Archery Documentation

Created by

Papon Kleubmongcol 6431326321
Natdanai Trintawat 6431316021

2110215 Programming Methodology
Semester 2 year 2021
Chulalongkorn University

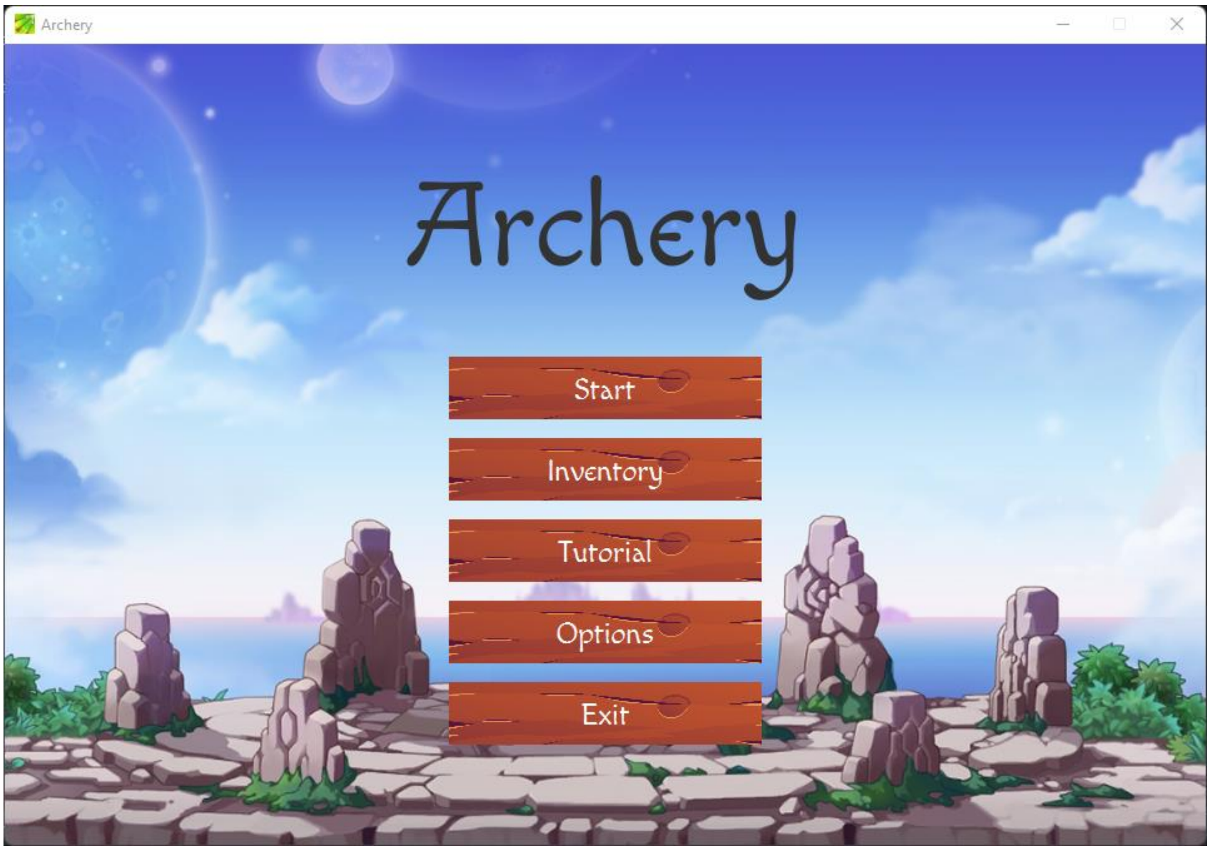
Introduction

Archero is a 2D Role-playing game (RPG). This game is inspired by the mobile game “Archero”. The objective of this game is to clear a stage by killing all monsters and getting equipment as a reward and so on.

How to play

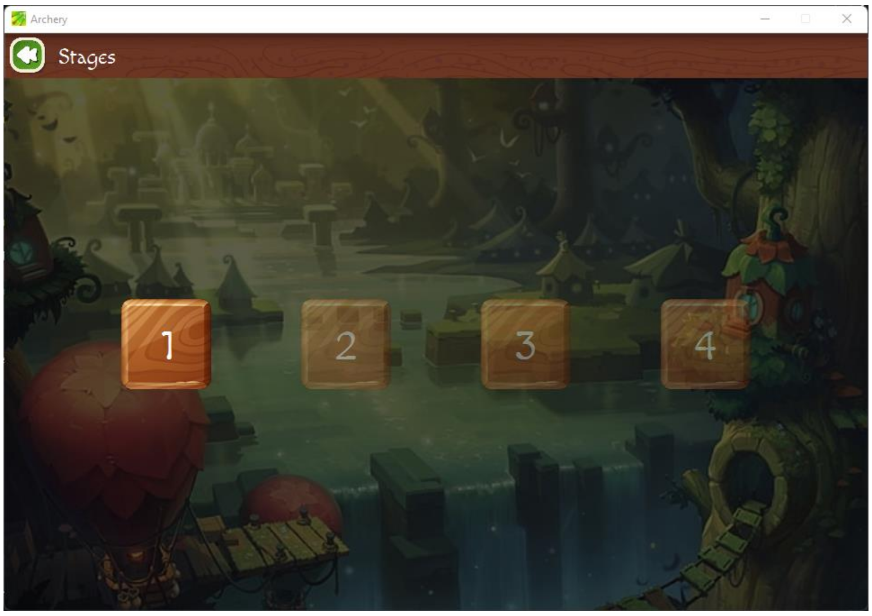
When you open the game. You will see the main menu screen, There are 5 buttons on this screen

-Start: Open stages selection screen
-Inventory: Select your equipment
-Tutorial
-Option: Adjust the sound volume
-Exit



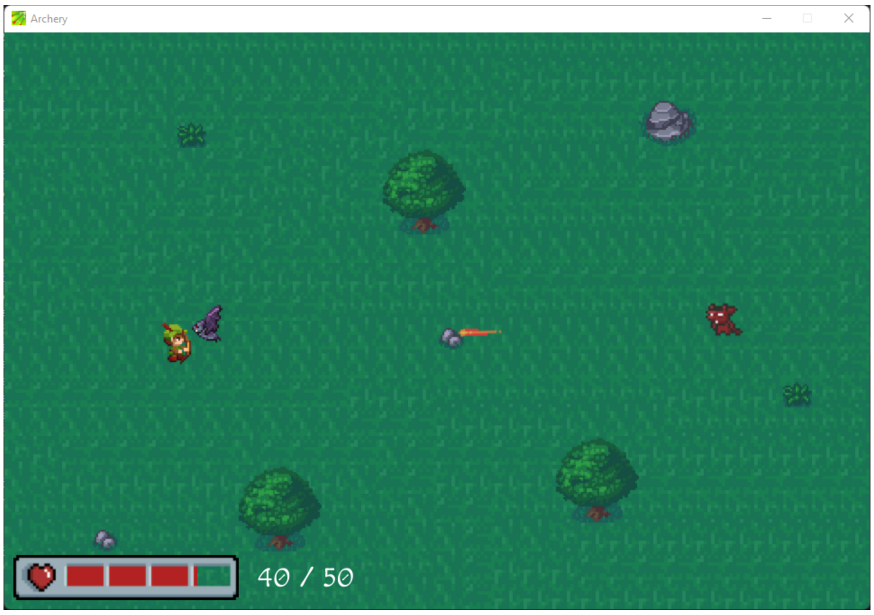
Main Screen

To enter the stage, Click on “Start”. You will see the stage selection screen. At first, You can access only the first stage. When you clear the stage, the next stage will be unlocked.



Stages select screen

After clicking on the stage button, you will enter the field. Each stage has 2 fields. Your task is to kill all monsters in the first field, go to the next field and kill all monsters again.

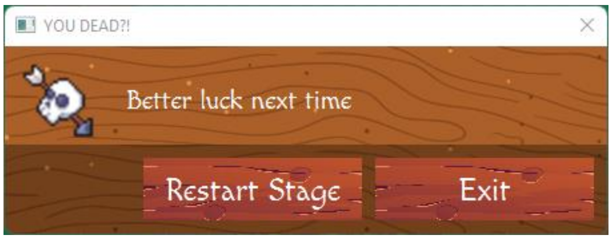


Field example

To control your hero, use W,A,S, and D keys to control walking direction like a general games(W for up, S for down, D for forward, and A for Backward)

To attack a monster, your hero will automatically shoot the nearest monster (sometimes he might attack an obstruction located between your hero and target monster), but he will stop shooting while walking. So, your strategy is walking around to avoid monster attacks, but you also have to stop for a short time to shoot a monster.

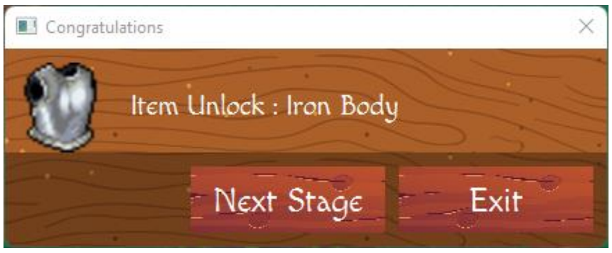
If you get attacked by a monster, your health will decrease, and if your health reaches 0 points, you will die.


YOU DEAD?!

Better luck next time

Pop-up shows when you died.

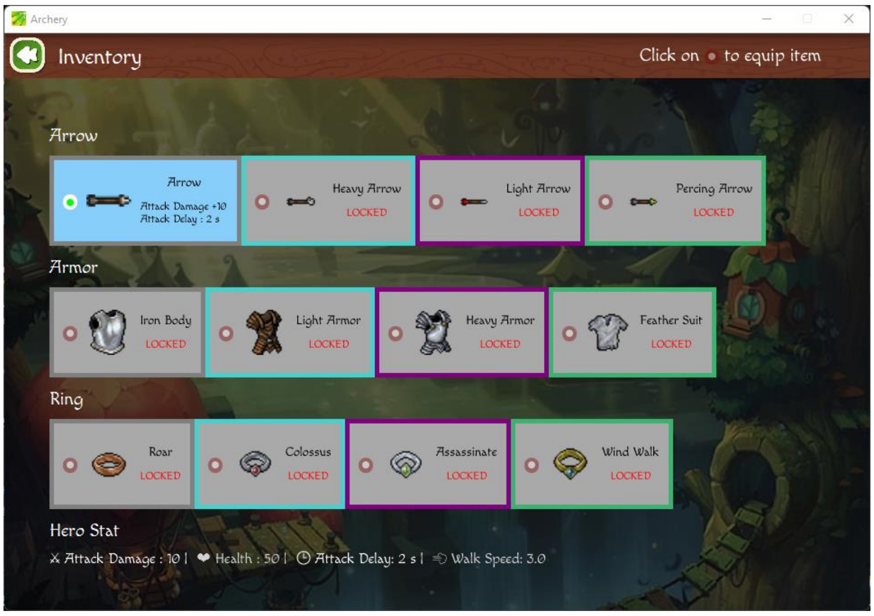
If you can clear the stage. You “might” get equipment as a reward.


Congratulations

Item Unlock : Iron Body

Example pop-up shows when you clear a stage.

After you get new equipment. You can equip them by clicking on “Inventory” in the main menu.



Arrow

Arrow Attack Damage +10 Attack Delay : 2 s | Heavy Arrow LOCKED | Light Arrow LOCKED | Percing Arrow LOCKED

Armor

Iron Body LOCKED | Light Armor LOCKED | Heavy Armor LOCKED | Feather Suit LOCKED

Ring

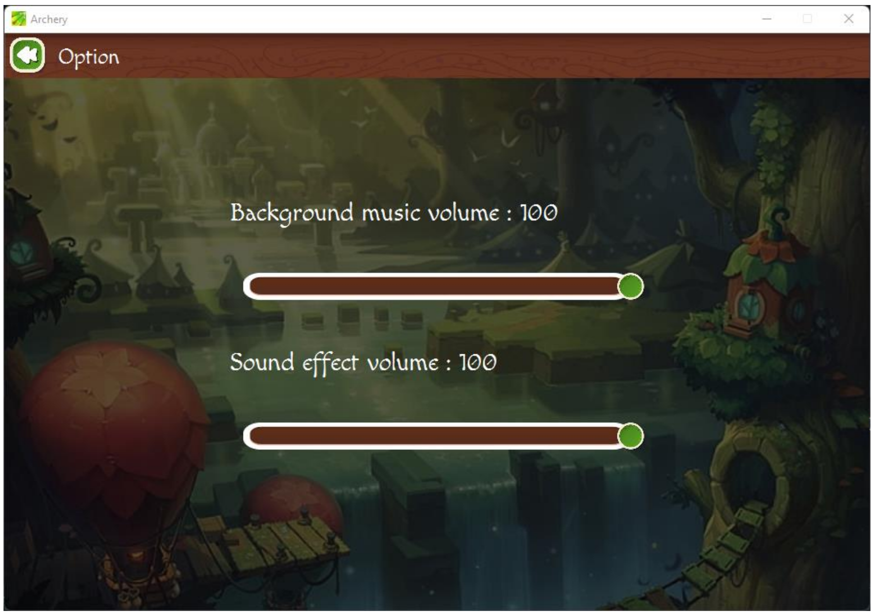
Roar LOCKED | Colossus LOCKED | Assassinate LOCKED | Wind Walk LOCKED

Hero Stat

⚔ Attack Damage : 10 | ❤ Health : 50 | ⏱ Attack Delay: 2 s | Walk Speed: 3.0

Inventory screen

You can adjust the background sound and effect sound volume by clicking on “Option” in the main menu.

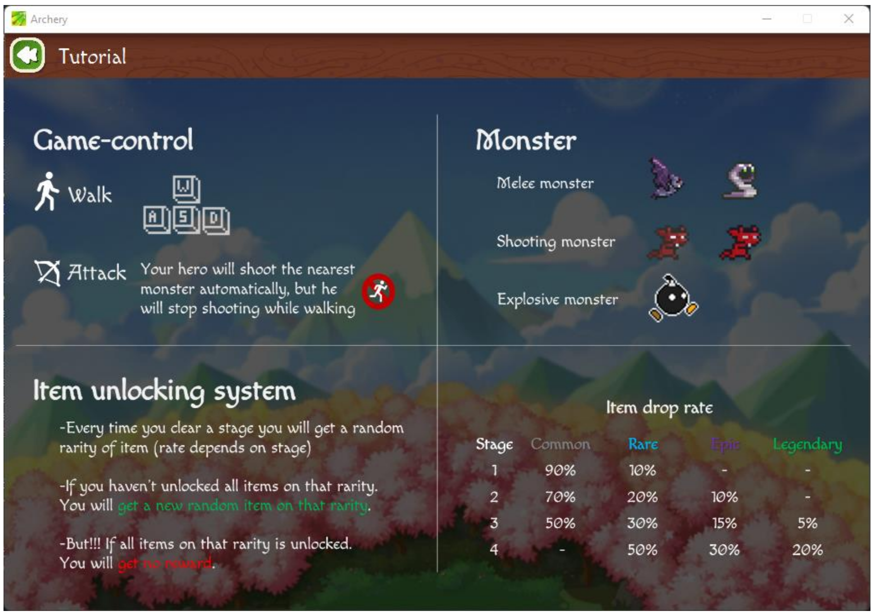


Background music volume : 100

Sound effect volume : 100

Option Screen

You can read the tutorial and rewarding system by clicking on “Tutorial” in the main menu



Your hero will shoot the nearest monster automatically, but he will stop shooting while walking

Item unlocking system

-Every time you clear a stage you will get a random rarity of item (rate depends on stage)

-If you haven't unlocked all items on that rarity. You will get a new random item on that rarity.

-But!!! If all items on that rarity is unlocked. You will get no reward.

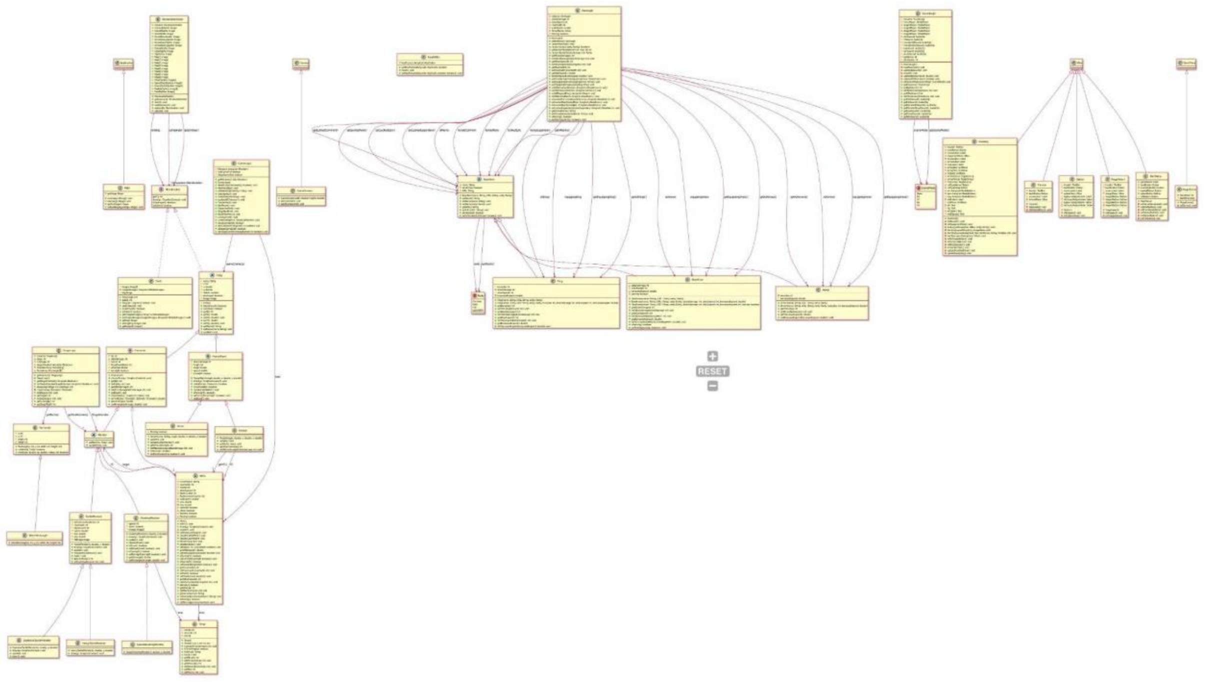
Item drop rate

Stage	Common	Rare	Epic	Legendary
1	90%	10%	-	-
2	70%	20%	10%	-
3	50%	30%	15%	5%
4	-	50%	30%	20%

Tutorial Screen

Class diagram

*Full Resolution on GitHub



1. Package items

This package consists of classes that represent all item that show and upgrade hero status.

1.1 Enum Rarity

This enumeration is for separated rarity of items which affect their color and reward rate.

{Common, Rare, Epic, Legendary}

1.2 Abstract class BaseItem

This class is a base class for all items in the game.

1.2.1 Fields

-String name	Name of the item.
-boolean isUnlocked	True if the item is unlocked.

#Rarity rarity	Rarity of the item.
#String iURL	URL of the image of the item.

1.2.2 Constructor

+BaseItem(String name, String iURL, Rarity rarity)	Initialize an item.

1.2.3 Methods

+Getters and setters for all fields

1.3 Class BowArrow extends Item

This class represents the item arrow which is the main weapon of your hero.

1.3.1 Fields

-int attackdamage	Attack damage per hit of arrow.
-int attackspeed	Attack delay between shot (seconds).
-int bonuswalkspeed	Walk speed multiplier.
-boolean percing	If true, an arrow will hit through all monsters.

1.3.2 Constructors

+BowArrow(String name, String iURL, Rarity rarity)	Initialize BowArrow with the value of parameters and with
	attackdamage = 10
	attackspeed = 2
	bonuswalkspeed = 0
	percing = false.

+BowArrow(String name, String iURL, Rarity rarity,int attackdamage,int	Initialize BowArrow with the value of parameters and with percing = false.

attackspeed,double bonuswalkspeed)	
+BowArrow(String name, String iURL, Rarity rarity,int attackdamage,int attackspeed,int bonuswalkspeed,boolean percing)	Initialize with the value of parameters.
+Getters and setters for all fields.	

1.4 Class Armor extends Item

This class represents the item armor which you can wear to upgrade your hero status.

1.4.1 Fields

-int bonushp	Bonus health point.
-double bonuswalkspeed	Walk speed Multiplier.

1.4.2 Constructors

+Armor(String name, String iURL, Rarity rarity)	Initialize armor with the value of parameters and with bonushp = 0 bonuswalkspeed= 0.
+Ring(String name, String iURL, Rarity rarity,int bonushp,int attackdamage,int attackspeed,double bonuswalkspeed)	Initialize armor with the value of parameters.

1.4.3 Methods

+Getters and setters for all fields.

1.5 Class Ring extends Item

This class represents the item ring which you can wear to upgrade your hero status.

1.5.1 Fields

-int bonushp	Bonus health point.
-int attackdamage	Bonus Attack Damage.
-int attackspeed	-NOT USE ANYMORE-
-double bonuswalkspeed	Walk speed multiplier.

1.5.2 Constructors

+Ring(String name, String iURL, Rarity rarity)	Initialize Ring with the value of parameters and with attackdamage = 0
	attackspeed = 0
	bonuswalkspeed = 0.

Ring(String name, String iURL, Rarity rarity,int bonushp,int attackdamage,int attackspeed,double bonuswalkspeed)	Initialize Ring with the value of parameters.

1.5.3 Methods

+Getters and setters for all fields.

2. Package sharedObject

This package consists of classes that contain status and ability to render objects.

2.1 Interface IRenderable

This interface indicates the ability to be able to draw on the canvas.

2.1.1 Methods

+int getZ()	get Z value.
+void draw(GraphicsContext gc)	This method will be called when drawing the Entity.
+boolean isDestroyed()	Self-explanatory.
+boolean isVisible()	Self-explanatory.

2.2 Class RenderableHolder

This class holds render objects and sprites.

2.2.1 Fields

-List<IRenderable> entities	List that contain IRenderable objects.
-Comparator<IRenderable> comparator	A comparator function that use to sort the IRenderable objects.
-final RenderableHolder instance	Instances of RenderableHolder.
+Image herowalkSprite	Hero walking images.
+Image herostillSprite	Hero stand still/ aiming images.
+Image tackleSprite	Tackle monster images.
+Image ArrowBasicSprite	Basic arrow image.
+Image ArrowHeavySprite	Heavy arrow image.
+Image ArrowQuickSprite	Quick arrow image.
+Image ArrowPercingSprite	Piercing arrow image.
+Image FireballSprite	Fireball image.
+Image HpbarSprite	Hp bar image.
+Image SignSprite	Arrowsign image.

+<u>Image Map11</u>	Map 1-1 image.
+<u>Image Map12</u>	Map 1-2 image.
+<u>Image Map21</u>	Map 2-1 image.
+<u>Image Map22</u>	Map 2-2 image.
+<u>Image Map31</u>	Map 3-1 image.
+<u>Image Map32</u>	Map 3-2 image.
+<u>Image Map41</u>	Map 4-1 image.
+<u>Image Map42</u>	Map 4-2 image.
+<u>Image[] ShootSprites</u>	Array of all shooting monsters images in game.
+<u>Image[] SpeedShootSprites</u>	Array of all speed shooting monsters images in game.
+<u>Image[] HeavyTackleSprites</u>	Array of all heavy tackle monsters images in game.
+<u>Image[] ExplodeSprites</u>	Array of all explode/bomb monsters images in game.
+<u>Image[] DeadSprites</u>	Array of all Dead particle images in game.

2.2.2 Constructor

+RenderableHolder()	Initialize RenderableHolder create new array for entities and set the comparator to sort the IRenderable array.

2.2.3 Methods

-<u>void loadResource()</u>	Load all images from the resource file. This function will run only once when initializing a new RenderableHolder.

+void add(IRenderable entity)	Add the entity parameter to this RenderableHolder entities and sort with a comparator function.
+void update()	Update all IRenderable object in this RenderableHolder entities if this object is destroyed and invisible will remove this object from the entities array.
+reset()	Create a new array for this RenderableHolder entities.
+List<IRenderable> getEntities()	Getter for entities.
+RenderableHolder getInstance()	Getter for instance.

3. Package entity

This package consists of classes that represent stage character and stage scene.

3.1 Abstract Class Entity implements IRenderable

This class is a base class for all entities.

3.1.1 Fields

-String name	Name this entity.
#double x	Position x.
#double y	Position y.
#int z	Position z (for sorted array).
#boolean visible	Self-explanatory.

#boolean destroyed	Self-explanatory.
#Image Image	Image this entity.

3.1.2 Constructor

#Entity()	Initialize entity
	visible = true
	destroyed = false.

3.1.3 Methods

+*void update()*	Use to update entity
+void setX(double x)	Set this.x to this entity x If x < 20, set this entity x to 20. If x > 940, set this entity x to 940.
+void setY(double y)	Set this.y to this entity y If y < 20, set this entity y to 20 If x > 608, set this entity y to 608.
+Getters and setters for all fields.	

3.2 Abstract Class Character extends Entity

This class is a base class for other Characters that can move and interact.

3.2.1 Fields

#int hp	Self-explanatory.
#int attackdamage	Self-explanatory.
#int radius	Set to 32.
#int DeadCountDown	Set to 0.

#double aimangle	Angle used for aiming targets.
#boolean turnright	Turnright or left.

3.2.2 Constructor

+Character()	Initialize Character. Set z = 1.

3.2.3 Methods

+void chooseTurn(GraphicsContext gc)	Draw Image this canvas. The Image will look left or right depending on this character turnright.
+void drawDead()	Draw images of dead particle depend on DeadCountDown field and if DeadCountDown is more than 15, set this character visible to false.
+double aim(Character attacker, Character defender)	Return angle(radians) between attacker and defender.
+void setHp(int hp)	If hp less than 0, set this character hp to 0.
+void setDead()	Update if this character hp less than 0. Then, set destroyed to true.
+Getters and setters for all fields.	

3.3 Class Hero extends Character

This class represents a hero or player character that the player can interact with.

3.3.1 Fields

-String ArrowName	Self-explanatory.

-int countwalk	Count walking step
-int maxhp	Self-explanatory.
-int attackspeed	Self-explanatory.
-int flashDurationCounter = 0	Duration of flashing Set to 0.
-int flashCounter	How many time to flashing. Set to 0.
-double walkspeed	Self-explanatory.
-double nex	Next x positon.
-double ney	Next y position.
-boolean standstill	Self-explanatory.
-boolean shoot	Self-explanatory.
-boolean flashing	Self-explanatory.
-boolean percing	if true, an arrow will hit through all monsters.
-Monster target	Target to aim.
-Timer time	Time count.

3.3.2 Constructor

+Hero()	initialize Hero. set z = 10, set name to “Hero”, and initXY().

3.3.3 Methods

+void initXY()	Set position x to 160. Set position y to 320. Set nex to this x position. Set ney to this y position.
+void draw()	Draw images of this Hero and draw a hp bar in the left bottom of this canvas. The

	image that use to draw will depend on this hero standstill and time.
+double aim(Character attacker, Character defender)	Return angle(radians) between attacker and defender.
+void update()	If hp less than 0, set this hero destroyed to true. Update this hero visible,countwalk,positionX&Y ,shoot and target and increaseTimer.
+void setShootandTarget()	If some monster is alive, set this hero target to the nearest monster, and if time second equal attackspeed hero, and if this hero is standstill, set shoot to true and reset time. if all monsters are dead set shoot to false.
+void UpdateCollideRect()	If position nex and ney collide to Rectangle object, set nex to x and set ney to y
+void UpdateCountWalk()	If this hero standstill, set countwalk to 0
+void MoveCharacter()	Move the character by keypress. -press W will move hero up -press S will move hero down -press A will move hero left -press D will move hero right Set standstill to false and +1 countwalk if key has pressed.
+void UpdateVisible()	Set visible to false if flashing and run loop flashCounter with the flashDurationCounter.
+void setHp(int hp,boolean	If this hero are not flashing

CannotWait)	or CannotWait, set this hero flashing to true, set flashCounter to 8, set flashDurationCounter to 8, and set this hero hp to hp.
+void setCountwalk(int countwalk)	Set this hero countwalk to countwalk mod 60.
+void setMaxhp(int maxhp)	Set this hero maxhp to maxhp . Set this hero hp to maxhp.
+Getters and setters for all fields.	

3.4 Abstract Class Monster extends Character

This class is a base class for all monsters in the game.

3.4.1 Fields

#Hero h1	Hero or target of this Monster.

3.4.2 Constructor

+Monster()	Do nothing.

3.4.3 Methods

+void setHero(Hero h1)	Set this monster h1 to h1.
+void UpdateAim()	Set this aimangle. Then, if cos(aimangle) < 0, set this monster turnright to false. Otherwise, set this monster turnright to true.

3.5 Class TackleMonster extends Monster

This class is used to draw and update the status of Tackle Monster.

3.5.1 Fields

#int refreshcountwalkrate	Mod for countwalk.
#int countwalk	Count walking step.
#int attackcount	Count attack when collide this tackle monster h1.
#double speed	Self-explanatory.
#double nex	Next x positon.
#double ney	Next y positon.
#Image initImage	This tackle monster image.

3.5.2 Constructor

+TackleMonster(double x,double y)	Initialize TackleMonster. Set this monster x to x. Set this monster y to y. Set this monster nex to x. Set this monster ney to y. Set this monster speed to 1.2 Set this monster. Refreshcountwalkrate to 75. set this monster attackdamage to 10. Set this monster hp to 15. Set this monster initImage to RenderableHolder.tackleSprite

3.5.3 Methods

+void draw(GraphicsContext gc)	Choose lookside on this image and draw cropped image in this monster. initImage depends on countwalk
+void update()	Update this monster

	destroyed, visible, angleaim, x y nex ney position, attackcound and countwalk.
+void AttackandCooldown()	If this monster collide hero/h1 and attackcount equal 100, decrease hero/h1 hp and +1 attackcount.
+void walk()	Set this monster nex,ney position moving to hero/h1 depend on this monster speed +1 countwalk.
+void setCountwalk(int count)	Set this monster countwalk to countwalk mod this monster refreshcountwalkrate.
+Getters and setters for all fields.	

3.6 Class HeavyTackleMonster extends TackleMonster

This class is used to draw and update the status of Heavy Tackle Monster.

3.6.1 Constructor

+HeavyTackleMonster(double x,double y)	Initialize HeavyTackleMonster. Set this monster x to x. Set this monster y to y. Set this monster nex to x. Set this monster ney to y. Set this monster speed to 1 Set this monster. Refreshcountwalkrate to 100. Set this monster attackdamage to 20. Set this monster radius to 16.

	Set this monster hp to 25.

3.6.2 Method

+void draw(GraphicsContext gc)	Choose lookside on this image and draw images in RenderableHolder.HeavyTackleSprites depends on countwalk.

3.7 Class ExplosiveTackleMonster extends TackleMonster

This class is used to draw and update the status of Explosive Tackle Monster.

3.7.1 Constructor

+HeavyTackleMonster(double x,double y)	Initialize ExplosiveMonster. Set this monster x to x. Set this monster y to y. Set this monster nex to x. Set this monster ney to y. Set this monster speed to 2.1. Set this monster attackdamage to 20. Set this monster radius to 16. Set this monster hp to 10. Set this refreshcountwalkrate to 15.

3.7.2 Methods

+void update()	update this monster destroy,visible,x,y,nex,ney,aimangle and if monster collide to this h1/hero decrease h1/hero hp
+void draw(GraphicsContext	choose lookside on this image

gc)	and draw images in RenderableHolder.ExplodeSprites depends on countwalk
+void boom()	set this monster hp to 0 and decrease h1/hero hp depend on this attackdamage

3.8 Class ShootingMonster extends Monster

This class is used to draw and update the status of Shooting Monster.

3.8.1 Fields

#int speed	this shooting monster attack speed
#boolean shoot	Self-explanatory.
#Timer time	time shooting count
#Image[] images	array of this shooting monster images

3.8.2 Constructor

+TackleMonster(double x,double y)	Initialize ShootingMonster. Set this monster x to x. Set this monster y to y. Set this monster nex to x. Set this monster ney to y. Set this monster attackdamage to 4. Set this speed to 5. Set this shoot to false. Set this radius to 22. Set this monster hp to 20. Set this monster images to RenderableHolder.ShootSprites.

3.8.3 Methods

+void draw(GraphicsContext gc)	Choose lookside on this image and draw image in canvas from this shooting monster images depends on countwalk
+void update()	if time seconds > 0 and time seconds equal this monster speed reset time and set shoot to true.
+void UpdateShoot()	If this monster collide hero/h1 and attackcount equal 100, decrease hero/h1 hp and +1 attackcount.
+Getters and setters for all fields.	

3.9 Class SpeedShootingMonster extends ShootingMonster

This class is used to draw and update the status of Speed Shooting Monster.

3.9.1 Constructor

+TackleMonster(double x,double y)	Initialize SpeedShootingMonster. Set this monster x to x. Set this monster y to y. Set this monster nex to x. Set this monster ney to y. Set this monster attackdamage to 4. Set this speed to 2. Set this shoot to false. Set this radius to 22. Set this monster hp to 20. Set this monster images to RenderableHolder.SpeedShootSprites.

3.10 Abstract Class FlyingObject extends Entity

This class is a base class for all flying objects.

3.10.1 Fields

#int attackdamage	Self-explanatory.
#int length	this flying object length
#double angle	this object move angle
#double speed	Self-explanatory.
#boolean turnright	Self-explanatory.

3.10.2 Constructor

+FlyingObject(double angle,double x, double y)	Initialize FlyingObject. Set this flying object x to x. Set this flying object y to y. Set this flying object angle to angle

3.10.3 Method

+void draw(GraphicsContext gc)	Choose lookside on this image and rotate and draw image in canvas depend on this flying object turnright and angle
+boolean collide(Character other)	return true if this object collide other and other is not destroyed
+boolean CheckCollide()	return true if x <= 0 or x >= 960 or y <= 0 or y >= 640

+void UpdateCollideRect()	if this flyingobject collide Rectangle Set this flyingobject visible to false. Set this flyingobject destroyed to true.
+Getters and setters for all fields.	

3.11 Class Arrow extends FlyingObject

This class is used to draw and update the status of an arrow.

3.11.1 Fields

-boolean Percing	if true, an arrow will hit through all monsters.

3.11.2 Constructor

+Arrow(String name,double angle,double x, double y)	Initialize ArrowObject. Set this flying object x to x. Set this flying object y to y. Set this flying object angle to angle Set this flying object speed to 7. Set this length to 40. Set this name to name. if (this name equal “Arrow”) set this image to RenderableHolder.ArrowBasicSprite if (this name equal “Heavy Arrow”) set this image to RenderableHolder.ArrowHeavySprite if (this name equal “Light Arrow”) set this image to RenderableHolder.ArrowQuickSprite if (this name equal “Percing

	Arrow”) set this image to RenderableHolder.ArrowPercingSprit e

3.11.3 Method

+void update()	update this arrow x,y position and checking collide
+void UpdateCollideMonster()	if this arrow collide monster and this arrow percing equal to false Set this arrow destroyed to true. Set this arrow visible to false.
+Getters and setters for all fields.	

3.12 Class Fireball extends FlyingObject

This class is used to draw and update the status of a fireball, which is a Shooting monster’s attack ability.

3.12.1 Fields

-Hero h1	Hero or target of this FlyingObject.

3.12.2 Constructor

+Fireball(double angle,double x, double y)	Initialize Fireball. Set this flying object x to x. Set this flying object y to y. Set this flying object angle to angle Set this length to 40. Set this speed to 5. Set this Image to RenderableHolder.FireballSprite.

3.12.3 Method

+void update()	update this arrow x,y position and checking collide if this fireballObject collide h1/hero decrease h1/hero hp by this attackdamage and set this object to dead
+Getters and setters for all fields.	

3.13 Class Field implements IRenderable

This class is used to cropped stage images and draw each field on the canvas.

3.13.1 Fields

-final Image[] images	Array of maps/fields images
-ArrayList<WritableImage> croppedimages	Array that contains croppedimage
-Image img	image of this field

3.13.2 Constructor

+Field(int stage)	Initailze Field Set img to this images[stages]. Create a new ArrayList to croppedimages then add cropped images to croppedimages.

3.13.3 Method

+void draw(GraphicsContext gc)	draw all images from this field croppedimages to canvas
+void addCropped()	create new ArrayList to this croppedimages. Loop and add cropped images to this croppedimages array.

+Getters and setters for all fields.

4. Package block

This package consists of classes that represent all of the areas that the hero can not walk through.

4.1 Class Rectangle

This class represents an area that the hero can’t walk through and all flying objects will be destroyed when colliding this area. Such as walls and trees.

4.1.1 Fields

-int x	x position of top-left of the rectangle
-int y	y position of top-left of the rectangle
-int width	width of the rectangle
-int height	height of the rectangle

4.1.2 Constructor

+Rectangle(int x,int y,int width,int height)	Initialize Rectangle.

4.1.3 Methods

+boolean collide(Entity obj)	Return true if obj collides this Rectangle.
+boolean collide(double dx,double dy,int radius)	Return true if the hero collides this Rectangle.

4.2 Class WaterRectangle extends Rectangle

This class represents a water area that the hero can’t walk through, but a flying object can fly through.

4.2.1 Constructor

+WaterRectangle(int x, int y, int width, int height)	Initialize WaterRectangle.

5. Package input

This package consists of classes that input utility.

5.1 Class InputUtility

This class is used to contain input and utility making interaction with player and hero.

5.1.1 Field

-ArrayList<KeyCode> keyPressed = new ArrayList<>();	static Arraylist that contains keycode

5.1.2 Methods

+boolean getKeyPressed(KeyCode keycode)	return true if this keyPressed contains keycode
+void reset()	Create new arraylist for keyPressed
+void setKeyPressed(KeyCode keycode,boolean pressed)	if pressed is true and keyPressed not contains keycode add keycode to keyPressed array if pressed is false remove keycode from keyPressed array

6. Package drawing

This package consists of classes that represent everything that can be seen on the stage.

6.1 Class GameScreen extends Canvas

This class is used to draw all render objects and add listeners to the stage canvas.

6.1.1 Constructor

+GameScreen(double width, double height)	Initialize GameScreen.

6.1.2 Methods

+void addListerner()	Set setOnKeyPressed if Pressed setKeyPressed(keypress, true) on InputUtility. Set setOnKeyReleased if Released setKeyPressed(keyrelease, false) on InputUtility.
+void paintComponent()	draw all entities from RenderableHolder instance If this entity is destroyed, draw a dead scene too. If StageClear draws a sign arrow on the middle right too.

7. Package logic

This package consists of classes that contain all logic that needs to update and run.

7.1 Enum SceneMode

This enumeration is for switching background music by class SoundLogic.

```
{Menu, S1, S2, S3, S4}
```

7.2 Class Timer

7.2.1 Fields

-int minute	Minutes.
-int seconds	Seconds.
-int ms	Miliseconds.

7.2.2 Constructor

+Timer()	Initialize Timer with all fields equal 0.
+Timer(int m, int s, int ms)	Initialize Timer with the value of parameters.

7.2.3 Methods

+void increaseTimer(int amount)	Increase ms equals to amount and adjust all fields's value.
+boolean isTimerEmpty()	Return true if time equals 0 or below.
+String toString()	Return string in format "minute:second:ms"
+void reset()	Set all fields to 0.

+Getter and setters for all fields.

7.3 Class ItemLogic

This class contains all items in the game, controls the unlocking and equipping item system, and calculates the hero’s status.

7.3.1 Fields

-final ItemLogic instance = new ItemLogic();	Instance of ItemLogic.
-ArrayList<BaseItem> allItems	ArrayList contains all items in the game.
-ArrayList<BowArrow> allArrows	ArrayList contains all BowArrows in the game.
-ArrayList<Armor> allArmors	ArrayList contains all armors in the game.
-ArrayList<Ring> allRings	ArrayList contains all rings in the game.
-ArrayList<BaseItem> lockedCommon	ArrayList contains all items in the game which are common and locked.
-ArrayList<BaseItem> lockedRare	ArrayList contains all items in the game which are rare and locked.
-ArrayList<BaseItem> lockedEpic	ArrayList contains all items in the game which are epic and locked.
-ArrayList<BaseItem> lockedLegendary	ArrayList contains all items in the game which are legendary and locked.
-BowArrow equippingArrow	Equipping arrow.
-Armor equippingArmor	Equipping armor.
-Ring equippingRing	Equipping ring.

-int attackDamage	Hero attack damage calculated by equipping BowArrow and ring.
-int attackSpeed	Hero attack speed calculated by equipping BowArrow.
-int maxHealth	Hero maximum health point calculated by equipping armor and ring.
-double walkSpeed	Hero walk speed calculated by equipping armor and ring.
-String ArrowName	Name of the equipping BowArrow.
-boolean Percing	true if BowArrow has the ability to shoot piercingly.

7.3.2 Constructor

+ItemLogic()	-Initialize all items in the game. -Set the default equipment and locked items.

7.3.3 Methods

+void updateHeroStat()	Calculated all hero’s stats by equipping items.
+BaseItem randomUnlock(Rarity rarity)	Random item to unlock by rarity parameter and return that item. If all items in that rarity unlocked, return null.
+<u>int getRandomNumber(int min, int max)</u>	Random integer from max to min.
+Rarity randomRarityToUnlock(int stage)	Random rarity to unlock an item. (Rate of chances depend on stage’s level.)

+Getter and setters for all fields.

7.4 Class GameLogic

This class is used to update all logic and stages.

7.4.1 Fields

+ArrayList<Monster> Monsters	ArrayList that contains monster in this field
-ArrayList<Entity> gameCharacter	ArrayList that contains all entity
-Hero hero	hero/player that use to play
-boolean SubGameEnd	Substage end or not
-boolean StageGameEnd	Stage end or not

7.4.2 Constructor

+GameLogic()	Initialize GameLogic reset all input Set SubGameEnd to false. Set StageGameEnd to false. Set a new arraylist to gameCharacter. Set a new arraylist to Monster. reset RenderableHolder update hero stat Set new hero to hero if substage equal 0. Create game field Add monster in field to this Monster arraylist.

7.4.3 Methods

+List<Monster> getMonsters()	get Monsters static field
+void ResetContainer(boolean	reset all input

resethp)	Set SubGameEnd to false. Set StageGameEnd to false. Set a new arraylist to gameCharacter. Set a new arraylist to Monster. reset RenderableHolder update hero stat Set new hero to hero if substage equal 0. Create game field Add monster in field to this Monster arraylist.
+void AddHeroStat()	Add all stats to hero form ItemLogic.
+void addNewObject(Entity entity)	Add an entity to RenderableHolder.
+void logicUpdate()	if all monster are dead and hero x position is more than 900 and substage = 1 or hero is destroyed Set StageGameEnd to true. Create alert If SubGameEnd equal true +1 SubStage and reset all entities If SubGame not end and StageGame not end update all character or add arrow/fireball in this gameCharacter array
+void createfireballsArray()	loop every entity in this gameCharacter array If this entity is ShootingMonster, not destroyed and this ShootingMonster shoot is true create fireball object
+void UpdateAllCharacter()	update all entity using Thread

	If this entity isDestroyed remove from this gameCharacter array
+void CreateAlert()	If the hero is destroyed, call StageBadEnd(). Otherwise, unlock the next stage and call StageEndAlert().
+void AddReset()	+1 SubStage then reset all fields value and reset all input.
+void StageEndAlert()	Show stage cleared alert.
+void StageBadEnd()	Show failed alert.
+void BackMainMenu()	Set scene to main menu.
+void createarrow()	Create an arrow sprite if the hero shooting state is true.
+void createfireball(ShootingMonster m1)	Create Fireball using x, y, aimangle,attackspeedstatus and turnright from ShootingMonster. Set Fireball h1 to this GameLogic hero. Add Fireball to this gameCharacter array
+Getters and setters for all fields.	

7.5 Class StageLogic

This class contains all monsters and Rectangle for each stage, and contains the current stage and substage level.

7.5.1 Fields

-<u>StageLogic instance = new StageLogic();</u>	Instance of StageLogic.

-int stage	Current stage.
-int substage	Current sub-stage.
-ArrayList<Boolean> stageUnlocked	True if the stage is unlocked.
-final Monster[][] MonstersArray	Array of arrays contain monsters for each substage.
-final Rectangle[][] RectsArray	Array of arrays contain Rectangles for each sub-stage.
-Monster[] StageMonster	array that contains monster in this substage

7.5.2 Constructor

+StageLogic()	Initialize StageLogic. Set stage and substage to 0. Set stageUnlocked to {true, false, false, false}.

7.5.3 Methods

-Monster[] copy(Monster[] marray)	return a copy monster array from marray
+Monster[] getStageMonster()	Set StageMonster from MonstersArray[stage] then return this StageMonster
+Rectangle[] getRects()	Return an array of Rectangle of current substage.
+void setStage(int s)	Setter for current stage. Set sceneMode of SoundLogic Instance to the current stage.
+StageLogic getInstance()	Return instance.
+Getters and Setters for remaining.	

7.6 Class Soundlogic

This class contain all background musics and sound effects, and controls the background music.

7.6.1 Fields

-<u>SoundLogic instance = new SoundLogic();</u>	Instance of SoundLogic.
-SceneMode sceneMode	Current sceneMode.
-MediaPlayer menuPlayer, stage1Player, stage2Player, stage3Player, stage4Player	MediaPlayer for background music for each sceneMode.
-AudioClip clickSound, hitSound, monsterHitSound, monsterShootSound, loseSound, winSound, shootSound;	AudioClip for sound effects.
-int bgVolume	Volume level of background music.
-int sfxVolume	Volume level of sound effects.

7.6.2 Constructor

+SoundLogic()	Set all volumes to 100.

7.6.3 Methods

-<u>void loadResource()</u>	Load all music and sound effects from the resource file. This function will run only once when initializing new SoundLogic.
-updateBgSound()	Call stopAll() and play background music depending on a current sceneMode.
-void stopAll()	Stop all MediaPlayer.
-void updateBgVolume(double vol)	Set all MediaPlayers for background music volume to

	vol.
-void updateSfxVolume(double vol)	Set all AudioClips for sound effect volume to vol.
+<u>SoundLogic getInstance()</u>	Return instance.
+void setSceneMode(SceneMode sceneMode)	Set current sceneMode and call updateBgSound().
+void setBgVolume(int bgVolume)	Set bgVolume and call updateBgVolume(((double) bgVolume)/100).
+void setSfxVolume(int sfxVolume)	Set sfxVolume and call updateSfxVolume(((double) sfxVolume)/100).
+Getters for sceneMode, bgVolume, sfxVolume, and all AudioClips.	

8. Package scene

This package consists of classes that represent everything that can be seen on each switchable scene.

8.1 Class MainMenu extends VBox

This class is a root pane of the main menu.

8.1.1 Fields

-Label gameName	Game title.
-Button startButton	Start button.
-Button inventoryButton	Inventory button.
-Button tutorialButton	Tutorial button.
-Button optionButton	Option button.
-Button exitButton	Exit button.

8.1.2 Constructor

+MainMenu()	Initialize MainMenu. Set preference size to 960x640. Set alignment to center. Set spacing to 15. Add CSS stylesheet. Set ID to “root”. Call methods to initialize the game title and all buttons.

8.1.3 Methods

-void initGameNameLabel()	Initialize gameName and add itself to MainMenu.
-void initStartButton()	Initialize startButton and add itself to MainMenu.
-void initInventoryButton()	Initialize inventoryButton and add itself to MainMenu.
-void initTutorialButton()	Initialize tutorialButton and add itself to MainMenu.
-void initOptionButton()	Initialize optionButton and add itself to MainMenu.
-void initExitButton()	Initialize exitButton and add itself to MainMenu.

8.2 Class StageSelect extends VBox

This class is a root pane of the stage selection menu.

8.2.1 Fields

-ToolBar header	Toolbar of the scene.
-Button backButton	Back to the main menu button.
-Label sceneLabel	Scene label (Stages).

-HBox stagesPane	HBox contains all stage buttons.
-Button stage1Button	Stage 1 Button.
-Button stage2Button	Stage 2 Button.
-Button stage3Button	Stage 3 Button.
-Button stage4Button	Stage 4 Button.

8.2.2 Constructor

+StageSelect()	Initialize StageSelect. Set preference size to 960x640. Add CSS stylesheet. Set ID to “root”. Call methods to initialize header and stagePane.

8.2.3 Methods

-initHeader()	Initialize header and add itself to StageSelect.
-initStagesPane()	Initialize stagesPane and add itself to StageSelect.

8.3 Class Inventory extends VBox

This class is a root pane of the inventory menu.

8.3.1 Fields

-ToolBar header	Toolbar of the scene.
-Button backButton	Back to the main menu button.
-Label sceneLabel	Scene label (Inventory).
-VBox equipmentPane	VBox contains all functions in Inventory scene.
-Label arrowLabel,	Label for each item type.

armorLabel, ringLabel	
-GridPane arrowSlot, armorSlot, ringSlot	GridPane contain all item for each types
-ToggleGroup arrowGroup, armorGroup, ringGroup	ToggleGroup for each item types
-Button unEquipArmor, unEquipRing	Button for unequipping Armor and Ring.
-ArrayList<RadioButton> arr	RadioButton for choosing equipping BowArrow.
-ArrayList<RadioButton> arm	RadioButton for choosing equipping Armor.
-ArrayList<RadioButton> ring	RadioButton for choosing equipping Ring.
-Label statLabel	label for showing hero’s stat.
-GridPane statPane	GridPane for each hero’s stat.
-Text att	Text for showing the hero’s attack damage.
-Text hp	Text for showing the hero's health point.
-Text attSpeed	Text for showing the hero's attack speed.
-Text walkSpeed	Text for showing the hero’s walk speed.

8.3.2 Constructor

+Inventory()	Initialize StageSelect. Set preference size to 960x640. Add CSS stylesheet. Set ID to “root”. Call methods to initialize header, equipmentPane. Call showHeroStat().

	Call updateEquipping().

8.3.3 Methods

-void initHeader()	Initialize header and add itself to Equipment.
-void initEquipmentPane()	Initialize equipemntPane. Initialize Label, GridPane and call method for initializing for each item type. Add everything to equipmentPane. Add itself to Inventory.
-void initArrowSelector()	Initialize everything in GridPane of BowArrow selection.
-void initArmorSelector()	Initialize everything in GridPane of Armor selection.
-void initRingSelector()	Initialize everything in GridPane of Ring selection.
-void boxLayoutSetup(HBox box, Rarity rarity)	Static method for setup layout of HBox for each item. Set Border color depending on the rarity of the item.
-void itemImgLayoutSetup(ImageView img)	Static method for setup layout of item image.
-void itemTextLayoutSetup(Text text, String fontName, int fontSize)	Static method for setup text font
-void textBoxLayoutSetup(VBox box)	Static method for setup layout of VBox of item details.
-void showHeroStat()	initialize statLabel and empty statPane

-void updateHeroStatPane()	Clear everything in statPane. Setup text showing hero's stats.
-void updateEquipping()	(Being called when equipment changes.) Update equipping items in ItemLogic. Call updateHeroStat() in ItemLogic. Call updateHeroStatPane()

8.4 Class Tutorial extends VBox

This class is a root pane of the tutorial menu.

8.4.1 Fields

-ToolBar header	Toolbar of the scene.
-Button backButton	Back to the main menu button.
-Label sceneLabel	Scene label (Tutorial).
-VBox tutorialPane	Empty pane for adding an image of tutorial as a background.

8.4.2 Constructor

+Tutorial()	Initialize Tutorial. Set preference size to 960x640. Add CSS stylesheet. Set ID to "root". Call initHeader(). Call initTutorialPane().

8.4.3 Methods

-initHeader()	Initialize header and add itself to Tutorial.

-initTutorialPane()	Initialize tutorialPane and add itself to Tutorial.

8.5 Class Option extends VBox

This class is a root pane of the options menu.

8.5.1 Fields

-ToolBar header	Toolbar of the scene.
-Button backButton	Back to the main menu button.
-Label sceneLabel	Scene label (Option).
-VBox optionPane	VBox contains all options.
-Label bgSoundAdjustLabel , sfxSoundAdjustLabel;	Label for each sound option.
-Slider bgSoundAdjustSlider, sfxSoundAdjustSlider	Slider for each sound option.

8.5.2 Constructor

+Option()	Initialize Tutorial. Set preference size to 960x640. Add CSS stylesheet. Set ID to “root”. Call initHeader(). Call initOptionPane().

8.5.3 Methods

-initHeader()	Initialize header and add itself to Option.
-initOptionPane()	Initialize optionPane and add itself to Option.

8.6 Class StageScene extends StackPane

This class is a root pane of the stage scene.

8.6.1 Fields

-int frameNum	Frame counter.

8.6.2 Constructor

+StageScene()	Initialize StageScene. Set preference size to 960x640. Call initScene()

8.6.3 Method

-void initScene()	Initialize GameLogic Initialize GameScreen and add itself to StageScene Initialize AnimationTimer (120 FPS). Then, Start the AnimationTimer.

9. Package application

9.1 Class Main

Main class for running the application.

9.1.1 Fields

-<u>Stage guiStage</u>	GUI Stage.

9.1.2 Methods

+void main(String[] args)	launch(args);
+void start(Stage arg0) throws Exception	Set MainMenu as a scene.
+<u>Getter and setter for guiStage</u>	