**Zenith Chronicle**

**Documentation**

**Presented By**

Sirapoom Leesuravanich 6330539421

Panithi Khamwangyang 6330301321

**2110215 Programming Methodology**

**Semester 1 Year 2021**

**Chulalongkorn University**

**Zenith Chronicle – RPG hack&slash 2D-platformer Game**

**Introduction**

Zenith Chronicle is a game where player take the role of a treasure hunter to find the lost artifacts that disappeared during the old war of the old gods. With various types of artifacts and power ups found on the way, players have to make their ways to the end of the journey and collect all of the artifacts.

**Main Character**

This is a character and only character   
that player can control. He always spawn  
with 100 HP.

**Powerup**

Powerups are items found on the ground. Each one has  
their own effects and is stackable if one has multiples.  
There are 4 types of powerup.



Red potion : instantly recover some health points for player.

Apple : give player a small attack buff.

Blue fish : give player a small movement buff.

Red flower : give player a small jump height buff.

**Artifact**

By slaying enemies, they will occasionally drop artifacts which will have its own effects while holding and is stackable. There are 6 types of artifacts.



Armor : give boosts to player’s max health points.

Sword : give boosts to player’s attack damage.

Boots : give boosts to player’s movement speed.

Ring : give boosts to player’s jump height.

Necklace : give boosts to player’s dash speed.

Shield : give boosts to player’s health point regeneration.

**Enemy**

There are two types of enemies in this game, which is boss and monster. Boss will be the only enemy in boss stage.   
 The enemies’ strength base are base on difficulty and is stronger in every stage.

**Monster**

Monster is a type of enemy and only contain 2 types : Skeleton and Mushroom. They gain HP, attack points and movement speed as the game progress.

**Boss**

Boss is a type of enemy and this game has only one boss. He gains HP, attack points and attack speed as the game progress.

**Game Control**

(How to play picture)

**Gameplay & Scene**

**Main Menu**

Figure 7 : Main Menu

When player open the game, they will see the Main Menu. This scene has 4 buttons which is “Start”, which will show the select game difficulty scene, “How to play”, to show how to play the game, “Credit”, to show credit, and “Quit”, which will end the game.

**How-to-play Scene**

Figure 8 : How-to-play

Only one option player can do is to go back to the Main Menu by press the “Back” button or press esc.

**Credit Scene**

Figure 9 : Credit Scene

Same as the How-to-play scene, player have only one option to go back to Main Menu by press the “Back” button or press esc.

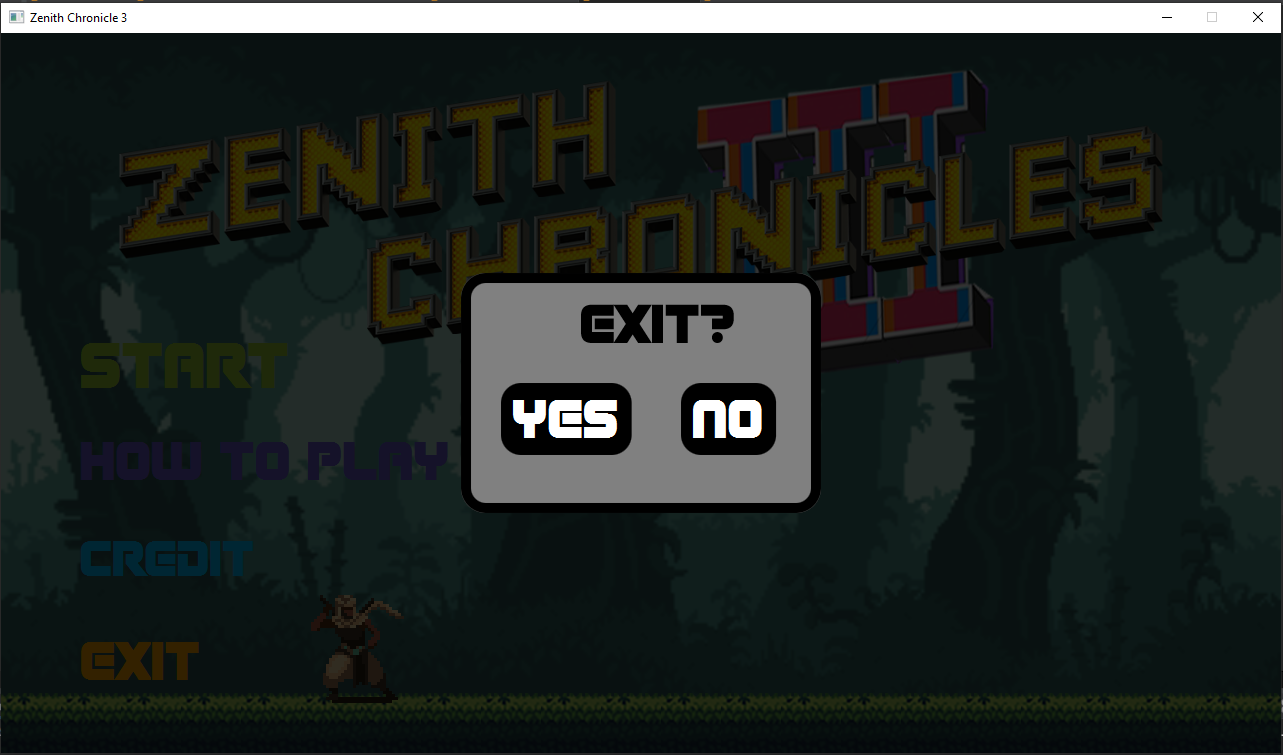
**Exit scene**

Figure 10 : Exit scene

Player have two options to end the game or to go back to the main menu.

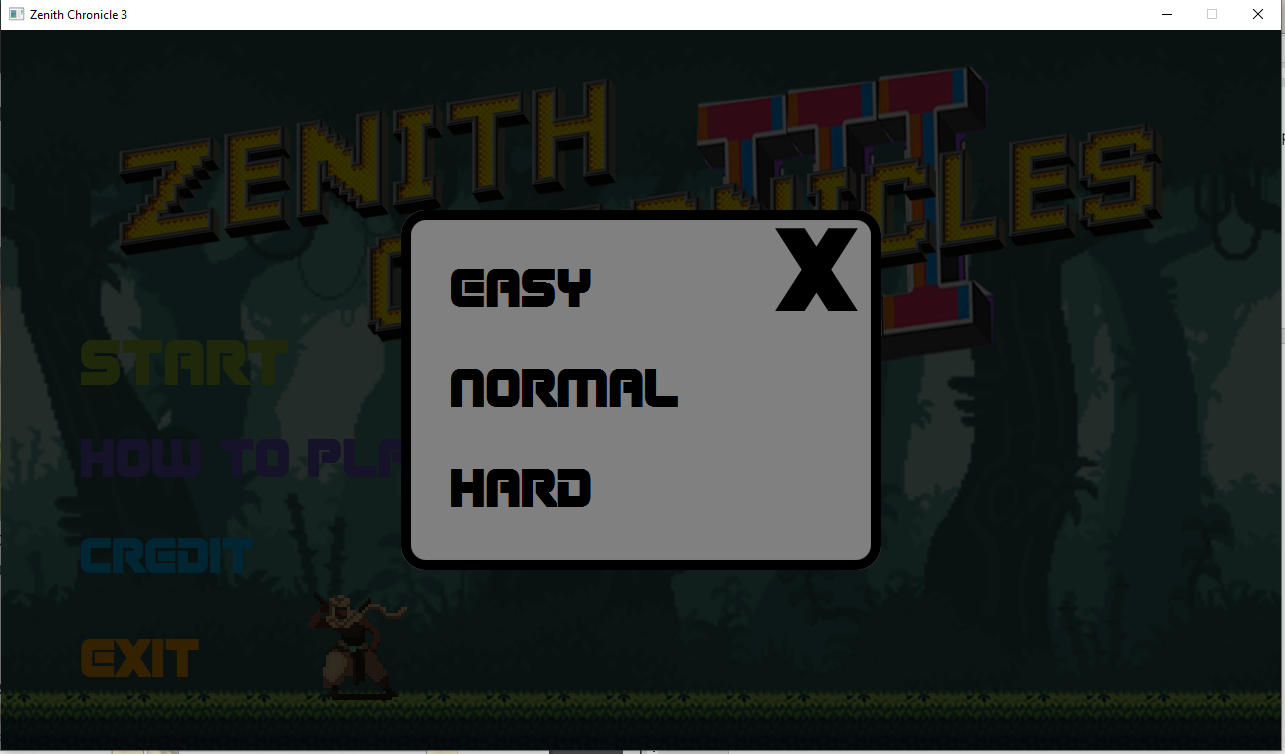
**Select difficulty scene**

Figure 11 : Select difficulty scene

Player have four options : Choose one of the three the difficulty and go to Playing Scene or they can choose to go back to main menu.

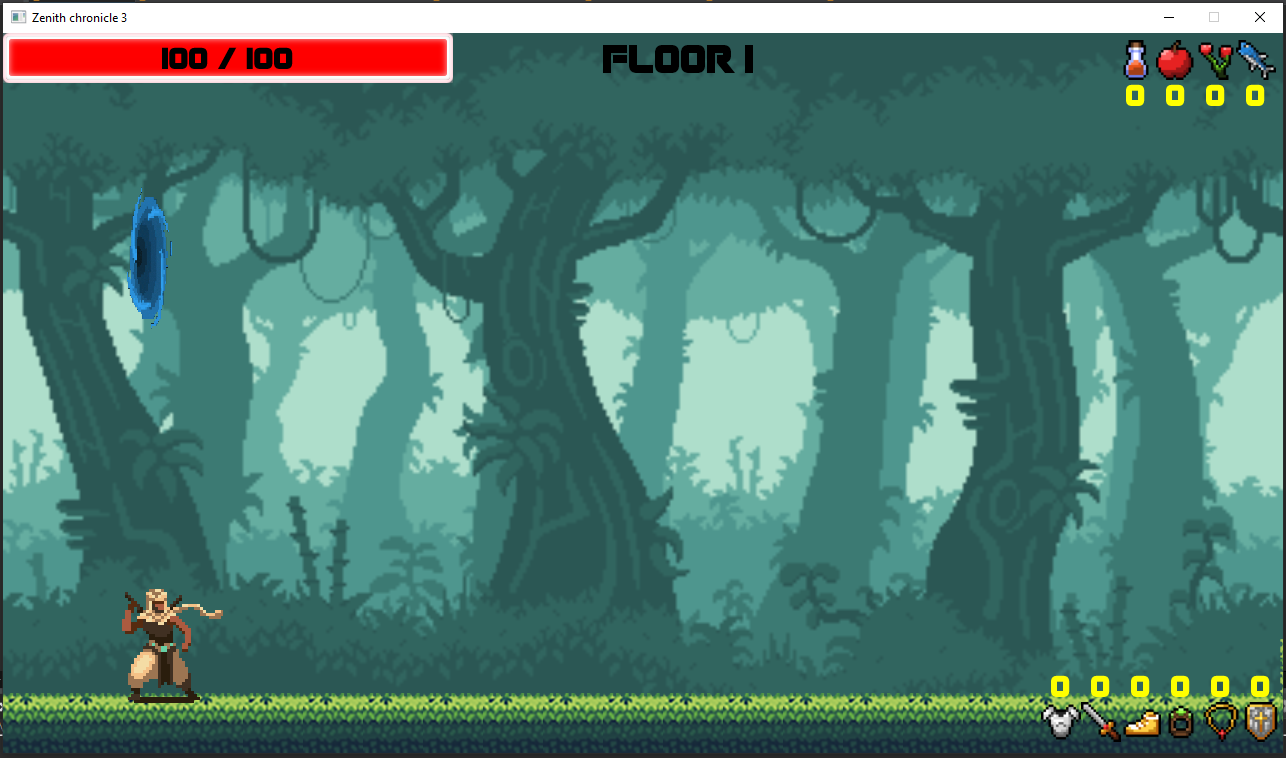
**Playing scene**

Figure 12 : Playing scene

This is game scene where the game will be playing. Player will always spawn on the left side of the screen. Top-left corner show health bar of the player. Top-right corner show the temporary buffs that player currently have. And button-right corner will show the artifact that player had collect throughout the game.

When the player make their way to the right-end of the scene, They will find a new portal to the next floor.

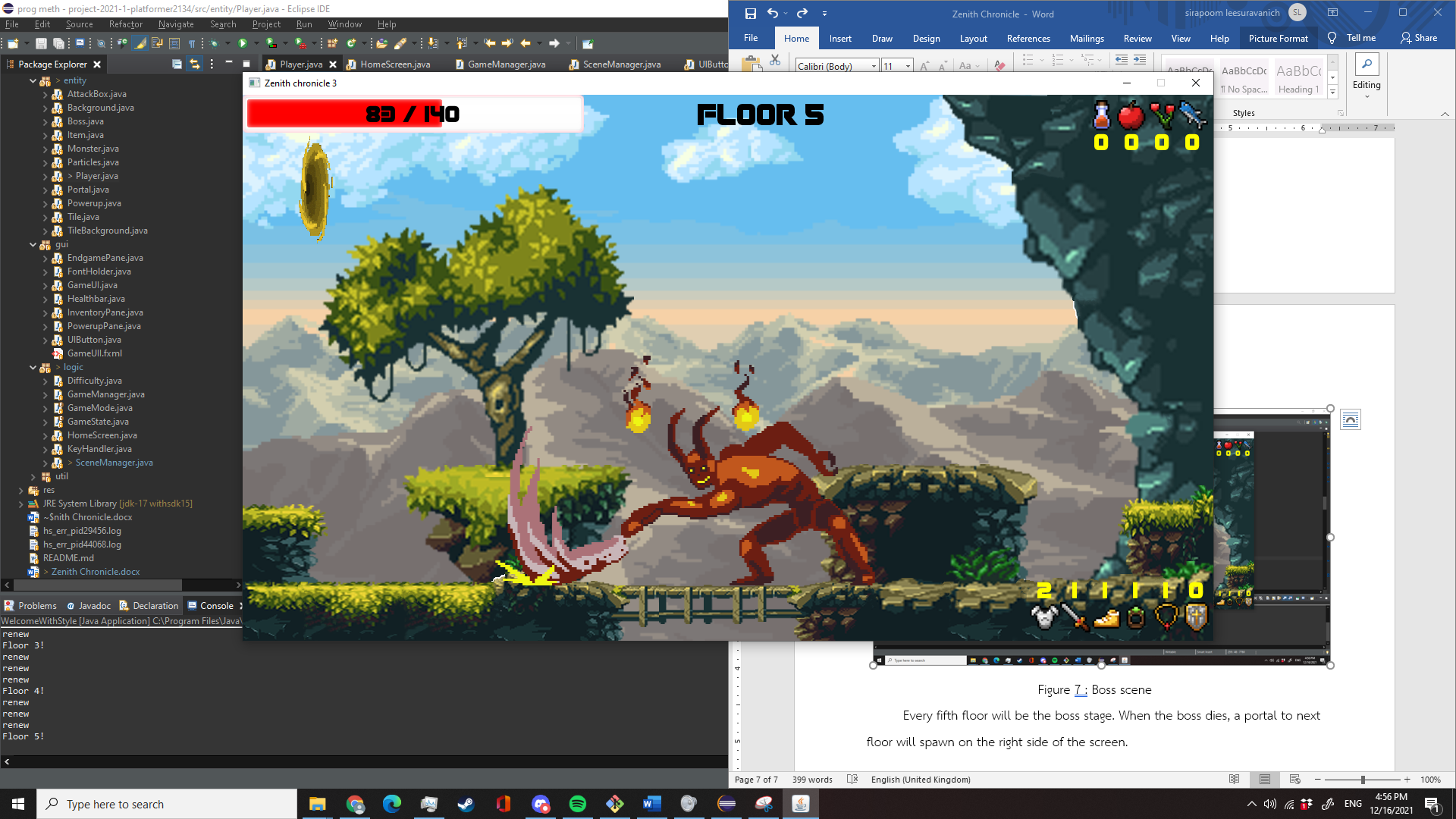
**Boss scene**

Figure 13 : Boss scene

Every fifth floor will be the boss stage. When the boss dies, a portal to next floor will spawn on the right side of the screen.

**Victory scene**

Figure 13 : Victory scene

When the player had collected all types of artifacts, they win the game. They have four options, Continue : Continue playing until dies, Restart : Start over from the beginning, Main menu : Go back to the main menu, Quit : Quit the game.

**Death scene**

Figure 14 : Death scene

After win the game, If player chose to continue fighting and die, they will see this scene showing how long they have managed to survived throughout the game. The player will have 3 options as same as Victory scene but without Continue button.

**Lose scene**

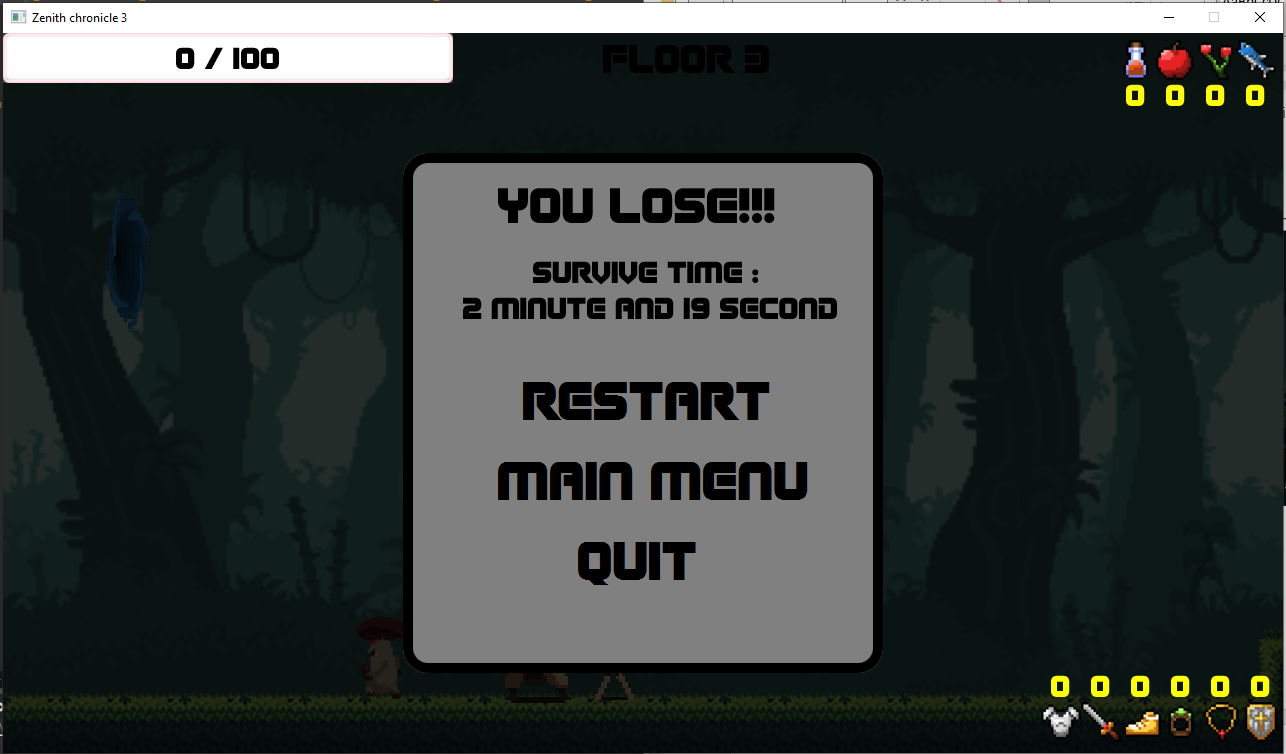


Figure 15 : Lose scene

If the player die without collecting all types of the artifacts. They will see this scene and will have the same options as in Death scene.

(uml here)

..

.

.

.

.

.

(doc here)

**1. Package component**

**1.1 class Sprite**

Contains image data for an in-game sprite.

1.1.1 Constructors

|  |  |
| --- | --- |
| + Sprite(string filepath) | Initialize these fields:  Set this.filepath as the argument filepath  Set image using new Image(ClassLoader.getSystemResource(  this.filepath).toString()) |

1.1.2 Fields

|  |  |
| --- | --- |
| -Image image | Image object contains sprite’s image |
| -String filepath | The image’s file path |
|  |  |

1.1.3 Methods

|  |  |
| --- | --- |
| +void loadImage() | Load image from filepath to image |
| Getters for each fields |  |

**2 Package entity.base**

**2.1 abstract class Entity**  
 A base structure for all entities that will exist in the game.

2.1.1 Constructors

|  |  |
| --- | --- |
| +void Entity(double x, double y, int, w, int h) | * Initialize x, y, w, h with the unput arguments * Set radius with w/2 |
| +void Entity(double x, double y, int r) | * Initialize x, y, radius with the input arguments * Set w and h with 2\*r |

2.1.2 Fields

|  |  |
| --- | --- |
| -double x | Coordinate x of the entity |
| -double y | Coordinate x of the entity |
| -int w | Entity’s width |
| -int h | Entity’s height |
| -int radius | Entity’s radius from center |
| #prevx | Coordinate x in the last frame |
| #prevy | Coordinate y in the last frame |

2.1.3 Methods

|  |  |
| --- | --- |
| *+void update()* | Update entity states and attributes each frame. |
| *+Sprite getImage()* | Returns Sprite object member of entity.  Note that an entity can have multiple sprites. |
| + draw(GraphicsContext gc, Image img, double x, double y, int w, int h) | Use graphics context to draw img on screen at coordinates (x, y) with w width and h height |
| + draw(GraphicsContext gc, boolean f) | * Call the original draw method with different arguments that depends on f argument * Will draw flipped image if f is true |
| +boolean collideWith(Entity other) | Check if with this entity and other collides |
| +void increaseX(double x) | Increase x value. Note that if x is negative then this will decrease x value. |
| +void increaseY(double y) | Increase y value |
| Getters and Setters for each field |  |

**2.2 abstract class FallObject extends Entity**

Objects of this class will fall ( increase in coordinate y ) every time fall() is called. Will stop falling once reached ground.

2.2.1 Constructors

|  |  |
| --- | --- |
| +void FallObject(double x,double y, int w, int h) | Calls super(x, y, w, h) |

2.2.2 Fields

|  |  |
| --- | --- |
| -double Vy | Determines how much y will change everytime fall() is called. |

2.2.3 Methods

|  |  |
| --- | --- |
| #int fall() | * Increases y value with Vy * Checks if entity clips through ground after increased y. If so, sets the entity on ground. * Returns 1 if entity falls downwards * Returns -1 if entity floats (y decreased) * Otherwise, returns 0 |
| Getters and Setters for each field |  |

**2.3 abstract class Character extends FallObject implements Collidable**

Class for entites that can move on terrain and can do/take damage.  
2.3.1 Constructors

|  |  |
| --- | --- |
| +void Character(double x, double y, int w, int h) | * Calls super(x, y, w, h) * Initialize these fields : * Set justTakeDamage = 0 * Set alive = true |

2.3.2 Fields

|  |  |
| --- | --- |
| #int hp | Health points of the character |
| #int maxHp | Max HP of the character |
| #int atk | Attack vale |
| #boolean alive | True if this character is alive |
| #double movespeed | Character’s movespeed (how much x will change if moveLeft() or moveRight is called. |
| #int justTakeDamage | Amount of frames left until can take damage again. |
| #boolean needRemove | True if this character needs to be remove or do something on the next update |

2.3.2 Methods

|  |  |
| --- | --- |
| +void update() | * Decrease justTakeDamage by 1 * Calls fall() |
| #void moveRight() | Increase x by movespeed and check if character clips through wall, if so, do not increase x. |
| #void moveLeft() | Same as moveRight() but decreases x |
| +void changeMaxHp(int hp) | Increase maxHp+=hp |
| +void takedamage(int atk) | Decrease hp and if hp < 0, set alive to false and call die() |
| +void changeAtk(int atk) | Increase this.atk+=atk |
| +void changeHp(int hp) | * this.hp += hp * hp can not go below zero |
| *#void die()* | Do something when a character dies |
| Getters and Setters for each field |  |

**2.4 abstract class Enemy extends Character**

Class for enemies.

2.4.1 Constructors

|  |  |
| --- | --- |
| +void Enemy(double x, double y, int w, int h) | Call super(x, y, w, h) |

2.4.3 Methods

|  |  |
| --- | --- |
| *+void checkCollide()* | Check collisions |
| +void takeDamage(int x) | * Call super.takeDamage(x) * Increase justTakeDamage by 30 |
| #void die() | Wait 0.4 sec and set needRemove to true. |

**2.5 interface Collidable**

Defines methods for Entity that collide and do something  
2.5.1 Methods

|  |  |
| --- | --- |
| *void checkCollide()* | Do something on collision |

**2.6 interface Interactable**

Defines methods for Entity that can be interacted by the player

2.6.1 Methods

|  |  |
| --- | --- |
| *void checkInteract()* | Do something on interaction |

**3. Package Entity**

**3.1 class Player extends Character**

Class for the player, the main character.

3.1.1 Constructors

|  |  |
| --- | --- |
| +void Player() | * Calls super(150 ,550, 120, 120)   (set x, y at 150, 550 and set width and height to 120x120   * Initialize these fields: * lastFrameStatus = IDLE * status = IDLE * jumpStatus = ONGROUND * face = RIGHT * direction = 0 * prevGround = 550+120 * maxDash = 1 * dashAvail = maxDash * initialize inventory with new ArrayList<Integer>(Arrays.asList(0,0,0,0,0,0)); * set alive = true * maxHp =100 * hp = maxhp * atk = 70 * moveSpeed = 7 * initJumpSpeed = 10 * dashSpeedMultiplier = 11/7 * load sprites for each action :   idle, run, jump, fall, dash, hurt, death   * start a timer thread that calls heal() every second |

3.1.2 Fields

|  |  |
| --- | --- |
| -int attackable | Frames left until can attack again |
| -int immune | Frames left until can take damage again |
| -PlayerStatus face | Direction player’s facing |
| -PlayerStatus status | Current status of player |
| -PlayerStatus lastFrameStatus | Status of player in the last frame |
| -int direction | - player’s moving direction  - -1 if moving left  - 0 if standing still  - 1 if moving right |
| -double moveSpeed | Player’s moveSpeed (in pixel) |
| -int dashing | Frames until can dash again |
| -double dashSpeed | Player’s dash length |
| -double dashSpeedMultiplier | Multiplier for moveSpeed to make dashSpeed |
| -int maxDash | Max dash player can perform before touching ground |
| -int dashAvail | Amount of dashes available left |
| -PlayerStatus jumpStatus | GOINGUP or FALLING or ONGROUND |
| -double initJumpSpeed | Initial vertical speed when player performs jump |
| -double prevGround | Last ground that player is on before going airborne (y value) |
| -Sprite idle,run,jump\_up,jump\_down,death,  attack, hurt, roll | Sprite object for each action |
| +ArrayList<integer> inventory | Player’s inventory store amount of each artifact type that player has |
| -int healing | Value that increase every second, if reaches threshold, player heals and the value reset to 0. Scale with an artifact. |
|  |  |

3.1.2 Methods

|  |  |
| --- | --- |
| +void update() | * Update player’s movement and state each frame depending on player’s input from KeyHandler and animation cooldown. * Check for collision with enemy and takeDamage(enemy’s attack) * fall() |
| #int fall() | Overrides FallObject’s fall() but add check clipping with upper Tile (collides with roof when jumping) |
| -void jump() | Set Vy to initJumpSpeed |
| -void moveRight() | - calls super.moveRight(moveSpeed)  -set offsetX in SceneManager to draw frame according to player’s coordinates |
| -void moveLeft | Same with moveRight but goes in opposite direction |
| -void dash() | moveLeft or Right with dashSpeed |
| +Sprite getImage() | Get Sprite object depending on status |
| +void draw(GraphicsContext gc, Boolean f) | Draws player with super.draw with arguments depending on status |
| -void attack() | Creates an AttackBox and check if it collides with an enemy, if so, enemy.takeDamage(atk) |
| +void takeDamage() | Take damage |
| +void heal() | Healing += 5 x amount of artifact Shield of Light |
| +void changeJumpH(int h) | Increase initJumpSpeed (which also increase jump height) |
| +changemvsp(int sp) | Increase moveSpeed |
| +void changeMaxDash(int x) | Increase maxDash |
| #protected void die() | Wait 10 seconds and set needRemove = true; |
| +boolean isimmune() | Check if player is immune to damage |
| Getters and Setters for each field |  |

**3.2 class Monster extends Character**

3.2.1 Constructors

|  |  |
| --- | --- |
| +Monster(double x,double y,int type) | Initialize each field and set type of this monster |
| +public Monster(double x,int type) | Initialize each field and set type of this monster |

3.2.1 Fields

|  |  |
| --- | --- |
| -ArrayList<Sprite> idle, run death, hurt | List of Sprite for each action of every monster type |
| -ArrayList<Integer> size | List of size for each monster type |
| -ArrayList<AudioClip> audio | List for audio effect for ech monster type |
| -int type | Monster type id |
| -int randomMove | Frames left before decide random movement again |
| -int direction | Monster’s direction 0 left, 1 right |
| -Random r | Random variable |

3.2.2 Methods

|  |  |
| --- | --- |
| +void update() | * Follows player if player is in range else do a random movement * fall() |
| +void takeDamage() | Take damage |
| +Sprite getImage | Get correct sprite depending on action |
| +void setUp() | Set up data of monsters (sprites, size, audio) |
| +void generateRandom() | Generate random monster at random x |
| +void generate(double x) | Generate random monster at x |
| +void generate(double x, double y) | Generate random monster at x, y |
| +void generate(int lowerBound, int upperBound) | Generate random monster at given x area at random y |
| +void generate(int lowerBound, int upperBound, double y) | Generate random monster at given x area and y |
| +void draw(GraphicsContext gc, boolean f) | super.draw(gc, direction==0)  draw monster on screen |
| Getters and Setters for each field |  |

**3.3 class Artifact extends FallObject implements Interactable, Collidable**

Items that can be picked up

3.3.1 Constructors

|  |  |
| --- | --- |
| +Artifact(double x,int type) | Call super constructor at x, set type |
| +Artifact(double x,double y, int type) | Call super constructor at x, y , set type |

3.3.1 Fields

|  |  |
| --- | --- |
| -ArrayList<Sprite> sprites | List of artifact sprites |
| -int type | Type id |
| -Random r | Random variable |
| -Audioclip dropSound, equipSound | sfx |

3.3.2 Methods

|  |  |
| --- | --- |
| +void update() | * fall() * checkInteract() |
| +Sprite getImage() | Get sprite |
| +void setup | Setup Sprites |
| +void generate(double x) | Generate random artifact at x |
| +void generate(double x, double y) | Generate random artifact at x, y |
| +void generate(int lowerBound, int upperBound) | Generate random artifact at given x area at random y |
| +void generate(int type, int lowerBound, int upperBound) | Generate random given type artifact at given x area |
| +void checkInteract() | Check if player interacts and add this to player’s inventory and grants effects |
| Getters and Setters for each field |  |

**3.4 class Powerup extends FallObject implements Collidable**  
3.4.1 Constructors

|  |  |
| --- | --- |
| +Powerup (double x,double y,int type) | Initialize fields  Set level = 0 |
| +Powerup(double x,double y,int type,boolean renewable,int level) | Initialize fields  Set level = 0 |
| +Powerup(double x,int type) | Initialize fields  Set level = 0 |
| + Powerup(double x,double y,int type,boolean renewable,int level) | Initialize fields |

3.4.2 fields

|  |  |
| --- | --- |
| -int type | Type id |
| -boolean renewable | If this itemcan respawm |
| -ArrayList<Sprite> sprites | List of powerups sprites |
| - Queue<Integer> renew | Queue of powerups to be renew next update |
| -ArrayList<Thread> thread | Threads of ongoing powerup effects |
| -boolean interrupt | If interrupt then don’t renew |

3.4.3 methods

|  |  |
| --- | --- |
| +Sprite getImage() | Get sprite |
| +void setUp() | Set up sprites |
| +void generate(int lowerBound, int upperBound) | Generate random powerup at given x area at random y |
| +void generate(int type, int lowerBound, int upperBound) | Generate random given type powerup at given x area |
| +void generate(x) | Generate random powerup at given x |
| +void generate(); | Generate powerups in level at exact positions |
| +void checkCollide | Check if collides with player and grant effects then start thread to count down effects cooldown |
| +void renewPowerUp() | Renew a powerup from queue |
| +void clear() | Clear threads |

**3.6 class Tile extends Entity**

Terrain object.

3.6.1 Constructor

|  |  |
| --- | --- |
| + Tile(double x, double y, int w, int h, boolean t) | Initialize fields |

3.6.2 Fields

|  |  |
| --- | --- |
| -boolean transparent | True if this tile is transparent (can jump through and not consider as ceiling) |
| -double upperBound | Upper bound of tile |
| -double lowerBound | Lower bound of tile |
| -double rightBound | Right Bound of tile |
| -double leftBound | Left bound of tile |

3.6.3 Methods

|  |  |
| --- | --- |
| +void generate() | Generate tiles in the level |
| +void generateBossArena() | Genrat tiles in boss arena |

**3.7 class Background extends Entity**  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2 ...  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2 ...  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2 ...  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2 ...  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2 ...  
1.2.1 Fields

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1.2.2 Methods

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |