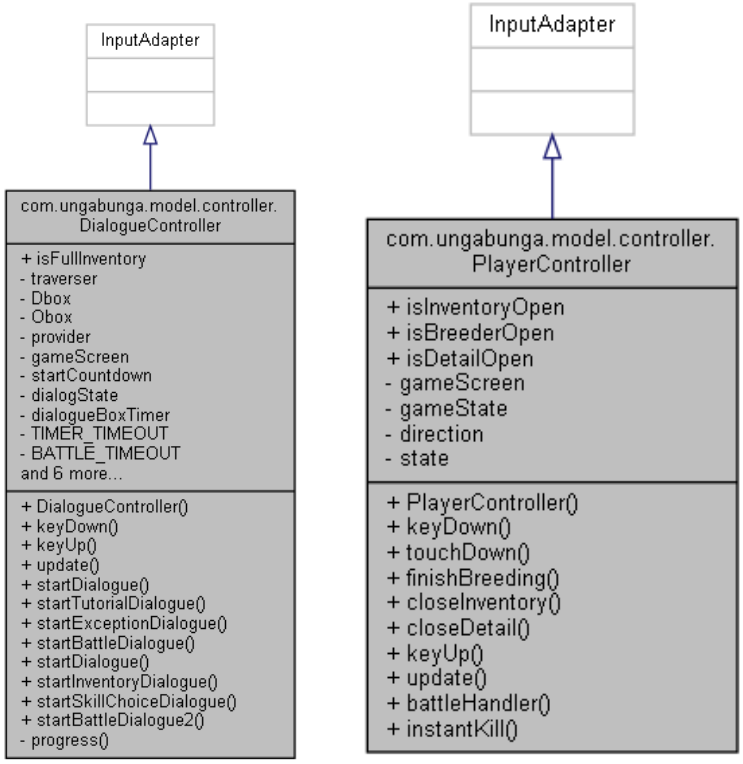
Kelas : K02
Nama Kelompok : Marcello Pokemon God
1. 13519059 / Denilsen Axel Candiasa
2. 13519068 / Roy H Simbolon
3. 13519079 / Jesson Gosal Yo
4. 13519086 / Marcello Faria
5. 13519090 / Alexander
6. 13519104 / Nabelanita Utami

Asisten Pembimbing : 13517068 / Abel Stanley

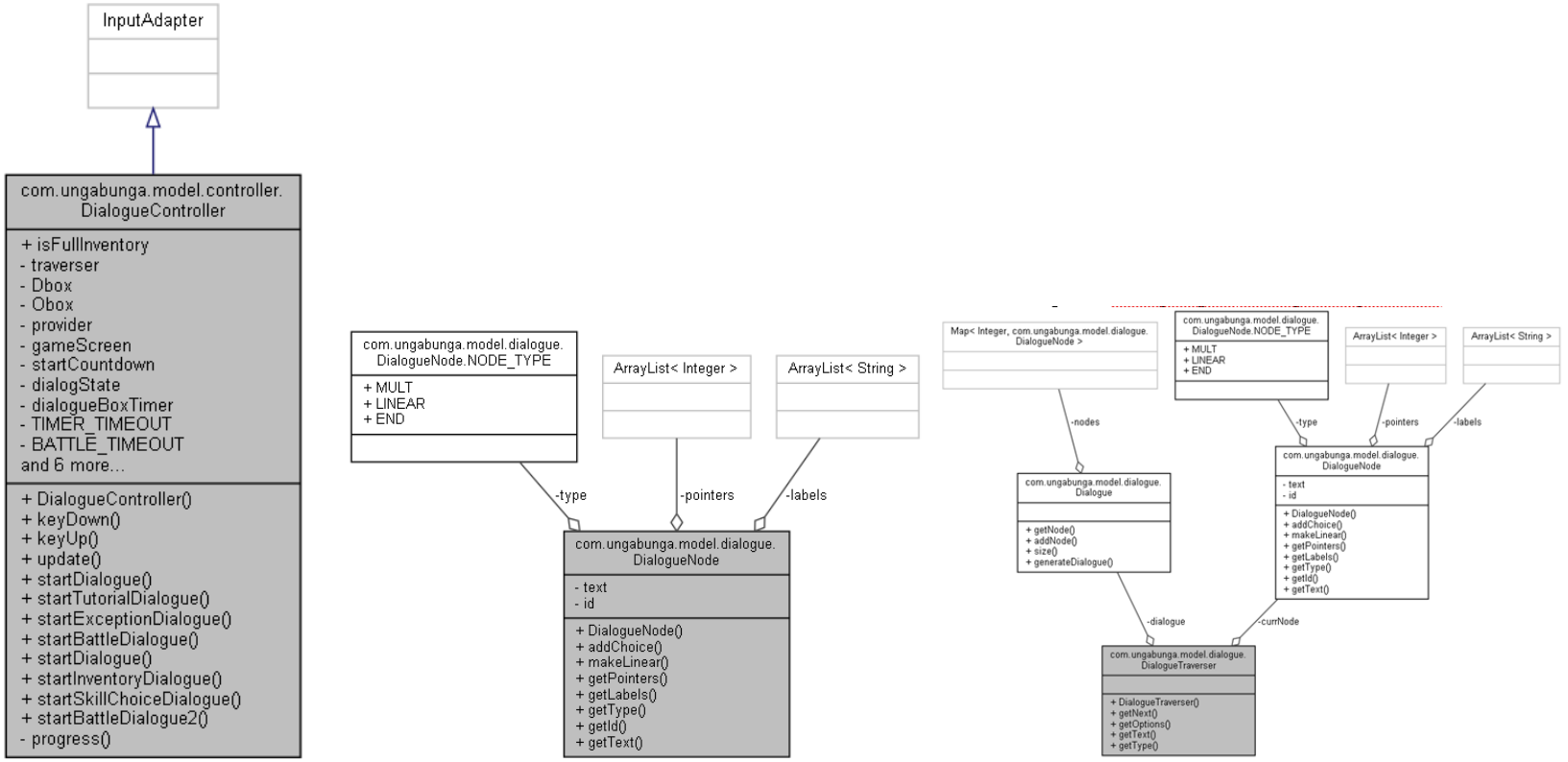
1. Diagram Kelas

Terdapat Total 72 Kelas, 2 Kelas Controller, 3 Dialog, 12 Entity, 1 Interface, 5 enums, 13 Exception, 1 Save, 9 Screen, 2 Thread, 8 UI, 6 utilities, 1 GameState,1 Engimon Game, 1 Settings dan 7 kelas Testing.

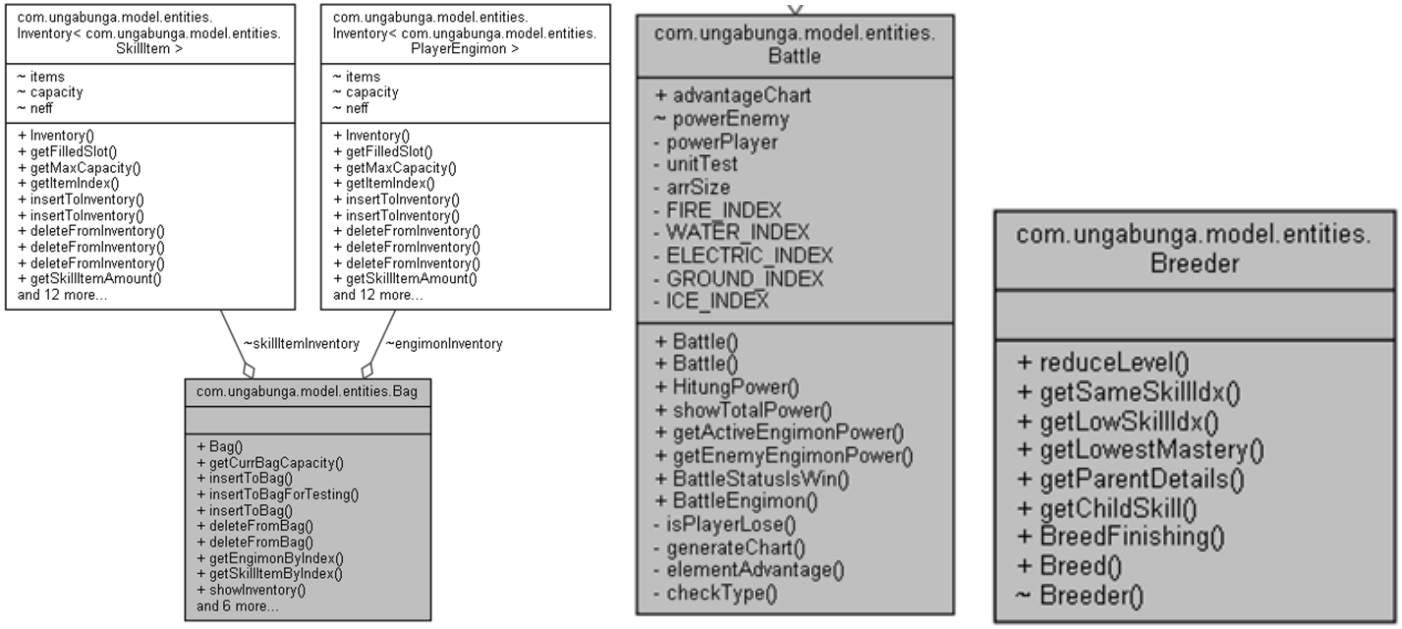
1.1. Controller

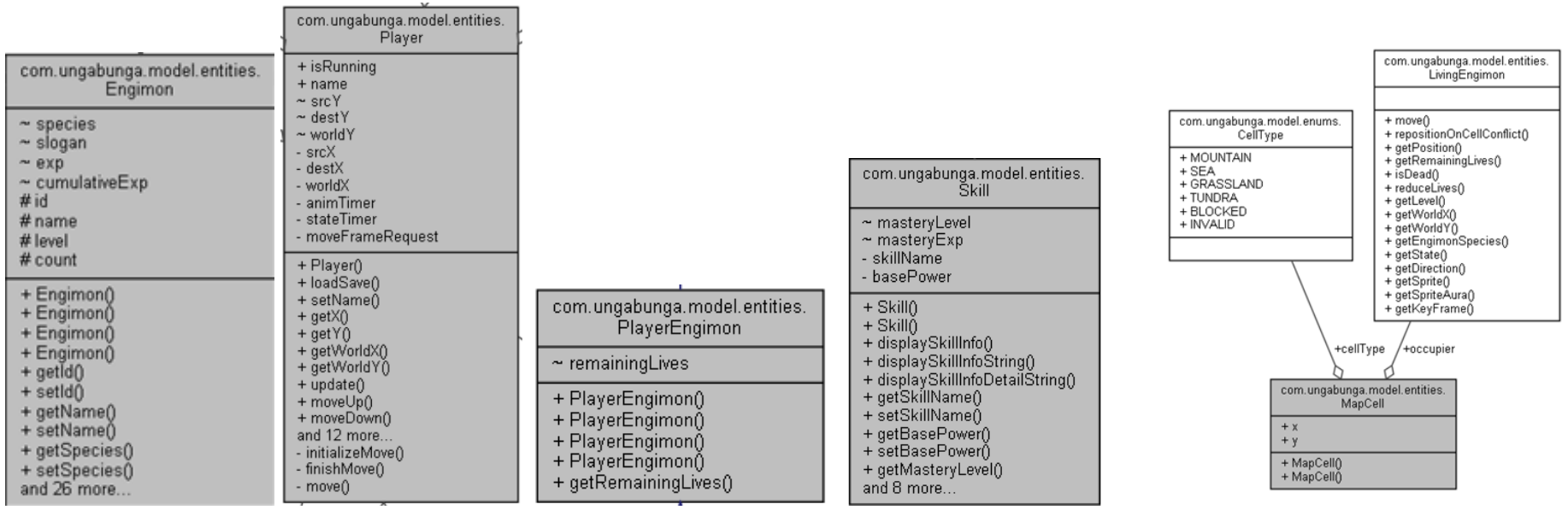


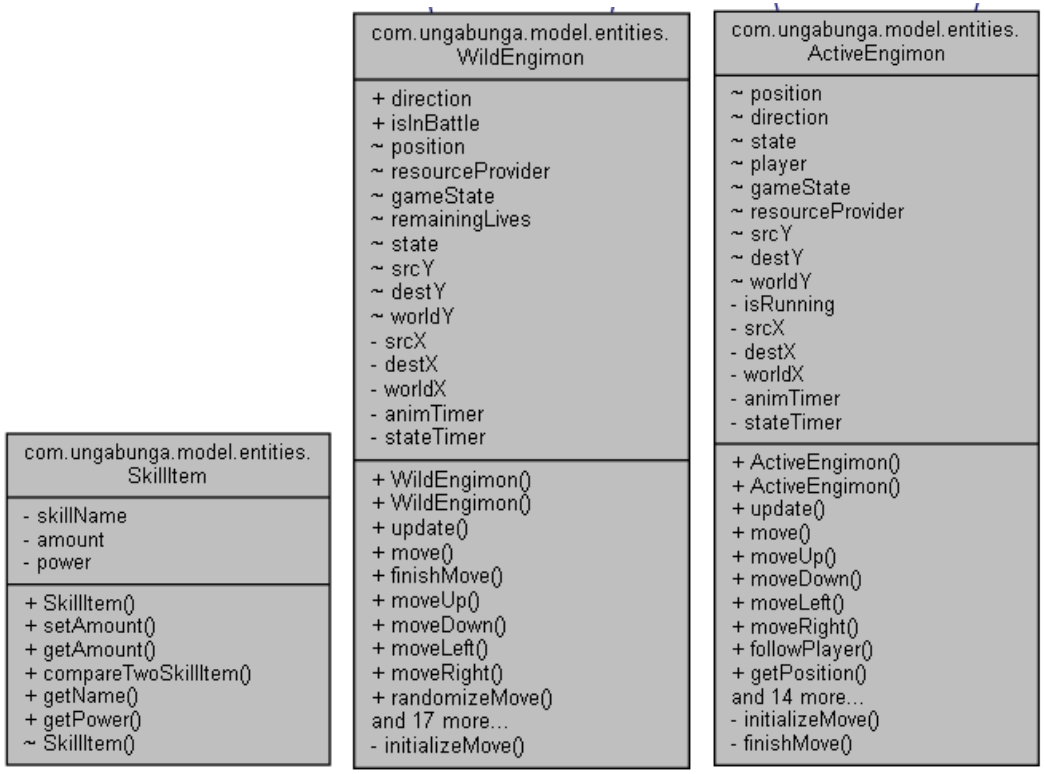
1.2. Dialog


InputAdapter
com.ungabunga.model.controller.
DialogueController
+ isFullInventory
- traverser
- Dbox
- Obox
- provider
- gameScreen
- startCountdown
- dialogState
- dialogueBoxTimer
- TIMER_TIMEOUT
- BATTLE_TIMEOUT
and 6 more...
+ DialogueController()
+ keyDown()
+ keyUp()
+ update()
+ startDialogue()
+ startTutorialDialogue()
+ startExceptionDialogue()
+ startBattleDialogue()
+ startDialogue()
+ startInventoryDialogue()
+ startSkillChoiceDialogue()
+ startBattleDialogue2()
- progress()
com.ungabunga.model.dialogue.
DialogueNode.NODE_TYPE
+ MULT
+ LINEAR
+ END
ArrayList< Integer >
ArrayList< String >
-type
-pointers
-labels
com.ungabunga.model.dialogue.
DialogueNode
- text
- id
+ DialogueNode()
+ addChoice()
+ makeLinear()
+ getPointers()
+ getLabels()
+ getType()
+ getId()
+ getText()
Map< Integer, com.ungabunga.model.dialogue.
DialogueNode >
-nodes
com.ungabunga.model.dialogue.
Dialogue
+ getNode()
+ addNode()
+ size()
+ generateDialogue()
-dialogue
-currNode
com.ungabunga.model.dialogue.
DialogueTraverser
+ DialogueTraverser()
+ getNext()
+ getOptions()
+ getText()
+ getType()

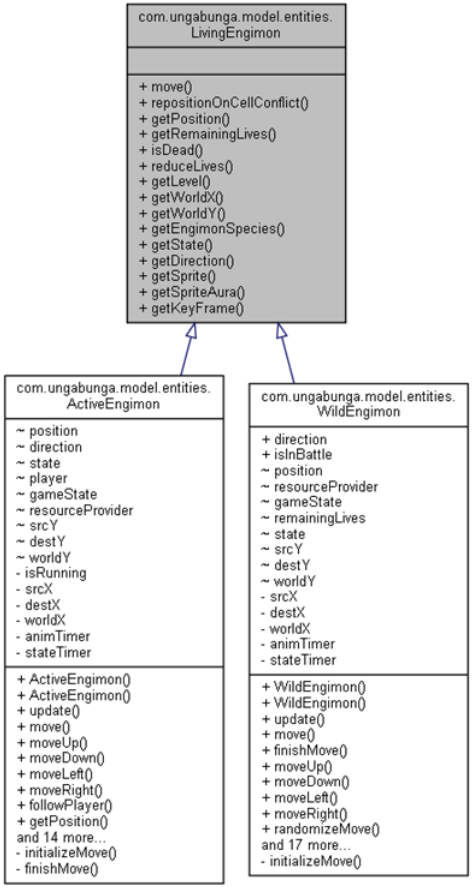
1.3. Entity


com.ungabunga.model.entities.
Inventory< com.ungabunga.model.entities.
SkillItem >
~ items
~ capacity
~ neff
+ Inventory()
+ getFilledSlot()
+ getMaxCapacity()
+ getItemIndex()
+ insertToInventory()
+ insertToInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ getSkillItemAmount()
and 12 more...
com.ungabunga.model.entities.
Inventory< com.ungabunga.model.entities.
PlayerEngimon >
~ items
~ capacity
~ neff
+ Inventory()
+ getFilledSlot()
+ getMaxCapacity()
+ getItemIndex()
+ insertToInventory()
+ insertToInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ getSkillItemAmount()
and 12 more...
~skillItemInventory
~engimonInventory
com.ungabunga.model.entities.Bag
+ Bag()
+ getCurrBagCapacity()
+ insertToBag()
+ insertToBagForTesting()
+ insertToBag()
+ deleteFromBag()
+ deleteFromBag()
+ getEngimonByIndex()
+ getSkillItemByIndex()
+ showInventory()
and 6 more...
com.ungabunga.model.entities.
Battle
+ advantageChart
~ powerEnemy
- powerPlayer
- unitTest
- arrSize
- FIRE_INDEX
- WATER_INDEX
- ELECTRIC_INDEX
- GROUND_INDEX
- ICE_INDEX
+ Battle()
+ Battle()
+ HitungPower()
+ showTotalPower()
+ getActiveEngimonPower()
+ getEnemyEngimonPower()
+ BattleStatusIsWin()
+ BattleEngimon()
- isPlayerLose()
- generateChart()
- elementAdvantage()
- checkType()
com.ungabunga.model.entities.
Breeder
+ reduceLevel()
+ getSameSkillIdx()
+ getLowSkillIdx()
+ getLowestMastery()
+ getParentDetails()
+ getChildSkill()
+ BreedFinishing()
+ Breed()
~ Breeder()

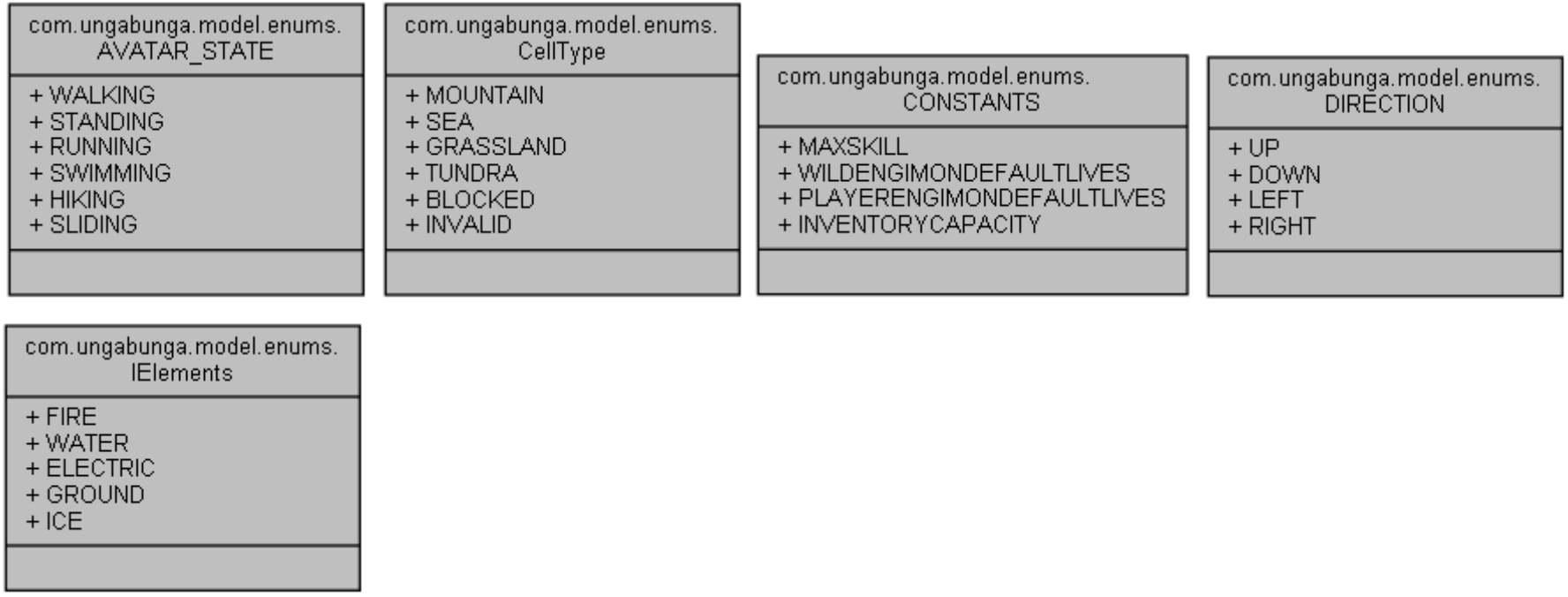

com.ungabunga.model.entities.
Engimon
~ species
~ slogan
~ exp
~ cumulativeExp
# id
# name
# level
# count
+ Engimon()
+ Engimon()
+ Engimon()
+ Engimon()
+ getId()
+ setId()
+ getName()
+ setName()
+ getSpecies()
+ setSpecies()
and 26 more...
com.ungabunga.model.entities.
Player
+ isRunning
+ name
~ srcY
~ destY
~ worldY
- srcX
- destX
- worldX
- animTimer
- stateTimer
- moveFrameRequest
+ Player()
+ loadSave()
+ setName()
+ getX()
+ getY()
+ getWorldX()
+ getWorldY()
+ update()
+ moveUp()
+ moveDown()
and 12 more...
- initializeMove()
- finishMove()
- move()
com.ungabunga.model.entities.
PlayerEngimon
~ remainingLives
+ PlayerEngimon()
+ PlayerEngimon()
+ PlayerEngimon()
+ PlayerEngimon()
+ getRemainingLives()
com.ungabunga.model.entities.
Skill
~ masteryLevel
~ masteryExp
- skillName
- basePower
+ Skill()
+ Skill()
+ displaySkillInfo()
+ displaySkillInfoString()
+ displaySkillInfoDetailString()
+ getSkillName()
+ setSkillName()
+ getBasePower()
+ setBasePower()
+ getMasteryLevel()
and 8 more...
com.ungabunga.model.enums.
CellType
+ MOUNTAIN
+ SEA
+ GRASSLAND
+ TUNDRA
+ BLOCKED
+ INVALID
com.ungabunga.model.entities.
LivingEngimon
+ move()
+ repositionOnCellConflict()
+ getPosition()
+ getRemainingLives()
+ isDead()
+ reduceLives()
+ getLevel()
+ getWorldX()
+ getWorldY()
+ getEngimonSpecies()
+ getState()
+ getDirection()
+ getSprite()
+ getSpriteAura()
+ getKeyFrame()
+cellType
+occupier
com.ungabunga.model.entities.
MapCell
+ x
+ y
+ MapCell()
+ MapCell()


com.ungabunga.model.entities.
SkillItem
- skillName
- amount
- power
+ SkillItem()
+ setAmount()
+ getAmount()
+ compareTwoSkillItem()
+ getName()
+ getPower()
~ SkillItem()
com.ungabunga.model.entities.
WildEngimon
+ direction
+ isInBattle
~ position
~ resourceProvider
~ gameState
~ remainingLives
~ state
~ srcY
~ destY
~ worldY
- srcX
- destX
- worldX
- animTimer
- stateTimer
+ WildEngimon()
+ WildEngimon()
+ update()
+ move()
+ finishMove()
+ moveUp()
+ moveDown()
+ moveLeft()
+ moveRight()
+ randomizeMove()
and 17 more...
- initializeMove()
com.ungabunga.model.entities.
ActiveEngimon
~ position
~ direction
~ state
~ player
~ gameState
~ resourceProvider
~ srcY
~ destY
~ worldY
- isRunning
- srcX
- destX
- worldX
- animTimer
- stateTimer
+ ActiveEngimon()
+ ActiveEngimon()
+ update()
+ move()
+ moveUp()
+ moveDown()
+ moveLeft()
+ moveRight()
+ followPlayer()
+ getPosition()
and 14 more...
- initializeMove()
- finishMove()

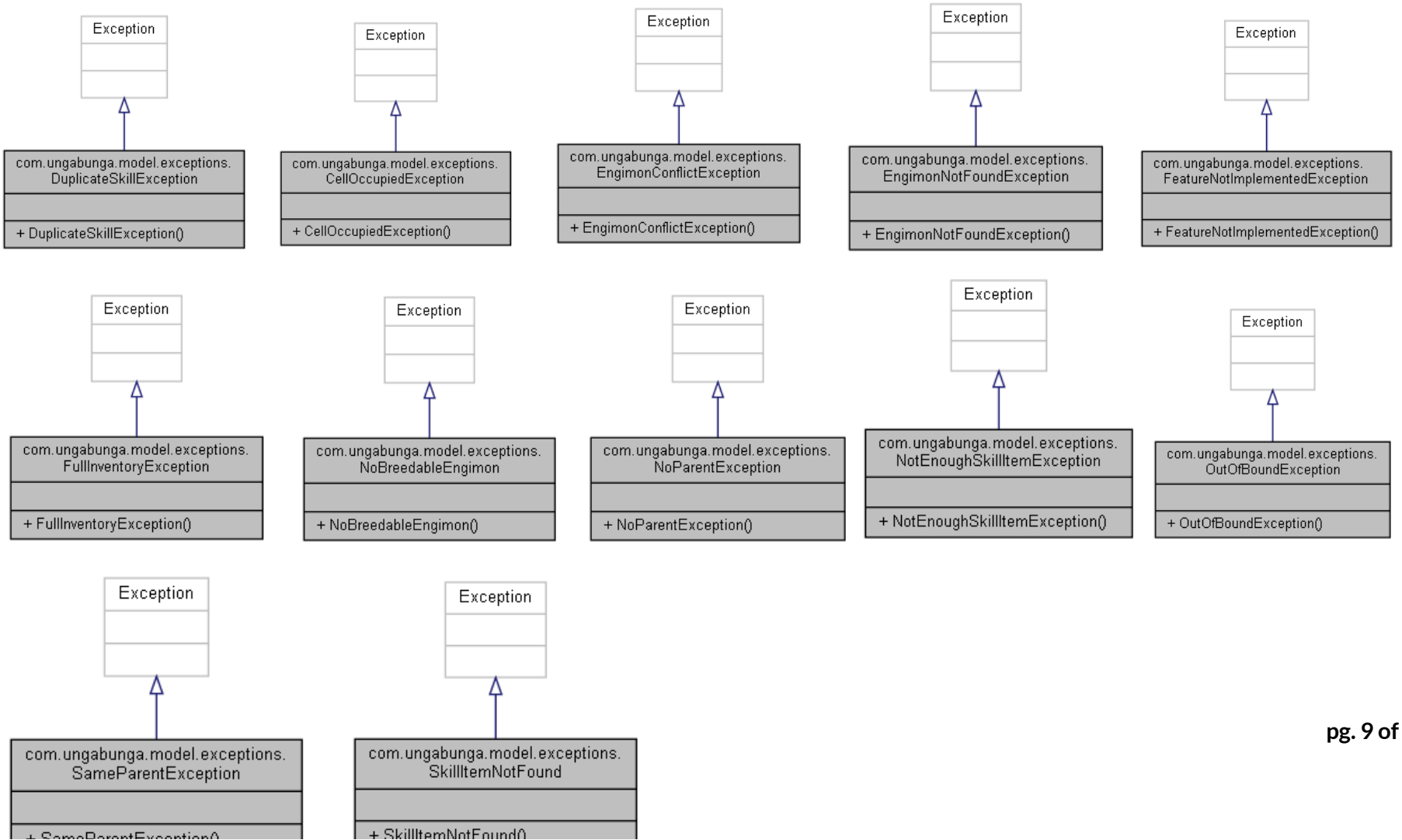
1.4. Interface



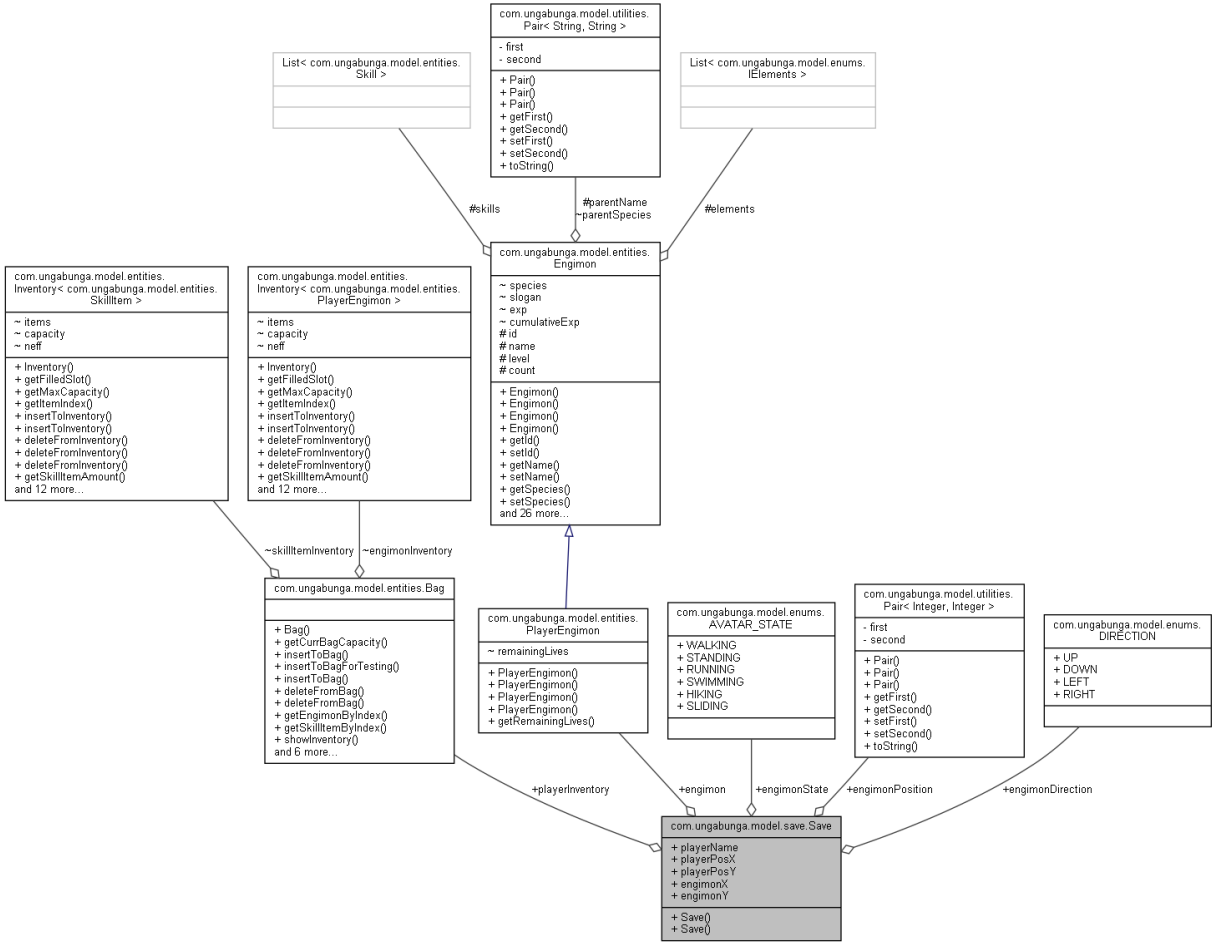
1.5. Enums


com.ungabunga.model.enums.
AVATAR_STATE
+ WALKING
+ STANDING
+ RUNNING
+ SWIMMING
+ HIKING
+ SLIDING
com.ungabunga.model.enums.
CellType
+ MOUNTAIN
+ SEA
+ GRASSLAND
+ TUNDRA
+ BLOCKED
+ INVALID
com.ungabunga.model.enums.
CONSTANTS
+ MAXSKILL
+ WILDENGIMONDEFAULTLIVES
+ PLAYERENGIMONDEFAULTLIVES
+ INVENTORYCAPACITY
com.ungabunga.model.enums.
DIRECTION
+ UP
+ DOWN
+ LEFT
+ RIGHT
com.ungabunga.model.enums.
IElements
+ FIRE
+ WATER
+ ELECTRIC
+ GROUND
+ ICE

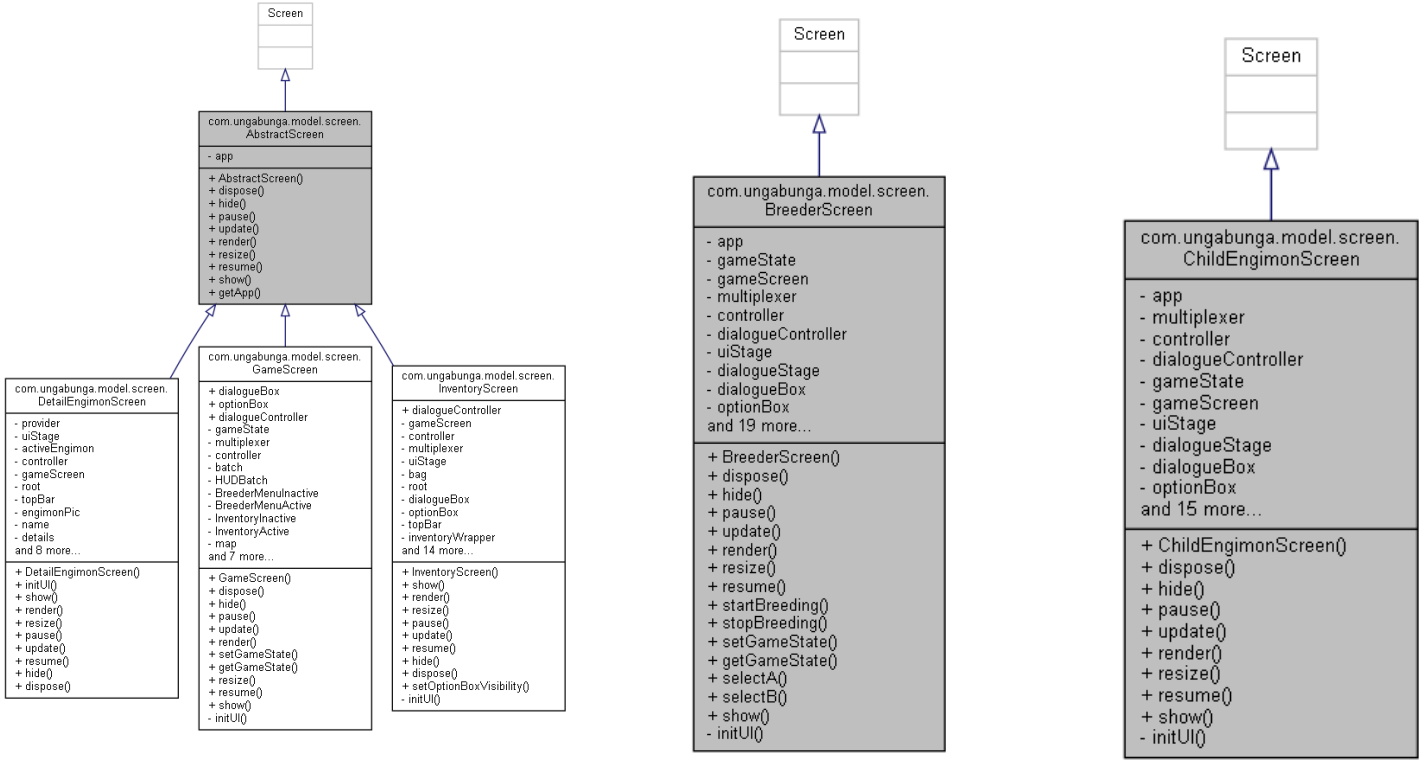
1.6. Exception

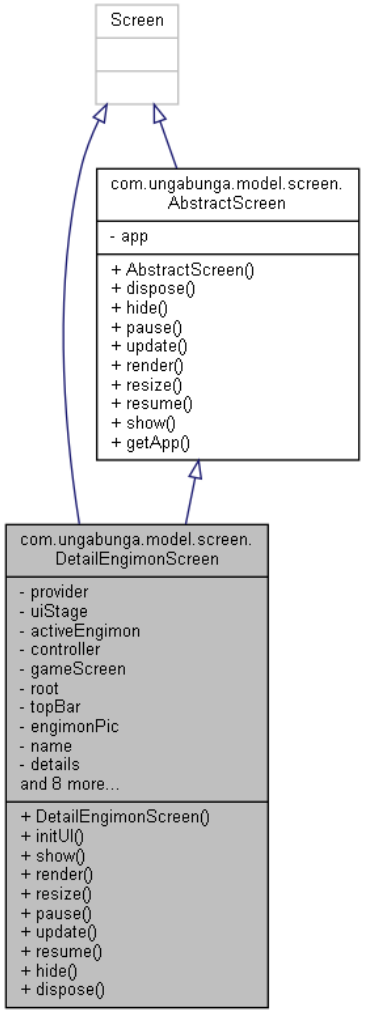


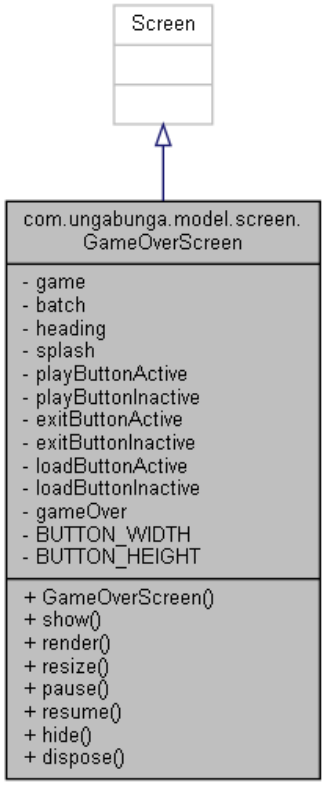
1.7. Save

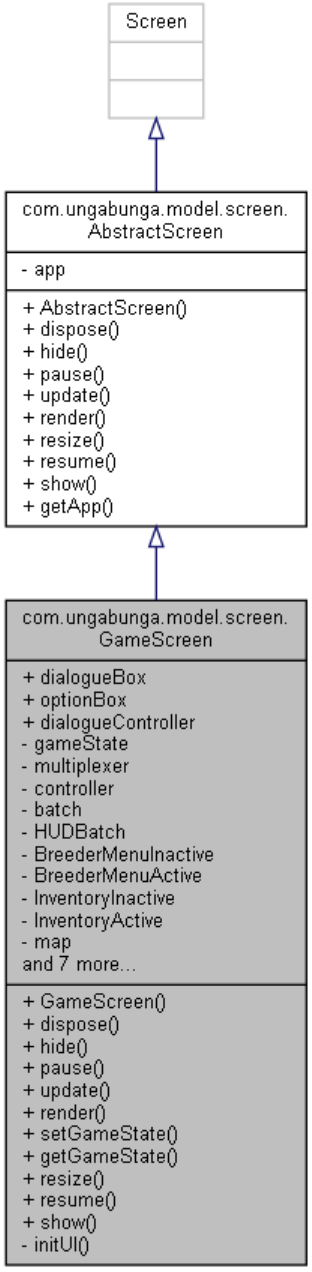

com.ungabunga.model.utilities.
Pair< String, String >
- first
- second
+ Pair()
+ Pair()
+ Pair()
+ getFirst()
+ getSecond()
+ setFirst()
+ setSecond()
+ toString()
List< com.ungabunga.model.entities.
Skill >
List< com.ungabunga.model.enums.
IElements >
#skills
#parentName
~parentSpecies
#elements
com.ungabunga.model.entities.
Engimon
~ species
~ slogan
~ exp
~ cumulativeExp
# id
# name
# level
# count
+ Engimon()
+ Engimon()
+ Engimon()
+ Engimon()
+ getId()
+ setId()
+ getName()
+ setName()
+ getSpecies()
+ setSpecies()
and 26 more...
com.ungabunga.model.entities.
Inventory< com.ungabunga.model.entities.
SkillItem >
~ items
~ capacity
~ neff
+ Inventory()
+ getFilledSlot()
+ getMaxCapacity()
+ getItemIndex()
+ insertToInventory()
+ insertToInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ getSkillItemAmount()
and 12 more...
com.ungabunga.model.entities.
Inventory< com.ungabunga.model.entities.
PlayerEngimon >
~ items
~ capacity
~ neff
+ Inventory()
+ getFilledSlot()
+ getMaxCapacity()
+ getItemIndex()
+ insertToInventory()
+ insertToInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ deleteFromInventory()
+ getSkillItemAmount()
and 12 more...
~skillItemInventory
~engimonInventory
com.ungabunga.model.entities.Bag
+ Bag()
+ getCurrBagCapacity()
+ insertToBag()
+ insertToBagForTesting()
+ insertToBag()
+ deleteFromBag()
+ deleteFromBag()
+ getEngimonByIndex()
+ getSkillItemByIndex()
+ showInventory()
and 6 more...
com.ungabunga.model.entities.
PlayerEngimon
~ remainingLives
+ PlayerEngimon()
+ PlayerEngimon()
+ PlayerEngimon()
+ PlayerEngimon()
+ getRemainingLives()
com.ungabunga.model.enums.
AVATAR_STATE
+ WALKING
+ STANDING
+ RUNNING
+ SWIMMING
+ HIKING
+ SLIDING
com.ungabunga.model.utilities.
Pair< Integer, Integer >
- first
- second
+ Pair()
+ Pair()
+ Pair()
+ getFirst()
+ getSecond()
+ setFirst()
+ setSecond()
+ toString()
com.ungabunga.model.enums.
DIRECTION
+ UP
+ DOWN
+ LEFT
+ RIGHT
+playerInventory
+engimon
+engimonState
+engimonPosition
+engimonDirection
com.ungabunga.model.save.Save
+ playerName
+ playerPosX
+ playerPosY
+ engimonX
+ engimonY
+ Save()
+ Save()

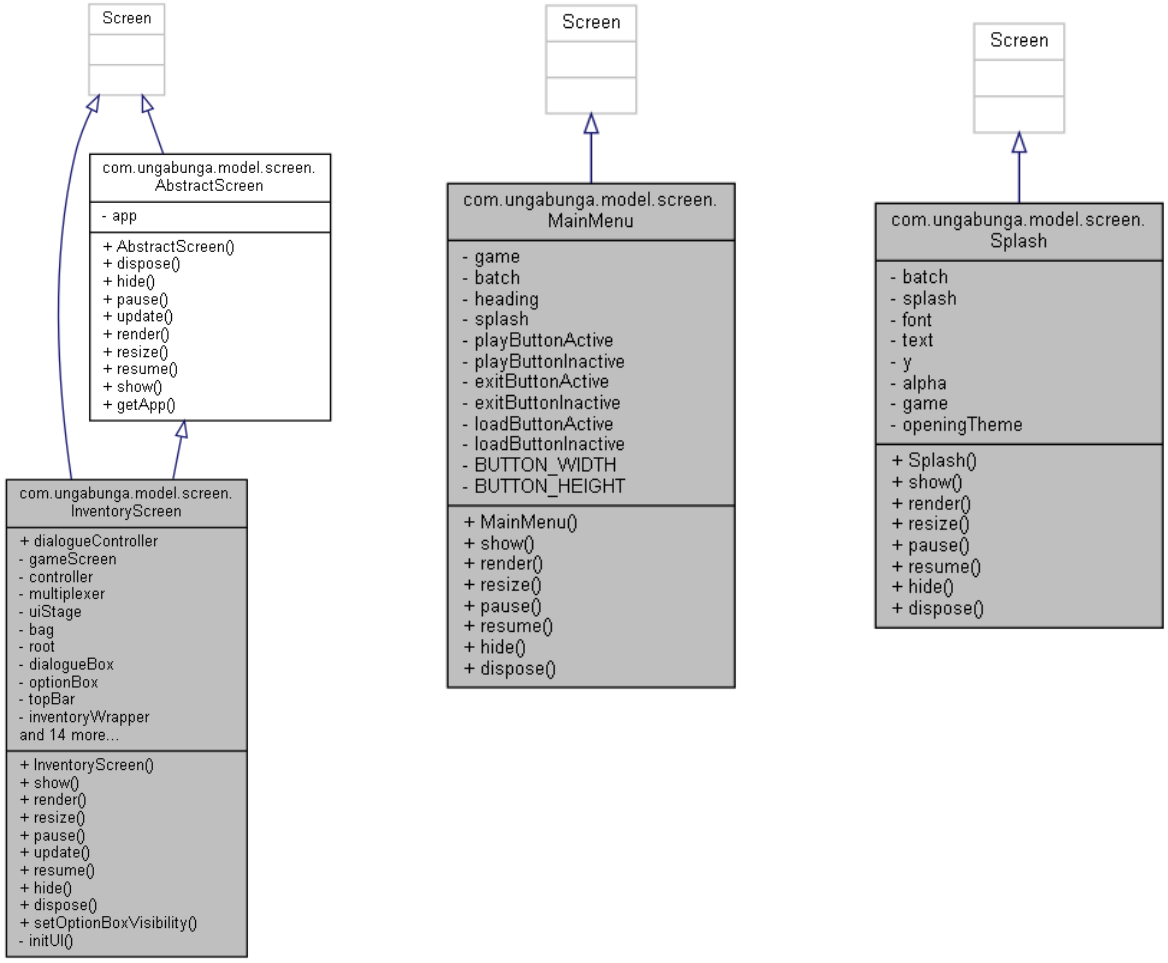
1.8. Screen


Screen
com.ungabunga.model.screen.AbstractScreen
- app
+ AbstractScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ show()
+ getApp()
com.ungabunga.model.screen.DetailEngimonScreen
- provider
- uiStage
- activeEngimon
- controller
- gameScreen
- root
- topBar
- engimonPic
- name
- details
and 8 more...
+ DetailEngimonScreen()
+ initUI()
+ show()
+ render()
+ resize()
+ pause()
+ update()
+ resume()
+ hide()
+ dispose()
com.ungabunga.model.screen.GameScreen
+ dialogueBox
+ optionBox
+ dialogueController
- gameState
- multiplexer
- controller
- batch
- HUDBatch
- BreederMenuInactive
- BreederMenuActive
- InventoryInactive
- InventoryActive
- map
and 7 more...
+ GameScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ setGameState()
+ getGameState()
+ resize()
+ resume()
+ show()
- initUI()
com.ungabunga.model.screen.InventoryScreen
+ dialogueController
- gameScreen
- controller
- multiplexer
- uiStage
- bag
- root
- dialogueBox
- optionBox
- topBar
- inventoryWrapper
and 14 more...
+ InventoryScreen()
+ show()
+ render()
+ resize()
+ pause()
+ update()
+ resume()
+ hide()
+ dispose()
+ setOptionBoxVisibility()
- initUI()
Screen
com.ungabunga.model.screen.BreederScreen
- app
- gameState
- gameScreen
- multiplexer
- controller
- dialogueController
- uiStage
- dialogueStage
- dialogueBox
- optionBox
and 19 more...
+ BreederScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ startBreeding()
+ stopBreeding()
+ setGameState()
+ getGameState()
+ selectA()
+ selectB()
+ show()
- initUI()
Screen
com.ungabunga.model.screen.ChildEngimonScreen
- app
- multiplexer
- controller
- dialogueController
- gameState
- gameScreen
- uiStage
- dialogueStage
- dialogueBox
- optionBox
and 15 more...
+ ChildEngimonScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ show()
- initUI()

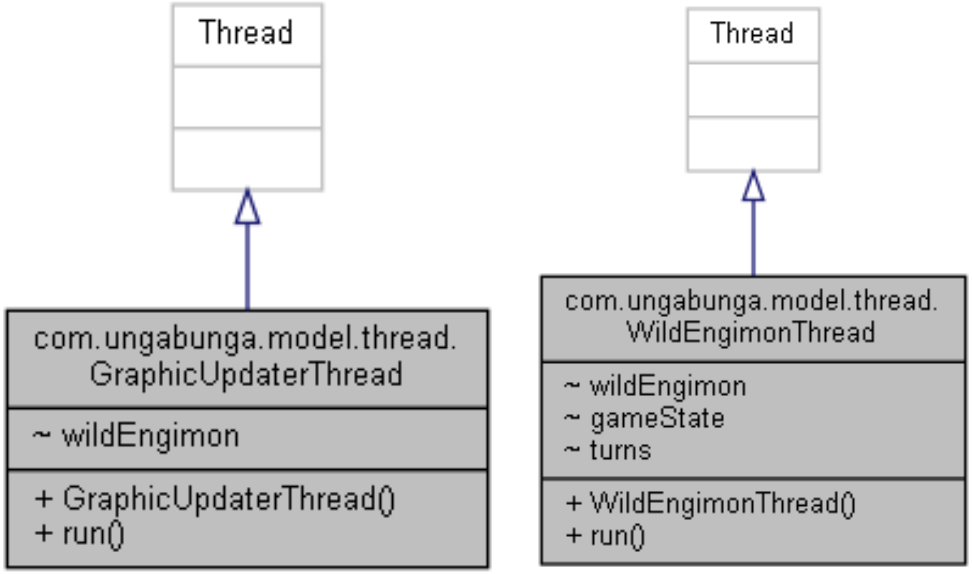

Screen
com.ungabunga.model.screen.
AbstractScreen
- app
+ AbstractScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ show()
+ getApp()
com.ungabunga.model.screen.
DetailEngimonScreen
- provider
- uiStage
- activeEngimon
- controller
- gameScreen
- root
- topBar
- engimonPic
- name
- details
and 8 more...
+ DetailEngimonScreen()
+ initUI()
+ show()
+ render()
+ resize()
+ pause()
+ update()
+ resume()
+ hide()
+ dispose()


Screen
com.ungabunga.model.screen.
GameOverScreen
- game
- batch
- heading
- splash
- playButtonActive
- playButtonInactive
- exitButtonActive
- exitButtonInactive
- loadButtonActive
- loadButtonInactive
- gameOver
- BUTTON_WIDTH
- BUTTON_HEIGHT
+ GameOverScreen()
+ show()
+ render()
+ resize()
+ pause()
+ resume()
+ hide()
+ dispose()


Screen
com.ungabunga.model.screen.
AbstractScreen
- app
+ AbstractScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ show()
+ getApp()
com.ungabunga.model.screen.
GameScreen
+ dialogueBox
+ optionBox
+ dialogueController
- gameState
- multiplexer
- controller
- batch
- HUDBatch
- BreederMenuInactive
- BreederMenuActive
- InventoryInactive
- InventoryActive
- map
and 7 more...
+ GameScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ setGameState()
+ getGameState()
+ resize()
+ resume()
+ show()
- initUI()


Screen
com.ungabunga.model.screen.
AbstractScreen
- app
+ AbstractScreen()
+ dispose()
+ hide()
+ pause()
+ update()
+ render()
+ resize()
+ resume()
+ show()
+ getApp()
com.ungabunga.model.screen.
InventoryScreen
+ dialogueController
- gameScreen
- controller
- multiplexer
- uiStage
- bag
- root
- dialogueBox
- optionBox
- topBar
- inventoryWrapper
and 14 more...
+ InventoryScreen()
+ show()
+ render()
+ resize()
+ pause()
+ update()
+ resume()
+ hide()
+ dispose()
+ setOptionBoxVisibility()
- initUI()
Screen
com.ungabunga.model.screen.
MainMenu
- game
- batch
- heading
- splash
- playButtonActive
- playButtonInactive
- exitButtonActive
- exitButtonInactive
- loadButtonActive
- loadButtonInactive
- BUTTON_WIDTH
- BUTTON_HEIGHT
+ MainMenu()
+ show()
+ render()
+ resize()
+ pause()
+ resume()
+ hide()
+ dispose()
Screen
com.ungabunga.model.screen.
Splash
- batch
- splash
- font
- text
- y
- alpha
- game
- openingTheme
+ Splash()
+ show()
+ render()
+ resize()
+ pause()
+ resume()
+ hide()
+ dispose()

1.9. Thread



1.10. User-Interface

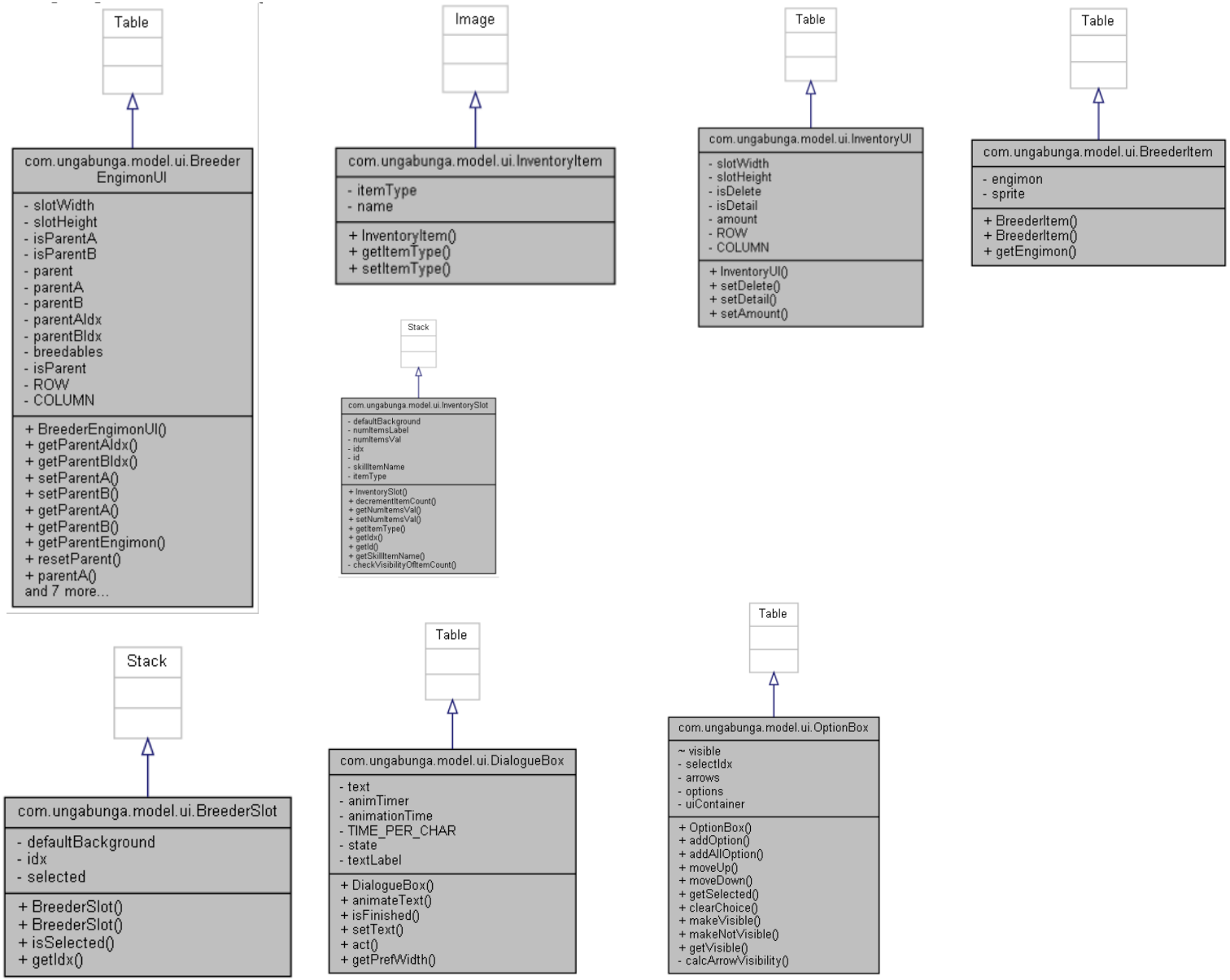
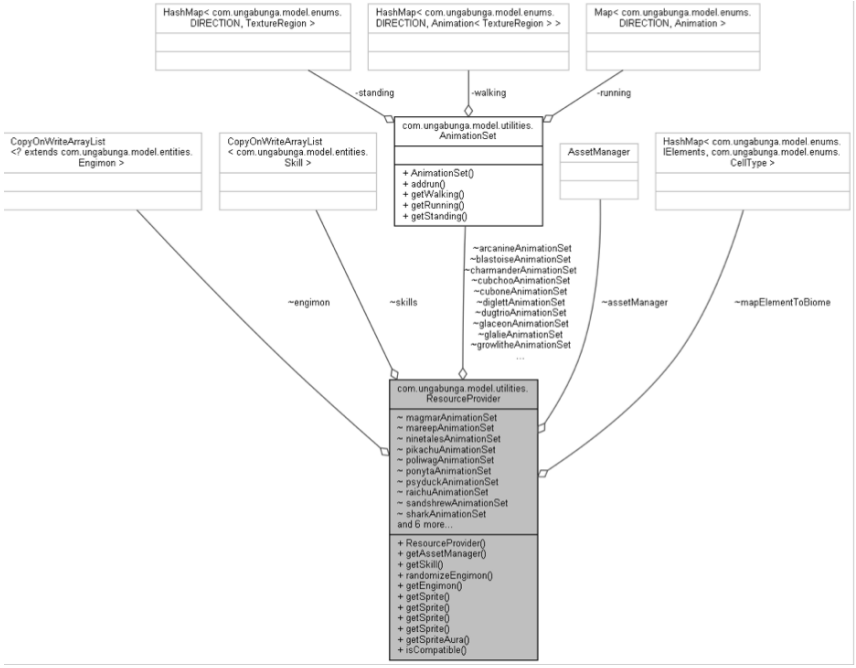
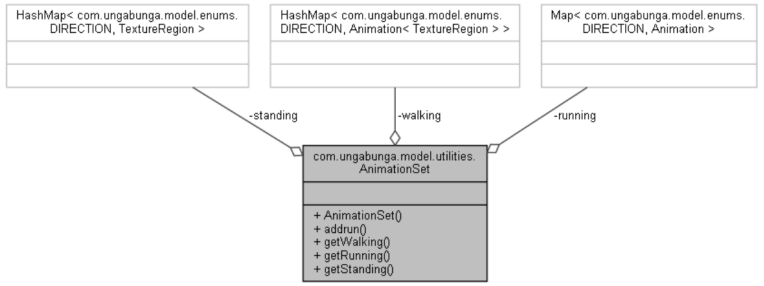
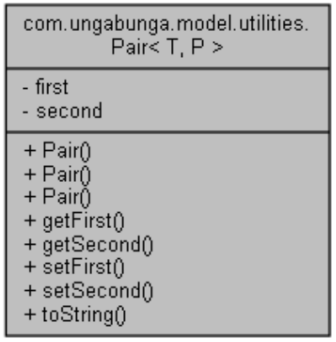


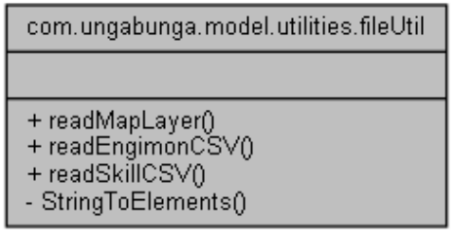
Table
com.ungabunga.model.ui.BreederEngimonUI
- slotWidth
- slotHeight
- isParentA
- isParentB
- parent
- parentA
- parentB
- parentAIdx
- parentBIdx
- breedables
- isParent
- ROW
- COLUMN
+ BreederEngimonUI()
+ getParentAIdx()
+ getParentBIdx()
+ setParentA()
+ setParentB()
+ getParentA()
+ getParentB()
+ getParentEngimon()
+ resetParent()
+ parentA()
and 7 more...
Image
com.ungabunga.model.ui.InventoryItem
- itemType
- name
+ InventoryItem()
+ getItemType()
+ setItemType()
Table
com.ungabunga.model.ui.InventoryUI
- slotWidth
- slotHeight
- isDelete
- isDetail
- amount
- ROW
- COLUMN
+ InventoryUI()
+ setDelete()
+ setDetail()
+ setAmount()
Table
com.ungabunga.model.ui.BreederItem
- engimon
- sprite
+ BreederItem()
+ BreederItem()
+ getEngimon()
Stack
com.ungabunga.model.ui.InventorySlot
Stack
com.ungabunga.model.ui.BreederSlot
- defaultBackground
- idx
- selected
+ BreederSlot()
+ BreederSlot()
+ isSelected()
+ getIdx()
Table
com.ungabunga.model.ui.DialogueBox
- text
- animTimer
- animationTime
- TIME_PER_CHAR
- state
- textLabel
+ DialogueBox()
+ animateText()
+ isFinished()
+ setText()
+ act()
+ getPrefWidth()
Table
com.ungabunga.model.ui.OptionBox
~ visible
- selectIdx
- arrows
- options
- uiContainer
+ OptionBox()
+ addOption()
+ addAllOption()
+ moveUp()
+ moveDown()
+ getSelected()
+ clearChoice()
+ makeVisible()
+ makeNotVisible()
+ getVisible()
- calcArrowVisibility()

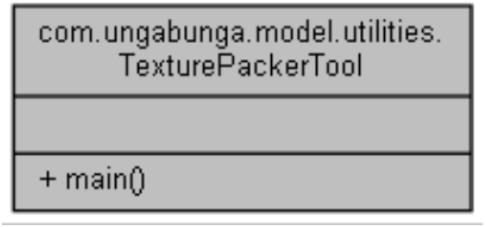
1.11. Utilities

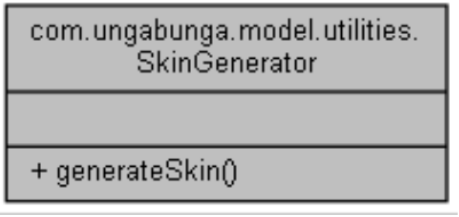








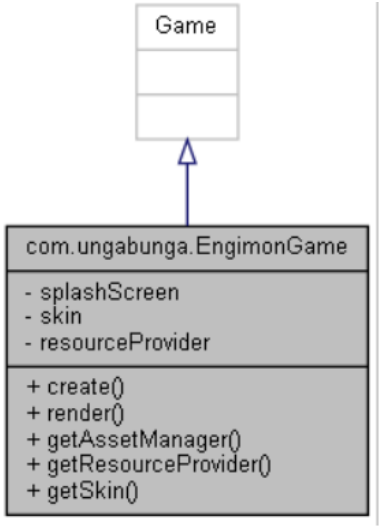




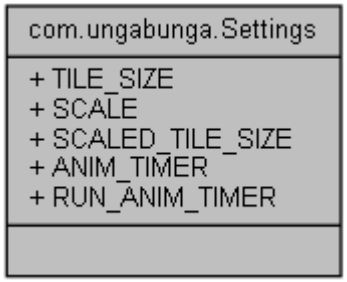
1.12. Game State

Dikarenakan Ukuran GameState yang terlalu besar maka gameState akan diletakan Terdapat pada dokumen doxygen pada halaman.

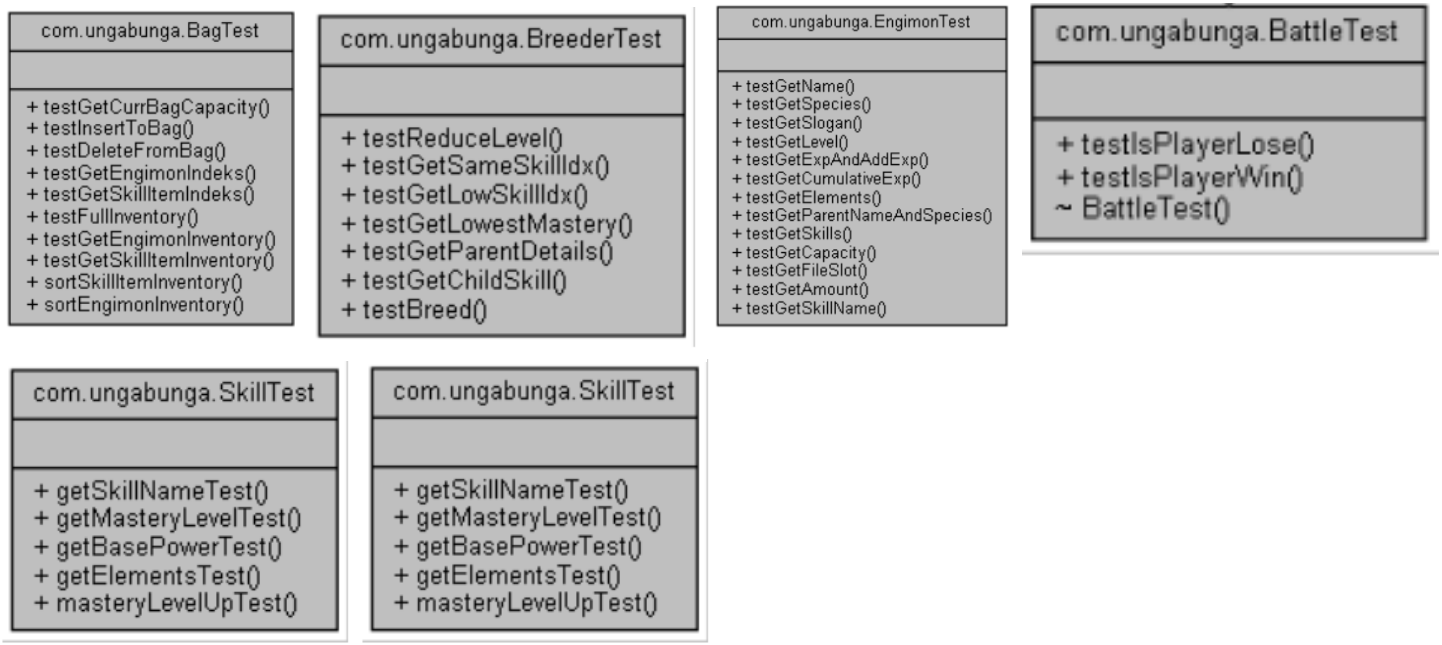
1.13. Engimon Game



1.14. Settings



1.15. Testing



Penggunaan Enum akan mempersingkat dan mempercepat pengerjaan kode dan waktu runtime, dan Pengkategorian Engimon cukup mempersulit pengolahan data data yang merupakan turunan atau implementasi dari Engimon (WildEngimon, LivingEngimon, Active Engimon), selain itu Enum juga sangat membantu dalam kenyamanan dan keterbacaan kode. Semua Diagram kelas diatas, didapatkan dengan bantuan aplikasi Doxygen dan untuk melihat Kelas dan Koneksinya dengan lebih lengkap, dapat dilihat pada tautan berikut.

2. Penerapan Konsep OOP

2.1. Polymorphism

Polymorphism digunakan dalam kelas Engimon dan turunannya, lebih tepatnya pada kelas PlayerEngimon dan ActiveEngimon. PlayerEngimon merupakan Engimon yang dimiliki oleh pemain dan disimpan di inventory, sedangkan ActiveEngimon merupakan Engimon yang sedang aktif. Polymorphism digunakan pada kedua kelas ini agar setiap Engimon dengan tipe tertentu dapat berperilaku dan memiliki atribut sesuai dengan tipe mereka. Misalnya, pada ActiveEngimon terdapat atribut yang menyimpan koordinat. Berikut merupakan cuplikan kode dari PlayerEngimon dan ActiveEngimon.

```
// PlayerEngimon.java
public class PlayerEngimon extends Engimon{
    int remainingLives;

    public PlayerEngimon(){

    }
```

```
    public PlayerEngimon(Engimon E){
        super(E);
        this.remainingLives = CONSTANTS.PLAYERENGIMONDEFAULTLIVES;
    }
}
```

```
// ActiveEngimon.java

public class ActiveEngimon extends PlayerEngimon implements LivingEngimon{
    Pair<Integer,Integer> position;
    DIRECTION direction;
    AVATAR_STATE state;

    Player player;
    GameState gameState;

    ResourceProvider resourceProvider;

    private boolean isRunning;

    private float srcX,srcY;
    private float destX,destY;
    private float worldX,worldY;
    private float animTimer;

    private float stateTimer;
```

```
    public ActiveEngimon(){

    }

    public ActiveEngimon(PlayerEngimon PE, Player P, int x, int y, GameState gameState, ResourceProvider
resourceProvider){
        super(PE);
        this.position = new Pair<Integer,Integer>(x,y);
        this.direction = P.getDirection();
        this.state = P.getState();


...
```

2.2. Inheritance/Composition/Aggregation

2.2.1. Inheritance

Inheritance digunakan dalam kelas Engimon beserta turunan - turunannya yakni ActiveEngimon, PlayerEngimon, dan WildEngimon. ActiveEngimon, PlayerEngimon, dan WildEngimon masing - masing memiliki sifat , tingkah laku, dan methodnya sendiri, akan tetapi masih memiliki sifat dasar yang dimiliki oleh Engimon. Alasan kami menggunakan konsep ini adalah karena masing - masing ActiveEngimon, PlayerEngimon, dan WildEngimon memiliki peran dan sifatnya masing - masing, tetapi tetap memiliki sifat dasar Engimon, contohnya PlayerEngimon yang memiliki life sebesar 3, sedangkan WildEngimon yang hanya memiliki 1 life, begitu pula ActiveEngimon merupakan PlayerEngimon yang sedang aktif sehingga bisa digunakan untuk battle.

```
public class WildEngimon extends Engimon implements LivingEngimon {
    Pair<Integer,Integer> position;

    ResourceProvider resourceProvider;
    GameState gameState;

    int remainingLives;

    public DIRECTION direction;
    AVATAR_STATE state;

    private float srcX,srcY;
    private float destX,destY;
    private float worldX,worldY;
    private float animTimer;

    private float stateTimer;
    public boolean isInBattle = false;
    public WildEngimon(){

    }

    public WildEngimon(Engimon E, int spawnX, int spawnY, ResourceProvider resourceProvider, GameState
gameState){
        super(E);
        this.remainingLives = CONSTANTS.WILDENGIMONDEFAULTLIVES;

        this.direction = DIRECTION.DOWN;
```

```
        this.state = AVATAR_STATE.STANDING;

        this.resourceProvider = resourceProvider;
        this.gameState = gameState;

        Pair<Integer,Integer> position = new Pair<>();
        position.setFirst(spawnX);
        position.setSecond(spawnY);

        this.position = position;

        this.worldX =  this.position.getFirst();
        this.worldY = this.position.getSecond();

        this.animTimer = 0f;
        this.stateTimer = 0f;
    }

...
```

2.2.2. Composition

Kami menggunakan konsep *Composition* dalam program kami, salah satunya seperti pada cuplikan kode di bawah, dimana class GameState yang menyimpan informasi mengenai state dari game memiliki atribut dengan *class* Player, EngimonGame, Bag, dll. yang merupakan class Inventory. Hal ini merupakan konsep *Composition* karena life cycle dari *class* Player, EngimonGame, Bag, dll. lifecycle class GameState, sehingga ketika GameState di-*destruct*, *class* Player, EngimonGame, Bag, dll. juga ikut ter-*destruct*.

```
public class GameState {
    public EngimonGame app;

    public Player player;
    public AtomicReferenceArray<AtomicReferenceArray<MapCell>> map;

    private Bag playerInventory;
    public DialogueBox dialogueBox;
    private float timeDelta;
    private float SPAWN_INTERVAL = 5.0f;

    private int wildEngimonCount;

...
```

2.2.3. Aggregation

Kami menggunakan konsep *Aggregation* dalam program kami, salah satunya seperti pada cuplikan kode di bawah, dimana class Engimon yang menyimpan informasi mengenai suatu engimon memiliki atribut skills yang merupakan class Skill. Hal ini merupakan konsep *Aggregation* karena life cycle dari class Skill tidak bergantung pada lifecycle class Engimon, sehingga ketika Engimon mati / di-*destruct*, class Skill yang dimilikinya tidak ikut ter-*destruct* karena bisa saja terdapat skill yang sama di Engimon lainnya.

```
public class Engimon {
    protected static int count = 0;
    protected int id;
    protected String name, species, slogan;
    protected int level, exp, cumulativeExp;
    protected List<IElements> elements;
    protected Pair<String, String> parentName, parentSpecies;
    protected List<Skill> skills;

    public Engimon(){

    }

    public Engimon(Engimon e) {
        this.id = e.id;
        this.name = e.name;
        this.species = e.species;
        this.slogan = e.slogan;
        this.elements = e.elements;
        this.skills = e.skills;
        this.level = e.level;
```

```
        this.parentName = e.parentName;
        this.parentSpecies = e.parentSpecies;
        this.exp = e.exp;
        this.cumulativeExp = e.cumulativeExp;
    }

...
```

2.3. Abstract Class

Kami menggunakan *abstract class* pada kelas AbstractScreen untuk meng-*set* screen game utama yang memiliki atribut dari kelas Game EngimonGame, sehingga kelas ini dapat di-*extend* di setiap screen yang ada dalam game kami.

```
public abstract class AbstractScreen implements Screen {

    private EngimonGame app;

    public AbstractScreen(EngimonGame app) {
        this.app = app;
    }


    @Override
    public abstract void dispose();

    @Override
```

```
    public abstract void hide();
      ...
```

2.4. Interface

Interface digunakan adalah interface LivingEngimon, yaitu Engimon yang masih hidup pada saat itu. Kelas yang mengimplementasikan LivingEngimon antara lain adalah kelas WildEngimon dan ActiveEngimon. WildEngimon merupakan engimon liar yang dapat melakukan *battle* dengan pemain, sedangkan ActiveEngimon merupakan Engimon yang sedang aktif. Berikut ini merupakan cuplikan dari kelas WildEngimon dan ActiveEngimon.

```java
// WildEngimon.java

public class WildEngimon extends Engimon implements LivingEngimon {
    Pair<Integer,Integer> position;

    ResourceProvider resourceProvider;
    GameState gameState;

    int remainingLives;

    DIRECTION direction;
    AVATAR_STATE state;

    private float srcX,srcY;
    private float destX,destY;
    private float worldX,worldY;
```

```
    private float animTimer;

    private float stateTimer;

    public WildEngimon(){

    }

    public WildEngimon(Engimon E, int spawnX, int spawnY, ResourceProvider resourceProvider, GameState gameState){
        super(E);
        this.remainingLives = CONSTANTS.WILDENGIMONDEFAULTLIVES;
```

```
// ActiveEngimon.java

public class ActiveEngimon extends PlayerEngimon implements LivingEngimon{
    Pair<Integer,Integer> position;
    DIRECTION direction;
    AVATAR_STATE state;

    Player player;
    GameState gameState;

    ResourceProvider resourceProvider;

    private boolean isRunning;
```

```
    private float srcX,srcY;
    private float destX,destY;
    private float worldX,worldY;
    private float animTimer;

    private float stateTimer;

    public ActiveEngimon(){

    }

    public ActiveEngimon(PlayerEngimon PE, Player P, int x, int y, GameState gameState, ResourceProvider
resourceProvider){
        super(PE);
        this.position = new Pair<Integer,Integer>(x,y);
        this.direction = P.getDirection();
        this.state = P.getState();
```

2.5. Generic Type & Wildcards

Inventory dapat menerima 2 kelas, yaitu Engimon dan SkillItem sehingga digunakan konsep Generic sehingga inventory dapat menerima keduanya, Karena Inventory dibagi menjadi 2 maka akan digunakan sebuah Kelas Bag yang menyimpan 2 inventory, yaitu inventory PlayerEngimon dan Inventory SkillItem.

```
// Implementasi Kelas Bag (Menyimpan 2 Inventory)
public class Bag {
    Inventory<SkillItem> skillItemInventory;
    Inventory<PlayerEngimon> engimonInventory;

    public Bag(){
        this.skillItemInventory = new Inventory<SkillItem>();
        this.engimonInventory = new Inventory<PlayerEngimon>();
    }
...
}
```

```
// Implementasi Kelas Inventory dengan memanfaatkan konsep Generic
public class Inventory<T> {
    ArrayList<T> items;
    Integer capacity;
    Integer neff;

    public Inventory(){
        this.capacity = CONSTANTS.INVENTORYCAPACITY;
```

```
        this.neff = 0;
        this.items = new ArrayList<T>();
    }
...
}
```

```
public class Pair<T, P> {
    private T first;
    private P second;

    public Pair(){
    }

    public Pair( T f, P s ) {
        first = f;
        second = s;
    }
    public Pair(Pair<T, P> pair) {
        first = pair.first;
        second = pair.second;
    }
    public T getFirst() { return first; }
    public P getSecond() { return second; }
    public void setFirst( T f ) { first = f; }
    public void setSecond( P s ) { second = s; }
```

```
    public String toString( ) { return "(" + first + ", " + second + ")"; }
}
```

2.6. Exception Handling

Exception digunakan di hampir semua bagian pada program agar tidak terjadi perilaku yang tidak diinginkan ketika terdapat error. Exception juga digunakan sehingga dapat ditampilkan *dialogue box* ketika terjadi suatu error, misalnya ketika pemain menabrak objek atau ketika *Bag* penuh. Berikut ini merupakan cuplikan dari beberapa kelas exception dan beberapa bagian yang menerapkannya.

```
public class CellOccupiedException extends Exception{
    public CellOccupiedException(String errMessage){
        super(errMessage);
    }
}

public class NoParentException extends Exception{
    public NoParentException(String errMessage){
        super(errMessage);
    }
}


public class SameParentException extends Exception{
    public SameParentException(String errMessage){
```

```
        super(errMessage);
    }
}
```

Penerapan exception di dalam kode program

```
if(direction == DIRECTION.UP && state!=AVATAR_STATE.STANDING){
            try{
                gameState.movePlayerUp();
            } catch(Exception e){
                gameScreen.dialogueController.startExceptionDialogue(e);
            }
        }

...

try {
                breedableEngimon.parentStatus();
                breedableEngimon.parentInfo();
                System.out.println(breedableEngimon.isParentFilled());
                if (breedableEngimon.isParentFilled()) {
```

```
                    if (!breedableEngimon.isParentSame()) {
                        ChildEngimonScreen childEngimonScreen = new ChildEngimonScreen(app, controller,
breedableEngimon.getParentA(), breedableEngimon.getParentB(), gameScreen, gameState);
                        app.setScreen(childEngimonScreen);
                        stopBreeding();
                    } else {
                        throw new SameParentException("Please choose two different Engimons!");
                    }
                } else {
                    throw new NoParentException("Please choose two Engimons to breed!");
                }
            } catch (Exception e) {
                System.out.println("Exception catched");
                dialogueController.startExceptionDialogue(e);
                ParentA.resetParent();
                ParentB.resetParent();
                stopBreeding();
            }

...
```

2.7. Java Collection

Java collection yang banyak digunakan pada pembuatan tugas ini merupakan List, lebih tepatnya adalah ArrayList karena ukurannya yang dinamis. Selain itu, pengaksesan yang mudah juga menjadi alasan digunakannya ArrayList. Berikut ini merupakan cuplikan beberapa kelas yang menggunakan ArrayList.

```
public class Inventory<T> {
    ArrayList<T> items;
    Integer capacity;
    Integer neff;

    public Inventory(){
        this.capacity = CONSTANTS.INVENTORYCAPACITY;
        this.neff = 0;
        this.items = new ArrayList<T>();
    }

…

public class Engimon {
    protected static int count = 0;
    ...
    protected List<IElements> elements;
    protected Pair<String, String> parentName, parentSpecies;
    protected List<Skill> skills;
```

```
    public Engimon(){

    }
…
```

2.8. Java API

Aggregation

Kami menggunakan konsep *Aggregation* dalam program kami, salah satunya seperti pada cuplikan kode di bawah, dimana class Engimon yang menyimpan informasi mengenai suatu engimon memiliki atribut skills yang merupakan class Skill. Hal ini merupakan konsep *Aggregation* karena life cycle dari class Skill tidak bergantung pada lifecycle class Engimon, sehingga ketika Engimon mati / di-*destruct*, class Skill yang dimilikinya tidak ikut ter-*destruct* karena bisa saja terdapat skill yang sama di Engimon lainnya.

```
public class Engimon {
    protected static int count = 0;
    protected int id;
    protected String name, species, slogan;
    protected int level, exp, cumulativeExp;
    protected List<IElements> elements;
    protected Pair<String, String> parentName, parentSpecies;
    protected List<Skill> skills;

    public Engimon(){
```

```
        }

        public Engimon(Engimon e) {
            this.id = e.id;
            this.name = e.name;
            this.species = e.species;
            ...
        }
```

Composition

Kami menggunakan konsep *Composition* dalam program kami, salah satunya seperti pada cuplikan kode di bawah, dimana class DialogueTranverser yang menyimpan informasi mengenai state dari game memiliki atribut `dialogueNode` yang merupakan class DialogueNode. Hal ini merupakan konsep *Composition* karena life cycle dari class DialogueNode bergantung pada lifecycle class DialogueTraverser , sehingga ketika DialogueTraverser di-*destruct*, class anaknya juga akan ter-*destruct*.

DialogueTraverser.java

```
public class DialogueTraverser {
    private Dialogue dialogue;
    private DialogueNode currNode;

    public DialogueTraverser(Dialogue dialogue) {
        this.dialogue = dialogue;
```

```
        this.currNode = dialogue.getNode(0);
    }
    public DialogueNode getNext(int ptrIdx) {
        DialogueNode next = dialogue.getNode(currNode.getPointers().get(ptrIdx));
        currNode = next;
        return next;
    }
...


DialogueNode.java
public class DialogueNode {
    private ArrayList<Integer> pointers = new ArrayList<Integer>();
    private ArrayList<String> labels = new ArrayList<String>();

    private String text;
    private int id;
    private NODE_TYPE type;

    public enum NODE_TYPE {
        MULT,
        LINEAR,
        END
    }
    public  DialogueNode(String text, int id) {
        this.text = text;
```

```
        this.id = id;
        this.type = NODE_TYPE.END;
    }
    public void addChoice(String opt, int NodeID) {
        if(this.type == NODE_TYPE.LINEAR) {
            this.pointers.clear();
        }
        this.labels.add(opt);
        this.pointers.add(NodeID);
        this.type = NODE_TYPE.MULT;
    }
...
```

3. Bonus Yang dikerjakan

3.1. Bonus yang diusulkan oleh spek

3.1.1. Multi-threading (real-time gameplay)

Ketika sebuah engimon liar dispawn ke map maka main thread akan melakukan spawn 2 buah thread. Thread pertama wildEngimonThread mengatur pergerakan engimon liar sambil menjaga aturan map yaitu tidak boleh ada 2 entitas yang menempati tempat yang sama. Thread ini juga akan menghandle apabila engimon mati maka akan langsung diremove dari map. Cara kerja WildEngimonThread adalah thread akan selalu melakukan looping untuk menentukan langkah yang akan diambil oleh engimon kemudian thread akan ditunda antara 2000ms hingga 5000ms. Setiap 3 turns, wildEngimon akan ditambah expnya sebanyak 25. Thread akan mati ketika seluruh blok kode di dalam run selesai dijalankan yaitu sesaat setelah wildEngimon mati

```
public class WildEngimonThread extends Thread{
    WildEngimon wildEngimon;
    GameState gameState;
    int turns;
    public WildEngimonThread(WildEngimon wildEngimon, GameState gameState){
        this.wildEngimon = wildEngimon;
        this.gameState = gameState;

        this.turns = 0;
    }
    public void run(){
        while(!wildEngimon.isDead()){
            if(!this.wildEngimon.isInBattle) {
```

```
                wildEngimon.randomizeMove();
            }
            this.turns++;
            if(turns % 3 ==0){
                wildEngimon.addExp(25);
            }
            try {
                Thread.sleep(ThreadLocalRandom.current().nextInt(2000,5000));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        wildEngimon.killEngimon();
        gameState.reduceWildEngimon();
...
```

Thread kedua adalah GraphicUpdaterThread, thread ini bertugas untuk menghaluskan pergerakan wildEngimon. Cara menghaluskan animasi adalah dengan mengoper nilai delta kepada childThread. Delta ini akan dipakai untuk menginterpolasi animasi engimon agar animasinya tidak patah-patah/ bergerak secara instan. Pada childThread tidak dilakukan rendering karena rendering OpenGL hanya bisa dilakukan oleh main thread namun hal ini tidak masalah karena ada attribute worldX dan worldY pada wild engimon.

```
public class GraphicUpdaterThread extends Thread{

    WildEngimon wildEngimon;
    public GraphicUpdaterThread(WildEngimon wildEngimon){
```

```
        this.wildEngimon = wildEngimon;
    }
    public void run(){
        while(!wildEngimon.isDead()){
            try{
                wildEngimon.update(Gdx.graphics.getDeltaTime());
                Thread.sleep(10);
            } catch (EngimonConflictException | InterruptedException e){
                e.printStackTrace();
            }
        }
    }
}
```

3.1.2. Unit Testing Implementation

Kami menggunakan JUnit untuk membantu implementasi unit testing untuk memastikan modul yang dibuat telah berjalan baik. Adapun testing kami lakukan terhadap method - method dalam modul yang sering digunakna dalam game. Berikut ini merupakan cuplikan kode dari unit testing untuk modul Battle.

```
public class BattleTest {
    @Test
    public void testGetHitungPower(){
        List<IElements> engimonElements = new ArrayList<>();
        List<IElements> skillElements = new ArrayList<>();
        List<Skill> skills = new ArrayList<>();
```

```
        Pair<String, String> parentName = new Pair("Parent A","Parent B");
        Pair<String, String> parentSpecies = new Pair("Species A","Species B");

        engimonElements.add(IElements.ELECTRIC);
        skillElements.add(IElements.ELECTRIC);

        Skill skill = new Skill("Capacitor", skillElements, 6, 1);
        skills.add(skill);

        Engimon engimon = new Engimon("NamaPokemon", "Jolteon", "Keep the energy", 1,
engimonElements, skills, parentName, parentSpecies);
        Assertions.assertEquals(1, engimon.getLevel());
        Assertions.assertEquals(6, engimon.getSkills().get(0).getBasePower());

...
```

3.2. Bonus Kreasi Mandiri

3.2.1. Background Music

Pada pembuatan tugas ini kami menambahkan *background music* pada permainan. Terdapat dua musik yang berbeda yaitu musik pada menu awal dan musik ketika permainan sudah dimulai

```
public class Splash implements Screen {
    private SpriteBatch batch;
    private Sprite splash;
    private BitmapFont font;
    private GlyphLayout text;
    private float y;
    private float alpha = 0;

    private EngimonGame game;

    private Music openingTheme;

    public Splash(EngimonGame game) {
        this.game = game;
    }

    @Override
    public void show() {
        batch = new SpriteBatch();

        Texture splashTexture = new Texture("Logo.png");
```

```
        splash = new Sprite(splashTexture);
        font = new BitmapFont(Gdx.files.internal("font/white.fnt"));
        text = new GlyphLayout(font, "PRESS SPACE TO CONTINUE!!!");
        splash.setSize(Gdx.graphics.getWidth(), Gdx.graphics.getHeight());
        this.openingTheme = Gdx.audio.newMusic(Gdx.files.internal("song/OpeningTheme.ogg"));

...
```

3.2.2. Animation

Kami menambahkan animasi berjalan dan berlari pada karakter player dan engimon. Animasi dibuat dengan memanfaatkan fungsi interpolasi dan delta

```
public class AnimationSet {

    private HashMap<DIRECTION, Animation<TextureRegion>> walking;
    private HashMap<DIRECTION, TextureRegion> standing;
    private Map<DIRECTION, Animation> running;

    public AnimationSet(Animation walkDown, Animation walkUp, Animation walkLeft, Animation
walkRight, TextureRegion standDown, TextureRegion standUp, TextureRegion standLeft,
TextureRegion standRight){
        walking = new HashMap<DIRECTION,Animation<TextureRegion>>();
        walking.put(DIRECTION.UP,walkUp);
        walking.put(DIRECTION.DOWN,walkDown);
        walking.put(DIRECTION.LEFT,walkLeft);
```

```
        walking.put(DIRECTION.RIGHT,walkRight);

        standing = new HashMap<DIRECTION, TextureRegion>();
        standing.put(DIRECTION.UP,standUp);
        standing.put(DIRECTION.DOWN, standDown);
        standing.put(DIRECTION.LEFT, standLeft);
        standing.put(DIRECTION.RIGHT, standRight);


    }

    public void addrun(Animation runUp, Animation runDown, Animation runLeft, Animation
runRight) {
        running = new HashMap<DIRECTION, Animation>();
        running.put(DIRECTION.UP, runUp);
        running.put(DIRECTION.DOWN, runDown);
        running.put(DIRECTION.LEFT, runLeft);
        running.put(DIRECTION.RIGHT, runRight);
    }

    public Animation<TextureRegion> getWalking(DIRECTION dir){
        return walking.get(dir);
    }

    public Animation<TextureRegion> getRunning(DIRECTION dir){
        return running.get(dir);
    }
```

```
    public TextureRegion getStanding(DIRECTION dir){
        return standing.get(dir);
    }

    public void update(float delta) throws EngimonConflictException {
        if(activeEngimon!=null){
            activeEngimon.update(delta);
        }
        float anime_time = 0f;
        animTimer += delta;
        stateTimer += delta;

        if(state == AVATAR_STATE.WALKING && !isRunning) {
            anime_time = ANIM_TIMER;
        } else if (state == AVATAR_STATE.WALKING && isRunning) {
            anime_time = RUN_ANIM_TIMER;
        }
        worldX = Interpolation.pow2.apply(this.srcX,this.destX,animTimer/anime_time);
        worldY = Interpolation.pow2.apply(this.srcY,this.destY,animTimer/anime_time);
        if(animTimer > anime_time){
            stateTimer -= (animTimer - anime_time);
            finishMove();
            if(moveFrameRequest){
                if(direction == DIRECTION.UP){
                    moveUp();
                }
                if(direction == DIRECTION.DOWN){
                    moveDown();
```

```
                }
                if(direction == DIRECTION.LEFT){
                    moveLeft();
                }
                if(direction == DIRECTION.RIGHT){
                    moveRight();
                }
            } else{
                this.isRunning = false;
                stateTimer = 0f;
            }
        }

        moveFrameRequest = false;
    }

}
public void update(float delta) throws EngimonConflictException {
        animTimer += delta;
        stateTimer += delta;

        worldX = Interpolation.pow2.apply(this.srcX,this.destX,animTimer/ANIM_TIMER);
        worldY = Interpolation.pow2.apply(this.srcY,this.destY,animTimer/ANIM_TIMER);
        if(animTimer > ANIM_TIMER){
            stateTimer -= (animTimer - ANIM_TIMER);
            finishMove();
            stateTimer = 0f;
```

```
        }
    }
```

3.2.3. Run Mode

Pada permainan ini terdapat mode berlari sehingga pemain bisa bergerak lebih cepat. Cara mengaktifkannya adalah dengan cara menahan tombol *Shift* setelah tombol bergerak yaitu W, A, S atau D, activeEngimon yang mengikuti akan ikut berlari sesuai dengan kecepatan player.

```
...
@Override
    public boolean keyUp(int keycode){
        if(keycode == Keys.W) {
            state = AVATAR_STATE.STANDING;
        }
        if(keycode == Keys.S) {
            state = AVATAR_STATE.STANDING;
        }
        if(keycode == Keys.A) {
            state = AVATAR_STATE.STANDING;
        }
        if(keycode == Keys.D) {
            state = AVATAR_STATE.STANDING;
        }
        if (keycode == Keys.SHIFT_LEFT && gameState.player.state == AVATAR_STATE.WALKING) {
```

```
            gameState.player.isRunning= false;
        }
        return false;
    }
...
```

3.2.4. Dialogue and Option Box

Pada permainan ini, ketika pemain mencoba untuk berjalan ke arah suatu objek pada peta, misalnya bangunan, pohon, engimon liar, atau bahkan engimon aktif pemain, pemain tidak akan bisa bergerak dan akan ditampilkan *dialogue box* yang menampilkan pesan bahwa *cell* telah terisi sehingga pemain harus melewati jalur lain. Selain itu, pemain juga diberikan informasi jika inventory penuh melalui dialogue box yang ditampilkan pada UI. Player juga akan diberikan exception untuk berhenti jika saat sedang berjalan/berlari saat ingin membuka inventory. Selain itu terdapat tutorial dan juga menerapkan sistem interactive story game.

```
...

   public void startTutorialDialogue(){
        dialogState = DIALOG_STATE.ELSE;
        Dialogue dialogue = new Dialogue();

        DialogueNode a = new DialogueNode("Welcome to Engimon, Curse of The Marcello Pokemon
God" + "\nPress Enter to close this message",0);
        DialogueNode b = new DialogueNode("Use UP and DOWN arrow to select choices, would you
like to skip the tutorial?", 1);
        DialogueNode c = new DialogueNode("To walk, use W A S D." +
```

```
                "\nPress H to restart the tutorial. " +
                "\nPress B for Battle", 2);
        DialogueNode c1 = new DialogueNode("Press I to open your inventory"+
                "\nPress R to remove active engimon"+
                "\nPress shift to walk faster",3);
        DialogueNode d = new DialogueNode("Press F5 to save the game"+
                "\nYou can breed and open the inventory by"+
                "\nclicking the icons at the top left of the screen",4);
        DialogueNode e = new DialogueNode("Enjoy the game!", 5);

        a.makeLinear(b.getId());
        c.makeLinear(c1.getId());
        c1.makeLinear(d.getId());
        b.addChoice("Yes",5);
        b.addChoice("No",2);

        dialogue.addNode(a);
        dialogue.addNode(b);
        dialogue.addNode(c);
        dialogue.addNode(c1);
        dialogue.addNode(d);
        dialogue.addNode(e);

        startDialogue(dialogue);
    }
...

...
```

```
   public void startExceptionDialogue(Exception e){
        dialogState = DIALOG_STATE.ELSE;
        Dialogue dialogue = new Dialogue();
        DialogueNode a = new DialogueNode(e.getMessage(), 0);
        dialogue.addNode(a);
        Obox.setVisible(false);
        startDialogue(dialogue);
    }
...
```

4. External Library

4.1. LibGDX

LibGDX merupakan *game engine* yang digunakan dalam pembuatan GUI tugas ini. Kami menggunakan LibGDX agar pembuatan UI bisa menjadi sedikit lebih mudah serta karena merupakan *game engine* sehingga nantinya akan tersedia tools yang dibutuhkan dalam pembuatan suatu permainan.

4.2. JUnit

JUnit merupakan sebuah framework unit testing yang disediakan khusus untuk pemrograman berbahasa Java. JUnit digunakan sebagai wadah untuk mengetes program dan merupakan salah satu kelompok dari framework unit testing yang paling sering digunakan. JUnit memberikan sebuah ide untuk mengetes sebelum mengoding sehingga kita mampu mengetes sebuah potongan kode terlebih dahulu sebelum di implementasikan.

5. Pembagian Tugas

Modul (dalam poin spek)	Designer	Implementer
Engimon	13519059	13519059
LivingEngimon	13519079	13519079
WildEngimon	13519079	13519079,13519086
PlayerEngimon	13519059	13519059
ActiveEngimon	13519079	13519079
Skill	13519104	13519059, 13519104
SkillItem	13519104	13519059,13519104
Player	13519086	13519086
Bag	13519090	13519059, 13519079, 13519090
Inventory	13519090	13519059, 13519079, 13519090
Battle	13519086	13519086
Breeder	13519104	13519104
MapCell	13519079	13519079
Save dan Load	13519079	13519079

Unit Testing	13519079	13519059, 13519068,13519086, 13519090,13519104