Dokemon Dungeon

Documentation

Created by

Naphat Khoprasertthaworn 6430098021
Narongkorn Suktongchaiyakun 6430104121

2110215 Programming Methodology
Semester 1 Academic Year 2022
Chulalongkorn University

Chapter 1

General Information

Dokemon Dungeon is inspired by pokemon and darkest dungeon game. In this game, you will be a group of heroes slaying all monsters in the dungeon.

1.1 Hero

Hero in this game is a unit that players can control them.

1.2 Monster

Monster in this game is a unit that is controlled by game logic.

1.3 Skill

Skill in this game contains buff or debuff and will give them to the target unit or user.

1.3.1 Attack Skill

This skill will target the enemy.

1.3.2 Defense Skill

llThis skill will target ally.

1.3.3 Single target skill

This skill will target only one opponent unit or ally.

1.3.4 Multi-target skill

This skill will target the opponent party or the allies party.

1.4 Item

All item in this game is the potion. The potion will make units that use them gain buff or debuff according to the buff or debuff in the potion.

Damage reduction potion

Exhaust potion

Enhance potion

Healing potion

Vulnerability potion

1.5 Buff

1.5.1 Damage reduction

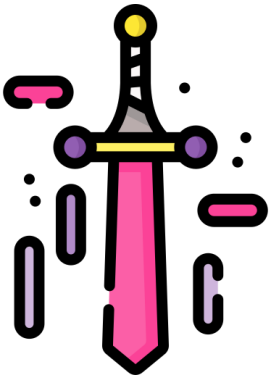
This buff will make the unit has increased defense.

1.5.2 Enhance

This buff will make the unit has an increased attack.

1.5.3 Exhaust

This debuff will make the unit has decreased attack.

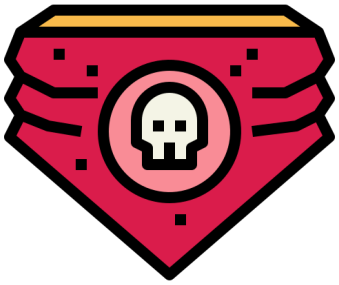


1.5.4 Regeneration

This buff will heal the unit directly and doesn’t have a turn.

1.5.5 Vulnerability

This buff will make the unit has decreased defense.



Chapter 2

Gameplay

2.1 Game rule

- In Dokemon Dungeon , you have to clear the dungeon through waves of monster. You will make a progress by rolling the dice. When your progress reaches the point, you will face the boss of the game. If you can defeat the boss, you'll pass the game!
- In each stage, You'll face with monsters and have to defeat them. When you pass your turn, the game will switch to monster turn and repeat until all heroes or all monsters die!
- The player can use the item only in player's turn.

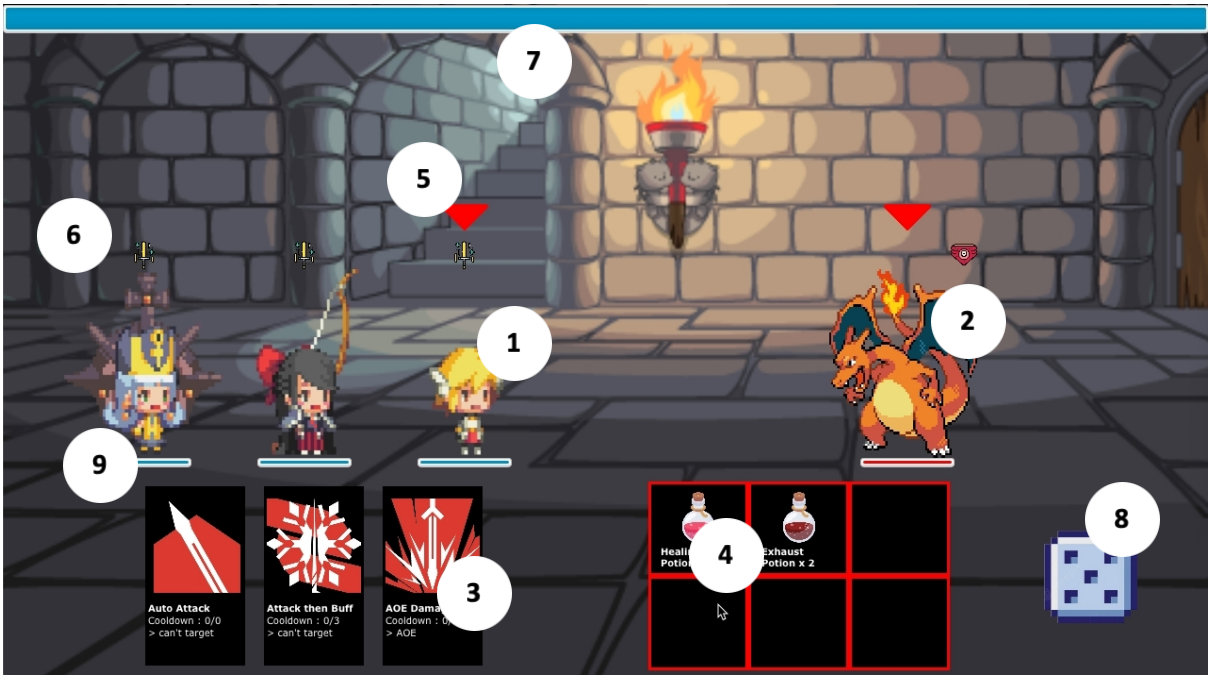
2.1.1 Target system

- Each side (Hero and monster) have a pointer on their side which can be selected by the Player.

2.1.2 Skill mechanic

- Using skill. If the skill is single target skill, then the skill will use on targeted unit. But if the skill can't target , the skill will use on front line unit.
- Using skill. If the skill is single target skill, then the skill will use on targeted unit. But if the skill can't target , the skill will use on front line unit.
- Multi target skill. The skill will use on every unit in the side depend on the skill is attack or defense skill. If it is attack skill , it will use on every monster. Otherwise it wil use on every heroes.
- Using item. If the item is buff item , then it will use on targeted ally. Otherwise it will use on targeted monster.
- Monster's single targer attack skill will always use on our back line unit.
- Monster's single defense skill will always use on lowest health monster.

2.2 Game Scene



1. Hero unit
2. Monster unit
3. Skills of hero in this turn.
4. Item and inventory.
5. Pointer in hero party and pointer in monster party.
6. Buff and debuff each unit.
7. progress bar.
8. dice.

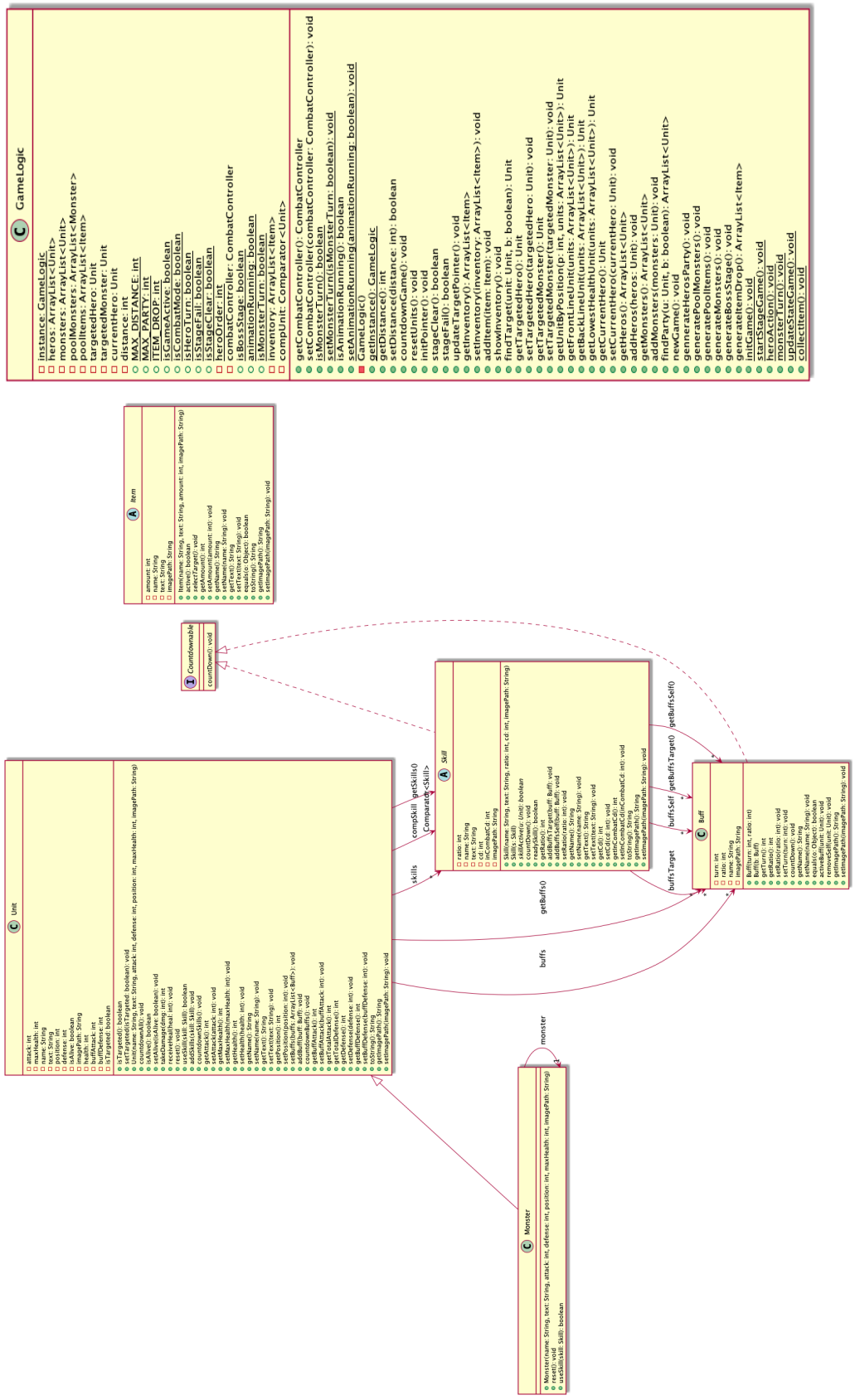
Chapter 3

Class diagram & JavaDoc

3.1 JavaDoc

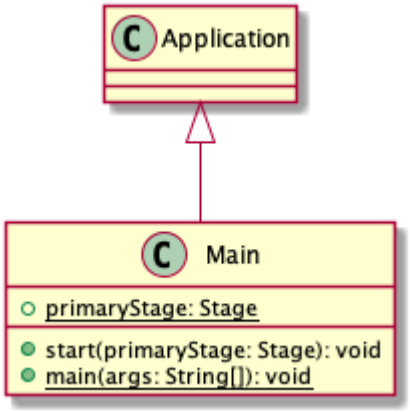
https://naphat-khoprasertthaworn.github.io/javadoc/

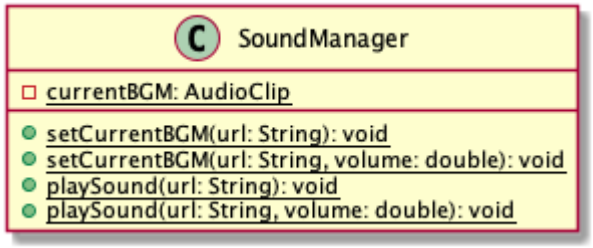
3.2 Class diagram

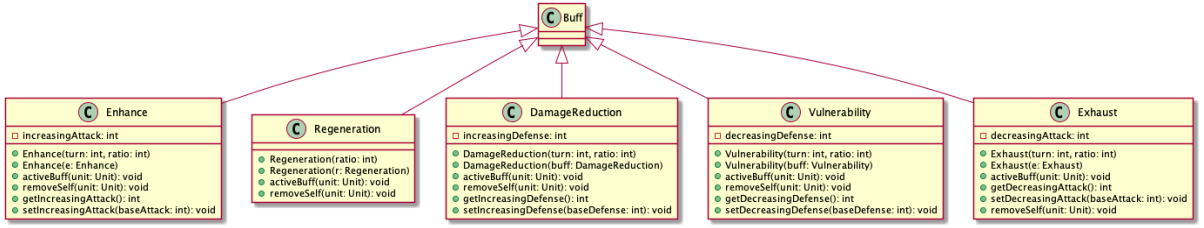


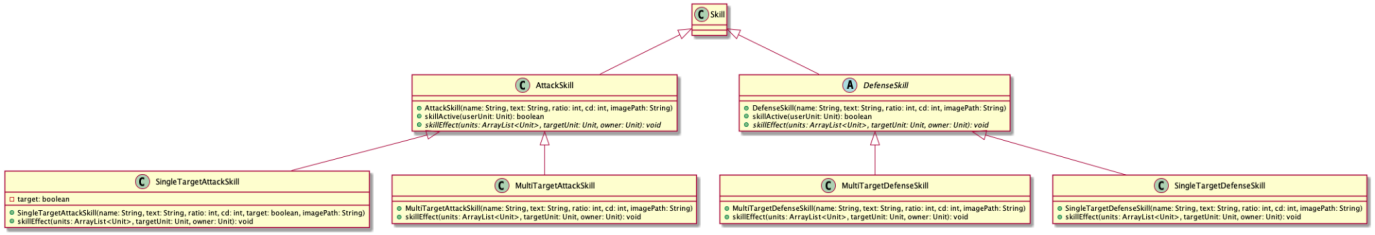
UML part 1

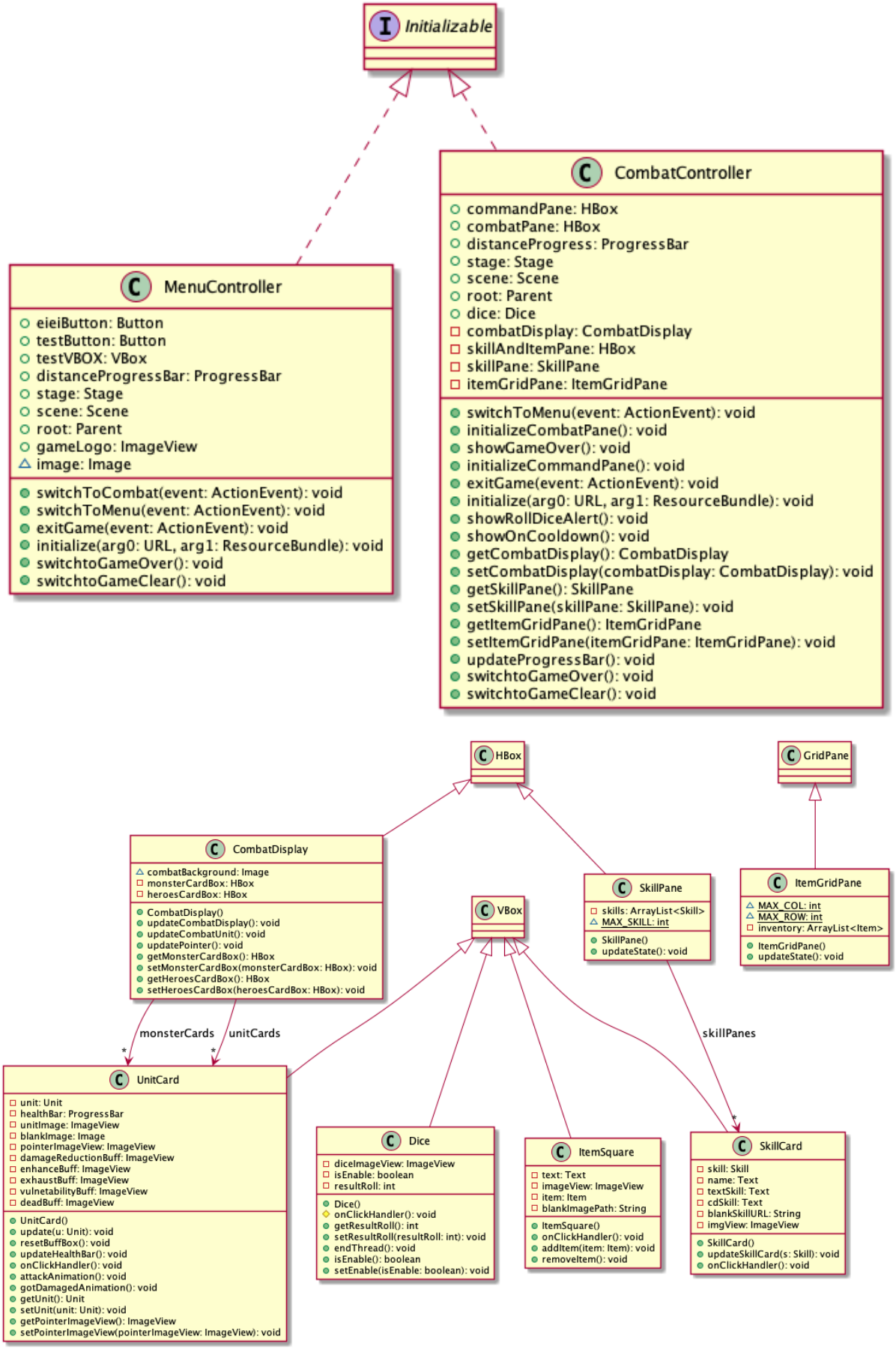
UML part 2


Application
Main
primaryStage: Stage
start(primaryStage: Stage): void
main(args: String[]): void


SoundManager
currentBGM: AudioClip
setCurrentBGM(url: String): void
setCurrentBGM(url: String, volume: double): void
playSound(url: String): void
playSound(url: String, volume: double): void

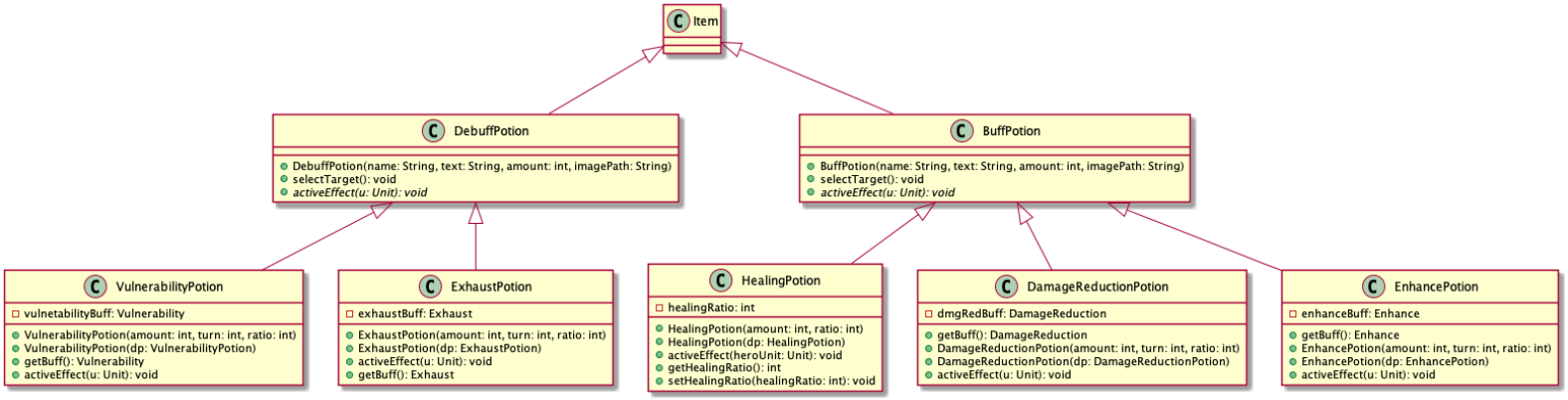

Buff
Enhance
increasingAttack: int
Enhance(turn: int, ratio: int)
Enhance(e: Enhance)
activeBuff(unit: Unit): void
removeSelf(unit: Unit): void
getIncreasingAttack(): int
setIncreasingAttack(baseAttack: int): void
Regeneration
Regeneration(ratio: int)
Regeneration(r: Regeneration)
activeBuff(unit: Unit): void
removeSelf(unit: Unit): void
DamageReduction
increasingDefense: int
DamageReduction(turn: int, ratio: int)
DamageReduction(buff: DamageReduction)
activeBuff(unit: Unit): void
removeSelf(unit: Unit): void
getIncreasingDefense(): int
setIncreasingDefense(baseDefense: int): void
Vulnerability
decreasingDefense: int
Vulnerability(turn: int, ratio: int)
Vulnerability(buff: Vulnerability)
activeBuff(unit: Unit): void
removeSelf(unit: Unit): void
getDecreasingDefense(): int
setDecreasingDefense(baseDefense: int): void
Exhaust
decreasingAttack: int
Exhaust(turn: int, ratio: int)
Exhaust(e: Exhaust)
activeBuff(unit: Unit): void
getDecreasingAttack(): int
setDecreasingAttack(baseAttack: int): void
removeSelf(unit: Unit): void


Skill
AttackSkill
AttackSkill(name: String, text: String, ratio: int, cd: int, imagePath: String)
skillActive(userUnit: Unit): boolean
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void
DefenseSkill
DefenseSkill(name: String, text: String, ratio: int, cd: int, imagePath: String)
skillActive(userUnit: Unit): boolean
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void
SingleTargetAttackSkill
target: boolean
SingleTargetAttackSkill(name: String, text: String, ratio: int, cd: int, target: boolean, imagePath: String)
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void
MultiTargetAttackSkill
MultiTargetAttackSkill(name: String, text: String, ratio: int, cd: int, imagePath: String)
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void
MultiTargetDefenseSkill
MultiTargetDefenseSkill(name: String, text: String, ratio: int, cd: int, imagePath: String)
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void
SingleTargetDefenseSkill
SingleTargetDefenseSkill(name: String, text: String, ratio: int, cd: int, imagePath: String)
skillEffect(units: ArrayList<Unit>, targetUnit: Unit, owner: Unit): void


Initializable
CombatController
commandPane: HBox
combatPane: HBox
distanceProgress: ProgressBar
stage: Stage
scene: Scene
root: Parent
dice: Dice
combatDisplay: CombatDisplay
skillAndItemPane: HBox
skillPane: SkillPane
itemGridPane: ItemGridPane
switchToMenu(event: ActionEvent): void
initializeCombatPane(): void
showGameOver(): void
initializeCommandPane(): void
exitGame(event: ActionEvent): void
initialize(arg0: URL, arg1: ResourceBundle): void
showRollDiceAlert(): void
showOnCooldown(): void
getCombatDisplay(): CombatDisplay
setCombatDisplay(combatDisplay: CombatDisplay): void
getSkillPane(): SkillPane
setSkillPane(skillPane: SkillPane): void
getItemGridPane(): ItemGridPane
setItemGridPane(itemGridPane: ItemGridPane): void
updateProgressBar(): void
switchtoGameOver(): void
switchtoGameClear(): void
MenuController
eieiButton: Button
testButton: Button
testVBOX: VBox
distanceProgressBar: ProgressBar
stage: Stage
scene: Scene
root: Parent
gameLogo: ImageView
image: Image
switchToCombat(event: ActionEvent): void
switchToMenu(event: ActionEvent): void
exitGame(event: ActionEvent): void
initialize(arg0: URL, arg1: ResourceBundle): void
switchtoGameOver(): void
switchtoGameClear(): void
HBox
GridPane
CombatDisplay
combatBackground: Image
monsterCardBox: HBox
heroesCardBox: HBox
CombatDisplay()
updateCombatDisplay(): void
updateCombatUnit(): void
updatePointer(): void
getMonsterCardBox(): HBox
setMonsterCardBox(monsterCardBox: HBox): void
getHeroesCardBox(): HBox
setHeroesCardBox(heroesCardBox: HBox): void
SkillPane
skills: ArrayList<Skill>
MAX_SKILL: int
SkillPane()
updateState(): void
ItemGridPane
MAX_COL: int
MAX_ROW: int
inventory: ArrayList<Item>
ItemGridPane()
updateState(): void
VBox
monsterCards
unitCards
skillPanes
UnitCard
unit: Unit
healthBar: ProgressBar
unitImage: ImageView
blankImage: Image
pointerImageView: ImageView
damageReductionBuff: ImageView
enhanceBuff: ImageView
exhaustBuff: ImageView
vulnetabilityBuff: ImageView
deadBuff: ImageView
UnitCard()
update(u: Unit): void
resetBuffBox(): void
updateHealthBar(): void
onClickHandler(): void
attackAnimation(): void
gotDamagedAnimation(): void
getUnit(): Unit
setUnit(unit: Unit): void
getPointerImageView(): ImageView
setPointerImageView(pointerImageView: ImageView): void
Dice
diceImageView: ImageView
isEnable: boolean
resultRoll: int
Dice()
onClickHandler(): void
getResultRoll(): int
setResultRoll(resultRoll: int): void
endThread(): void
isEnable(): boolean
setEnable(isEnable: boolean): void
ItemSquare
text: Text
imageView: ImageView
item: Item
blankImagePath: String
ItemSquare()
onClickHandler(): void
addItem(item: Item): void
removeItem(): void
SkillCard
skill: Skill
name: Text
textSkill: Text
cdSkill: Text
blankSkillURL: String
imgView: ImageView
SkillCard()
updateSkillCard(s: Skill): void
onClickHandler(): void

UML part 3

UML part 4


Item
DebuffPotion
DebuffPotion(name: String, text: String, amount: int, imagePath: String)
selectTarget(): void
activeEffect(u: Unit): void
BuffPotion
BuffPotion(name: String, text: String, amount: int, imagePath: String)
selectTarget(): void
activeEffect(u: Unit): void
VulnerabilityPotion
vulnetabilityBuff: Vulnerability
VulnerabilityPotion(amount: int, turn: int, ratio: int)
VulnerabilityPotion(dp: VulnerabilityPotion)
getBuff(): Vulnerability
activeEffect(u: Unit): void
ExhaustPotion
exhaustBuff: Exhaust
ExhaustPotion(amount: int, turn: int, ratio: int)
ExhaustPotion(dp: ExhaustPotion)
activeEffect(u: Unit): void
getBuff(): Exhaust
HealingPotion
healingRatio: int
HealingPotion(amount: int, ratio: int)
HealingPotion(dp: HealingPotion)
activeEffect(heroUnit: Unit): void
getHealingRatio(): int
setHealingRatio(healingRatio: int): void
DamageReductionPotion
dmgRedBuff: DamageReduction
getBuff(): DamageReduction
DamageReductionPotion(amount: int, turn: int, ratio: int)
DamageReductionPotion(dp: DamageReductionPotion)
activeEffect(u: Unit): void
EnhancePotion
enhanceBuff: Enhance
getBuff(): Enhance
EnhancePotion(amount: int, turn: int, ratio: int)
EnhancePotion(dp: EnhancePotion)
activeEffect(u: Unit): void