP
+ int getCooldownTimeMillis()	Return 0

## 11.6 Class PowerOfJoe

This class is a subclass of AttackSkill that represents attacking using key E on the keyboard.

### 11.6.1 Method

+ String getName()	Return string "Power of Joe"
+ String getDescription()	Return string "2 MP, -50% hit continuously"
+ double getDamageMultiplier()	Return 0.5
+ double getAttackRange()	Return 40
+ int getMaxEntity()	Return 1
+ int getCooldownTick()	Return 8
+ int getMpUse()	Return 2 since this skill uses 2 MP
+ int getCooldownTimeMillis()	Return 0
+ void use()	Utilize <code>use()</code> of its superclass Add particle PowerOfJoeEffect

## 11.7 Class GradeASure

This class is a subclass of Skill that represents skill "Grade A for sure".

### 11.7.1 Method

+ String getName()	Return string "Grade A for sure"
+ String getDescription()	Return string "40 MP, Kill all monsters"
+ void use()	Force kill all monsters in the current map Add particle GradeAEffct

+ int getMpUse()	Return 40 since this skill uses 40 MP
+ int getCooldownTimeMillis()	Return 120000 since cooldown is 2 minutes

## 12. Package buff

### 12.1 Class Buff

This class represents buff that is shown on the top right corner of the screen.

#### 12.1.1 Field

- long startTimeMillis	Timestamp that the buff is created
- long buffTimeMillis	Duration of buff in milliseconds
- Image image	Buff image that is shown in the top right corner

#### 12.1.2 Constructor

+ Buff(long buffTimeMillis, Image image)	Set buffTimeMillis and image to the corresponding parameters Set startTimeMillis to the current timestamp
--	--

#### 12.1.3 Method

+ double getAttackMultiplier()	Abstract method, return increase in attack multiplier
+ double getExperienceMultiplier()	Abstract method, return increase in experience multiplier
+ void refresh()	Set startTimeMillis to current time
+ boolean isExpired()	Return true is the buff is expired
+ int getRemainingTime()	Return
+ Image getImage()	Getter of image

### 12.2 Class PowerUpBuff

This class represents Power Up buff which can be obtained by using skill Power Up (key W)

#### 12.2.1 Constructor

+ PowerUpBuff()	Initialize buff with the time of 90000 milliseconds and set the image to the corresponding image
-----------------	--

### 12.2.2 Method

+ double getAttackMultiplier()	Return 0.5
+ double getExperienceMultiplier()	Return 0

## 12.3 Class FastLearnBuff

This class represents Fast Learn buff which can be obtained by using the item Fast Learn Scroll (dropped from Node)

### 12.3.1 Constructor

+ FastLearnBuff()	Initialize buff with the time of 60000 milliseconds and set the image to the corresponding image
-------------------	--

### 12.3.2 Method

+ double getAttackMultiplier()	Return 0
+ double getExperienceMultiplier()	Return 0.5

## 13. Package ui

### 13.1 Class StartScene

This class is a subclass of Scene class. It controls everything on the screen when the game is starting.

#### 13.1.1 Field

- Pane root	Store pane of the scene graph
- Canvas canvas	Canvas for draw start screen
- TextField nameField	TextField for input your character's name
- Button backBtn	Button to back to start screen
- Button startBtn	Button for start game
- Timeline screenloop	Store timeline for play animation changing screen
- int page	Store page (0) 0 : Draw startscreen with play button to canvas 1 : Draw input character's name to canvas 2 : Set scene to game scene

- int animationTick	Store tick for keyframe to play animation for changing screen from start screen to input character screen, input character screen to game screen and input character screen back to start screen
---------------------	--

### 13.1.2 Constructor

+ StartScene()	<p>Initial start screen scene with play button, setting keyframe and add it to timeline</p> <p>In keyframe,</p> <ul style="list-style-type: none"> <li>- If page is equal 0 or 2, then decrease animationTick</li> <li>- If page is equal 1, then increase animationTick</li> </ul> <p>Draw image with startscreen, playbutton, and input character screen</p> <ul style="list-style-type: none"> <li>- If page is equal 2, play animation to game screen</li> <li>- If page is equal 1 and animationTick is equal 30 [on character input screen], set nameField, backBtn and startBtn to visible</li> <li>- If page is equal 2 and animationTick is equal 0 [successfull playing animation to game scene], set scene to gamescene [Get gamescene from Main]</li> </ul> <p>Add canvas, nameField, backBtn and startBtn to root</p>
----------------	--

### 13.1.3 Method

- void addBackBtnHandler()	Hide input character's name, back button and start button then play screenloop to play animation back to start screen
- void addCanvasEventHandler()	Handle animation canvas when moving or clicking [isOnPlayButton method] on play button when clicking on play button, play sceenloop for playing animation to input character's screen
- void addStartBtnHandler()	Handle when clicking on "Start game!", check nameField then play screenloop to play animation to game scene
- boolean isOnPlayButton(MouseEvent)	Return true when mouse curson is on play button

## 13.2 Class GameScene

This class is subclass of Scene. It controls updating frames and logic periodically.

### 13.2.1 Constructor

+ <u>GameScene()</u>	Initialize game scene with size 1000x600 Create canvas with size 1000x600 and add it to this scene Use KeyInput to bind this scene Handle mouse moving to show the Tooltip Create keyframe, <ul style="list-style-type: none"><li>- If game is running<ul style="list-style-type: none"><li>- If game is not pausing, then motion player, motionAll and update</li><li>- Render canvas</li></ul></li></ul> Create timeline, add keyframe to the timeline and play timeline
----------------------	--

## 13.3 Class StatusBar

This class does the rendering job of the status bar, message and time.

### 13.3.1 Field

+ <u>double HEIGHT</u>	Height of the status bar (60)
+ <u>double EXPERIENCE_HEIGHT</u>	Height of the experience bar (5)
+ <u>double HP_MP_WIDTH</u>	Width of the full HP and MP bar (200)
+ <u>double HP_MP_HEIGHT</u>	Height of the HP and MP bar (16)
+ <u>double HP_MP_X</u>	X coordinate of HP and MP bar (140)
+ <u>double HP_Y</u>	Y coordinate of HP bar (related to top of status bar) (10)
+ <u>double MP_Y</u>	Y coordinate of MP bar (related to top of status bar) (29)
+ <u>double SKILL_ITEM_Y</u>	Y coordinate of skill icons and item icons (related to top of status bar) (10)
+ <u>double ITEM_X</u>	X coordinate of the first item icon
- <u>Color BACKGROUND_COLOR</u>	Background color of the status bar
- <u>Color EXPERIENCE_COLOR</u>	Color of the experience bar
- <u>Color HP_COLOR</u>	Color of HP bar
- <u>Color MP_COLOR</u>	Color of MP bar

- <u>Font HP_MP_LABEL_FONT</u>	Font of HP and MP label
- <u>Font HP_MP_BAR_FONT</u>	Font of the value of HP and MP on the bar
- <u>Font EXPERIENCE_FONT</u>	Font of the experience value
- <u>Font KEYBOARD_FONT</u>	Font of the keyboard keys
- <u>Font MESSAGE_FONT</u>	Font of the message
- <u>Font ITEM_FONT</u>	Font of the item count
- <u>double hpWidth</u>	Current width of HP bar
- <u>double mpWidth</u>	Current width of MP bar
- <u>double expWidth</u>	Current width of experience bar
- <u>Player player</u>	Current player

### 13.3.2 Method

+ <u>void render(GraphicsContext gc)</u>	<p>Set player to the current player  Fill background of the status bar  Set expWidth using the formula (current experience/experience in that level + 4 * expWidth)/5. This causes animation to occur.  Draw experience bar  Set hpWidth and mpWidth using the formula similar to expWidth to create animation  Draw HP and MP bar  Draw Skill images  Draw black transparent rectangle to represent skill cooldown  Draw white item background, item icon and item count number if count is more than 1  Draw remaining text using method <code>drawText(gc)</code></p>
+ <u>void drawText(GraphicsContext gc)</u>	<p>Draw experience value, HP label, MP label, HP value, MP value, skills' keyboard keys and items' keyboard keys  Draw message above the status bar  Draw time passed in the top center of the screen</p>

## 13.4 Class Tooltip

This class does the job of rendering tooltip.

### 13.4.1 Field

- <u>double x</u>	Store x value of the cursor
- <u>double y</u>	Store y value of the cursor
- <u>String title</u>	Store title of the tooltip to be shown
- <u>String description</u>	Store description of the tooltip to be shown
- <u>boolean shouldShow</u>	Store true if the current tick will show the tooltip and false otherwise
- <u>Color BACKGROUND_COLOR</u>	Color of the background of the tooltip
- <u>Font TITLE_FONT</u>	Font of title text
- <u>Font DESCRIPTION_FONT</u>	Font of description text

### 13.4.2 Method

+ <u>void update(double x, double y)</u>	Update the value of x and y Check if current cursor position is on the skill icons or item icons Update title and description if the cursor is on skill or item Update shouldShow value
- <u>boolean isMouseOnSkill(int i)</u>	Return true if cursor is on the i-th skill's icon
- <u>boolean isMouseOnItem(int i)</u>	Return true if cursor is on the i-th item icon
+ <u>void render(GraphicsContext gc)</u>	Render background rectangle, title and description if shouldShow is true

## 14. Package constants

### 14.1 Class Constants

This class contains integer constants in the game.

#### 14.1.1 Field

+ <u>int WINDOW_WIDTH</u>	Store width of window (1000)
+ <u>int WINDOW_HEIGHT</u>	Store height of window (600)
+ <u>int LEVEL_EXPERIENCE[21]</u>	Store the maximum experience value for each level which need to reach for going to next level [ 0, 10, 16, 25, 40, 62, 100, 161, 245, 376,

	504, 686, 913, 1178, 1645, 2305, 3207, 4555, 6102, 8047, 10385]
<u>+ int MAX_LEVEL</u>	Store the max level, calculate from the size of LEVEL_EXPERIENCE - 1
<u>+ int LEVEL_HP</u>	Store the maximum health point for each level [0, 50, 60, 75, 90, 110, 130, 155, 180, 210, 240, 275, 310, 350, 390, 435, 480, 530, 580, 635, 690]
<u>+ int LEVEL_MP</u>	Store the maximum magical point for each level [0, 10, 12, 15, 20, 25, 30, 36, 42, 50, 58, 66, 76, 86, 98, 110, 125, 140, 155, 170, 185]

## 14.2 Class Images

This class contains a collection of images of the game.

### 14.2.1 Field

<u>+ Image startscreen</u>	Store the image start screen background
<u>+ Image playbutton</u>	Store the image playbutton on start screen
<u>+ Image playbutton_highlight</u>	Store the image playbutton with highlight on start screen
<u>+ Image gardenBackground</u>	Store the image garden background map, set width to 1300 and height to 900
<u>+ Image skycafeBackground</u>	Store the image sky-cafe background map, set width to 1300 and height to 900
<u>+ Image building4Background</u>	Store the image building4 background map, set width to 1300 and height to 900
<u>+ Image[] portal</u>	Store all images of portal to play animation
<u>+ Image monsterProgmethR</u>	Store the Prog Meth monster image on the right side
<u>+ Image monsterProgmethL</u>	Store the Prog Meth monster image on the left side
<u>+ Image monsterMiniTreeUp</u>	Store the Mini Tree monster image which take hand up
<u>+ Image monsterMiniTreeDown</u>	Store the Mini Tree monster image which take hand down



+ <u>Image monsterNodeL</u>	Store the Node monster image on the left side
+ <u>Image monsterNodeLWalk</u>	Store the Node monster image on the left side with walking
+ <u>Image monsterNodeR</u>	Store the Node monster image on the right side
+ <u>Image monsterNodeRWalk</u>	Store the Node monster image on the right side with walking
+ <u>Image cpEngPlayerL</u>	Store the CP Engineer player image on the left side
+ <u>Image cpEngPlayerWalkL</u>	Store the CP Engineer player image on the left side with walking
+ <u>Image cpEngPlayerCryL</u>	Store the CP Engineer player image on the left side with crying
+ <u>Image cpEngPlayerWalkCryL</u>	Store the CP Engineer player image on the left side with walking and crying
+ <u>Image cpEngPlayerFightL1</u>	Store the CP Engineer player image on the left side with fighting
+ <u>Image cpEngPlayerDeadL</u>	Store the CP Engineer player image on the left side when CP Engineer player died
+ <u>Image cpEngPlayerDeadFightL</u>	Store the CP Engineer player image on the left side when CP Engineer player died and he is fighting
+ <u>Image cpEngPlayerR</u>	Store the CP Engineer player image on the right side
+ <u>Image cpEngPlayerWalkR</u>	Store the CP Engineer player image on the right side with walking
+ <u>Image cpEngPlayerCryR</u>	Store the CP Engineer player image on the right side with crying
+ <u>Image cpEngPlayerWalkCryR</u>	Store the CP Engineer player image on the right side with walking and crying
+ <u>Image cpEngPlayerFightR1</u>	Store the CP Engineer player image on the right side with fighting
+ <u>Image cpEngPlayerDeadR</u>	Store the CP Engineer player image on the right side when CP Engineer player died
+ <u>Image cpEngPlayerDeadFightR</u>	Store the CP Engineer player image on the

	right side when CP Engineer player died and he is fighting
+ <u>Image redPotionItem</u>	Store the image of red potion item
+ <u>Image bluePotionItem</u>	Store the image of blue potion item
+ <u>Image fastLearnItem</u>	Store the image of fast learn item
+ <u>Image[] normalAttackEffect</u>	Store images of normal attacking effect
+ <u>Image[] levelUpEffect</u>	Store images of level up effect
+ <u>Image[] powerUpEffect</u>	Store images of power up effect skill
+ <u>Image[] gradeAEffect</u>	Store images of grade A effect skill
+ <u>Image fastLearnBuff</u>	Store the icon of fastLearnBuff skill
+ <u>Image skillMultiKill</u>	Store the icon of multi kill skill
+ <u>Image skillPowerUp</u>	Store the icon of power up skill
+ <u>Image skillPowerOfJoe</u>	Store the icon of power of Joe skill
+ <u>Image skillGradeA</u>	Store the icon of grade A skill

#### 14.2.2 Static Block

Set normalAttackEffect, levelUpEffect, powerUpEffect, gradeAEffect and portal image to array of each variables with ordered.

### 14.3 Class Sounds

This class contains a collection of all audio in the game.

#### 14.3.1 Field

+ <u>AudioClip punchSound</u>	Sound of attacking
+ <u>AudioClip deadSound</u>	Sound of monster death
+ <u>AudioClip levelUpSound</u>	Sound of level up
+ <u>AudioClip tadaSound</u>	Sound of "Grade A for sure" skill
+ <u>AudioClip fireSound</u>	Sound of "Power Up" skill
+ <u>AudioClip gardenBgm</u>	Background music of Garden map
+ <u>AudioClip building4Bgm</u>	Background music of Building 4 map
+ <u>AudioClip skycafeBgm</u>	Background music of Sky Cafe map

### 14.3.2 Static Block

Set punchSound volume to 25%

## 15. Package controller

### 15.1 Class GameManager

This class contains the main logic of the game.

#### 15.1.1 Field

- <u>GameManager</u> instance	Singleton of GameManager class
- boolean isGameRunning	Stores the value whether the game is running
- boolean isPausing	Stores the value whether the game is pausing
- List<Map> maps	Collections of all maps
- Player player	Return the current player
- Map currentMap	Stores the current map that the player is currently in
- int warpTick	Stores the number of ticks since the player start warping, this is for animation and delay purpose
- int maxWarpTick	Maximum value of warpTick, when warpTick reach this value, the player can warp again
- MonsterGen monsterGen	Store the thread that controls monster generation
- MonsterAi monsterAi	Store the thread that controls monster artificial intelligence
- String message	Store the message which is shown over the status bar
- long startTimeMillis	Timestamp that the game is started

#### 15.1.2 Constructor

+ GameManager()	Generate maps Bind portal over the maps Start monster generation thread
-----------------	---

	Start monster artificial intelligence thread Create new player
--	---

### 15.1.3 Method

+ void render(GraphicsContext gc)	Draw the current map Draw black rectangle that indicates warping if the player is warping Draw buff images and its remaining time Draw status bar Draw tooltip
+ void update()	Increase warpTick is the player is warping Update the current map
- void generateMap()	Initialize Garden, Building4 and SkyCafe class Set current map to the first map
- void bindPortal()	Create portal from Garden to Building4 Create portal from Building4 to Garden Create portal from Building4 to SkyCafe Create two portals from SkyCafe to Building4
+ boolean isWarping()	Return true if warpTick is less than maxWarpTick
+ boolean shouldWarp()	Return true if the player is not warping and is standing on the portal
+ void warp()	If the player can warp, set warpTick to 0 and warp out of the current map
- void moveOfWarp()	Set the current map to the portal's destination Warp in to the map that the player is standing on Set position x and y to the portal destination's x and y
+ boolean shouldJumpDown()	Return true if the player is standing on a structure and the structure allows the player to pass
+ void startGame()	Set isGameRunning to true Set start time Warp in to the current map
+ void stopGame()	Set isGameRunning to false Stop monster generation thread and ai thread

	Clear KeyInput list Clear SharedEntity list Create new start scene Show information alert to end the game, this contains total used time and score After the user closes the alert, set scene to start scene, create new instance of GameManager and warp out of the current map.
+ void terminate()	Interrupt monsterGen and monsterAi thread
+ void setMessage(String message)	Set message with the given message if the message is not null.
Getter of maps, player, currentMap, instance, isGameRunning, isPausing, message and startTimeMillis and setter of player, currentMap, isGameRunning and isPausing	

## 15.2 Class MonsterGen

This class is a subclass of Thread class. It does monster generating job.

### 15.2.1 Constructor

+ MonsterGen()	Initialize and start a thread which randomly generates monster in the current map every 1400 milliseconds using <code>spawnRandom()</code> method in Map class.
----------------	---

## 15.3 Class MonsterAi

This class is a subclass of Thread class. It does monster artificial intelligence job.

### 15.3.1 Constructor

+ MonsterAi()	Initialize and start a thread which randomly apply monster's walking time, its walking direction and the amount of time it stops. In case that the monster is aggressive (monster's <code>isAggressive()</code> is true), apply walking direction towards player and let the monster walk that way for half a second.
---------------	---

## 16. Package exception

### 16.1 Class CooldownException

Throw this exception when the player use the skill that cooldown time isn't out.

### 16.2 Class InventoryFullException

Throw this exception when the player try to collect item when the player's inventory is full.

### 16.3 Class ListEmptyException

Throw this exception when passing empty list to the singularly random function. That is

```
Random.weightedRandomInList(List<Pair<T,Double>> list)
```

### 16.4 Class MpNotEnoughException

Throw this exception when a player try to use a skill that requires more MP that he currently has.

### 16.5 Class NegativeWeightedRandomException

Throw this exception when weighted random get a negative value.