ALIENOID

Kornchanok Jiravesayakul 5631001721
Vasvaroos Ngamdamrongkiat 5631086021

ALIENOID

PROBLEM STATEMENT: ALIENOID GAME

The concept of Alienoid is to stop the construction of the spaceship. Hence the goal is to survive as long as possible.

The gameplay of Alienoid mixes action and management of resources: the magic powers. The player has the powers that she/he may send at a chosen location to eliminate the aliens that are going to build the spaceship, the only limitation being the recharging time for each power.

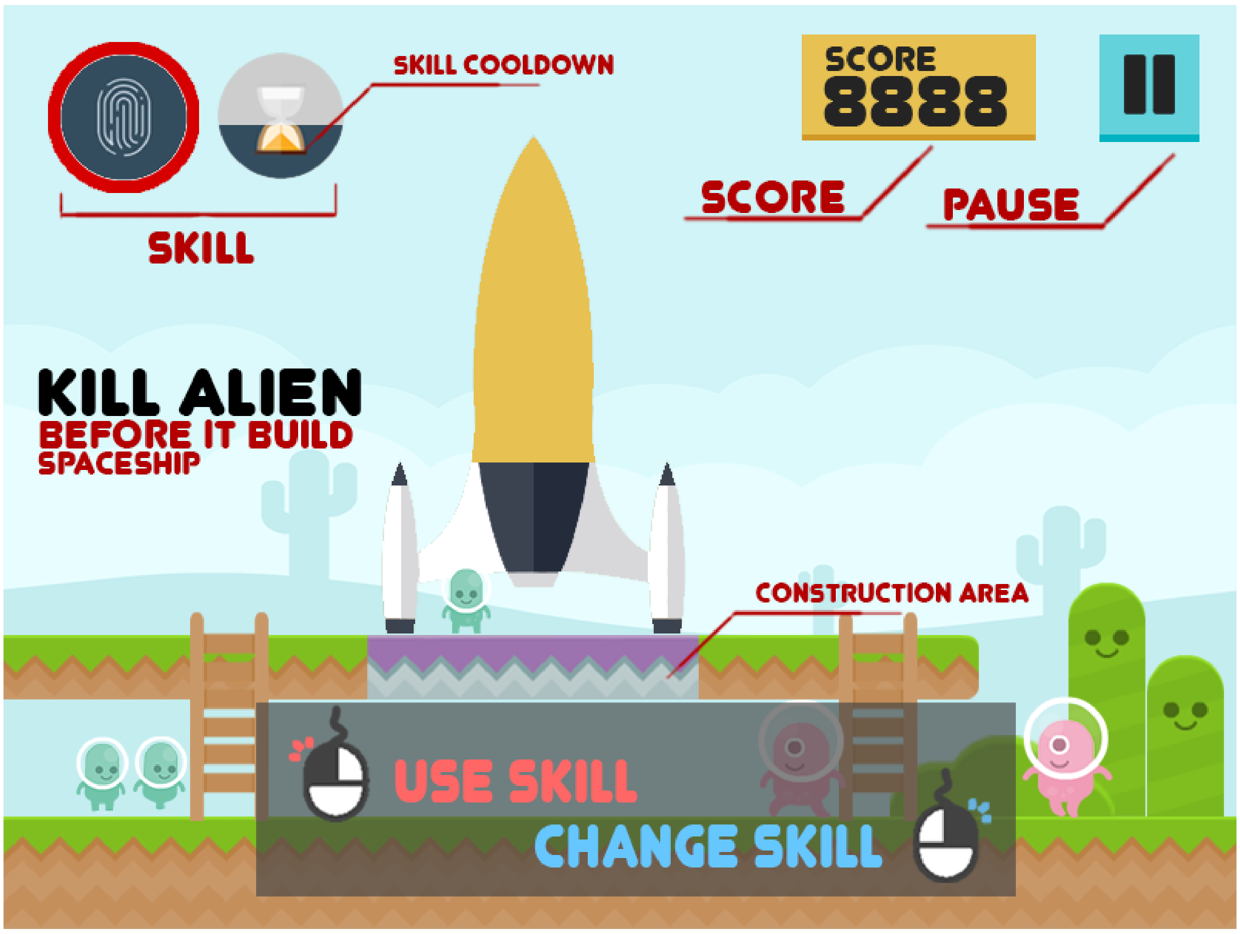


Figure 1 The game tutorial screenshot

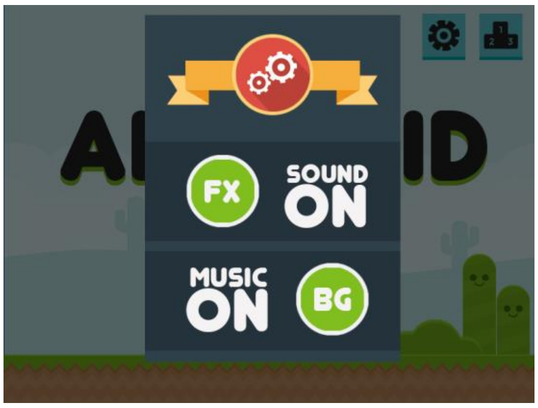


Figure 2 Setting page

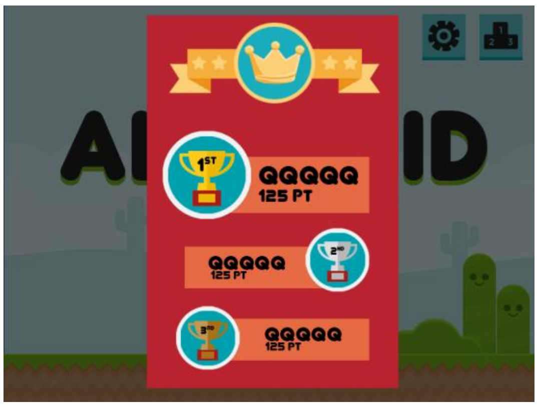


Figure 3 Highscore page

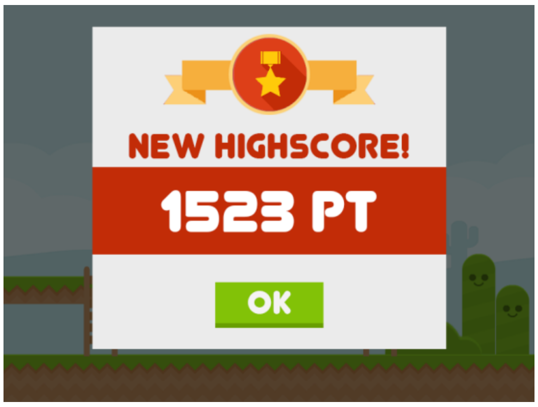


Figure 4 New highscore

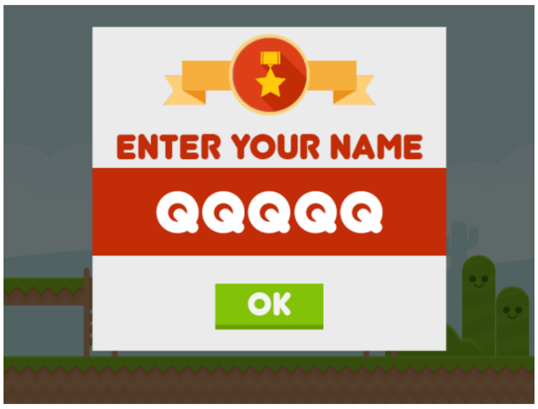


Figure 5 New highscore

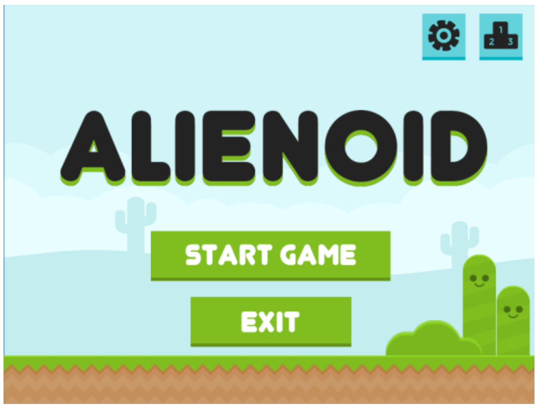


Figure 6 Game Title page

UML DIAGRAM

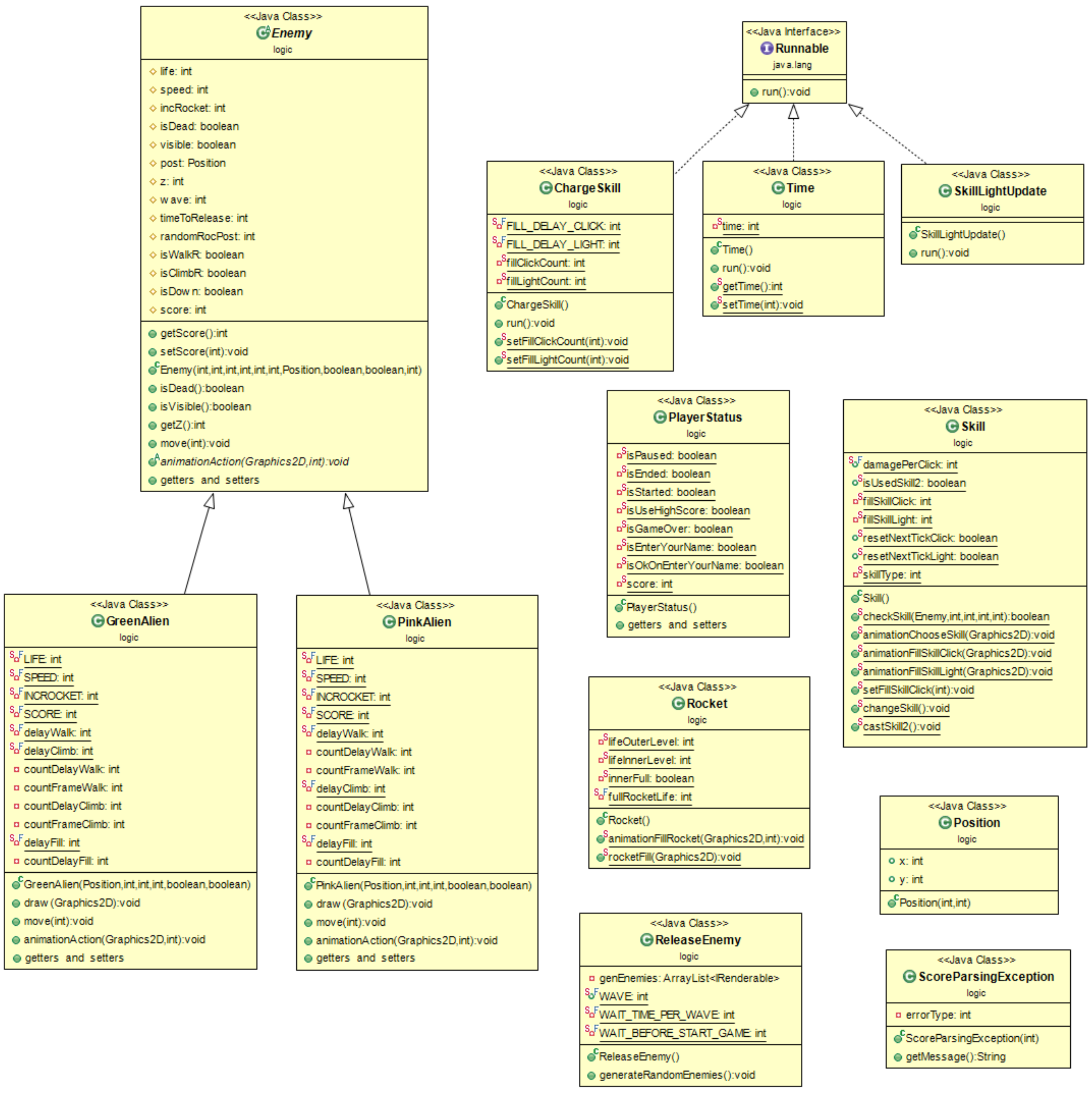
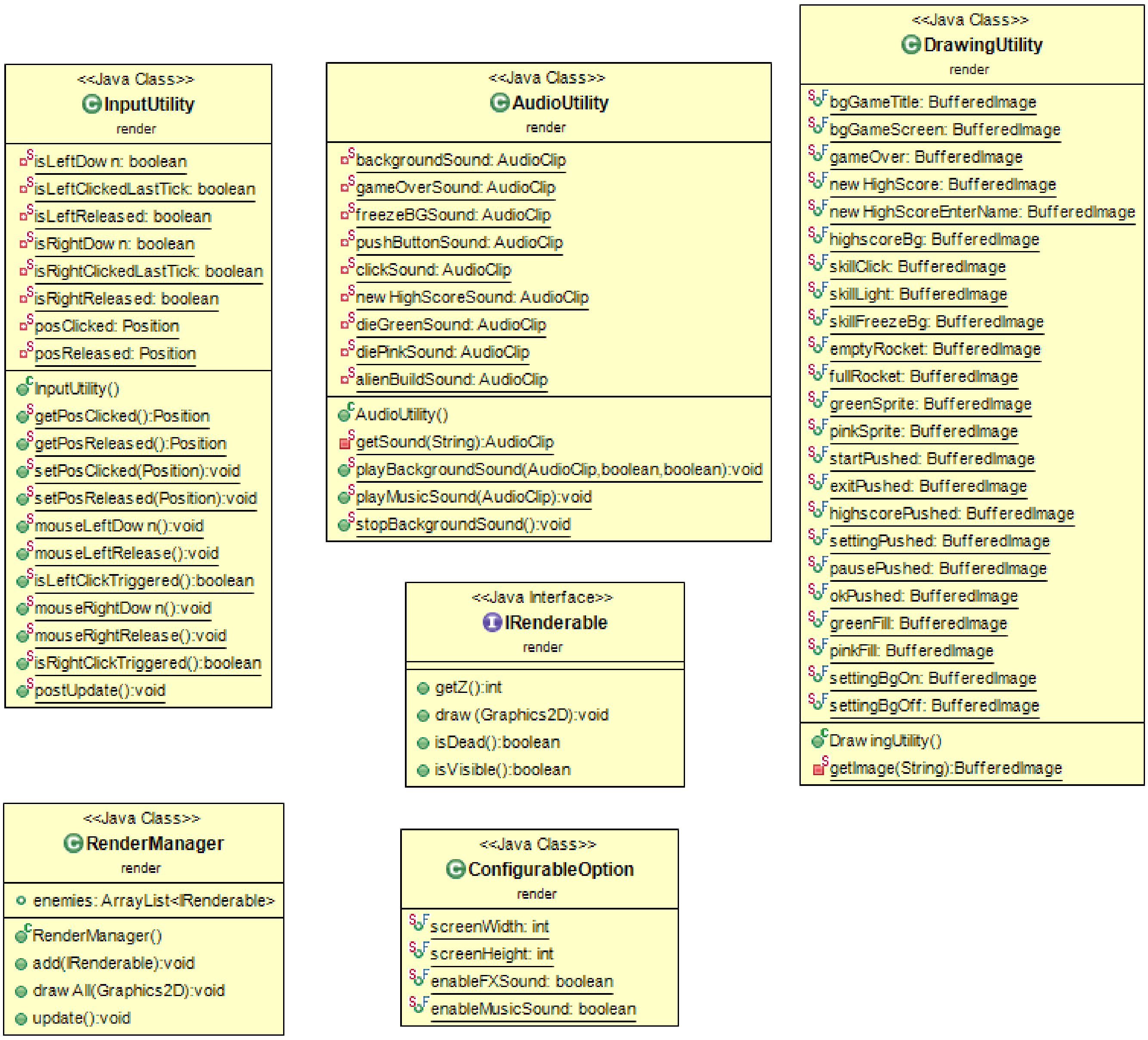


Figure 7 Structure of classes in Package "logic"

UML DIAGRAM



<<Java Class>>
InputUtility
render
isLeftDown: boolean
isLeftClickedLastTick: boolean
isLeftReleased: boolean
isRightDown: boolean
isRightClickedLastTick: boolean
isRightReleased: boolean
posClicked: Position
posReleased: Position
InputUtility()
getPosClicked():Position
getPosReleased():Position
setPosClicked(Position):void
setPosReleased(Position):void
mouseLeftDown():void
mouseLeftRelease():void
isLeftClickTriggered():boolean
mouseRightDown():void
mouseRightRelease():void
isRightClickTriggered():boolean
postUpdate():void
<<Java Class>>
AudioUtility
render
backgroundSound: AudioClip
gameOverSound: AudioClip
freezeBGSound: AudioClip
pushButtonSound: AudioClip
clickSound: AudioClip
newHighScoreSound: AudioClip
dieGreenSound: AudioClip
diePinkSound: AudioClip
alienBuildSound: AudioClip
AudioUtility()
getSound(String):AudioClip
playBackgroundSound(AudioClip,boolean,boolean):void
playMusicSound(AudioClip):void
stopBackgroundSound():void
<<Java Class>>
DrawingUtility
render
bgGameTitle: BufferedImage
bgGameScreen: BufferedImage
gameOver: BufferedImage
newHighScore: BufferedImage
newHighScoreEnterName: BufferedImage
highscoreBg: BufferedImage
skillClick: BufferedImage
skillLight: BufferedImage
skillFreezeBg: BufferedImage
emptyRocket: BufferedImage
fullRocket: BufferedImage
greenSprite: BufferedImage
pinkSprite: BufferedImage
startPushed: BufferedImage
exitPushed: BufferedImage
highscorePushed: BufferedImage
settingPushed: BufferedImage
pausePushed: BufferedImage
okPushed: BufferedImage
greenFill: BufferedImage
pinkFill: BufferedImage
settingBgOn: BufferedImage
settingBgOff: BufferedImage
DrawingUtility()
getImage(String):BufferedImage
<<Java Interface>>
IRenderable
render
getZ():int
draw(Graphics2D):void
isDead():boolean
isVisible():boolean
<<Java Class>>
RenderManager
render
enemies: ArrayList<IRenderable>
RenderManager()
add(IRenderable):void
drawAll(Graphics2D):void
update():void
<<Java Class>>
ConfigurableOption
render
screenWidth: int
screenHeight: int
enableFXSound: boolean
enableMusicSound: boolean

Figure 8 Structure of classes in Package "render"

UML DIAGRAM

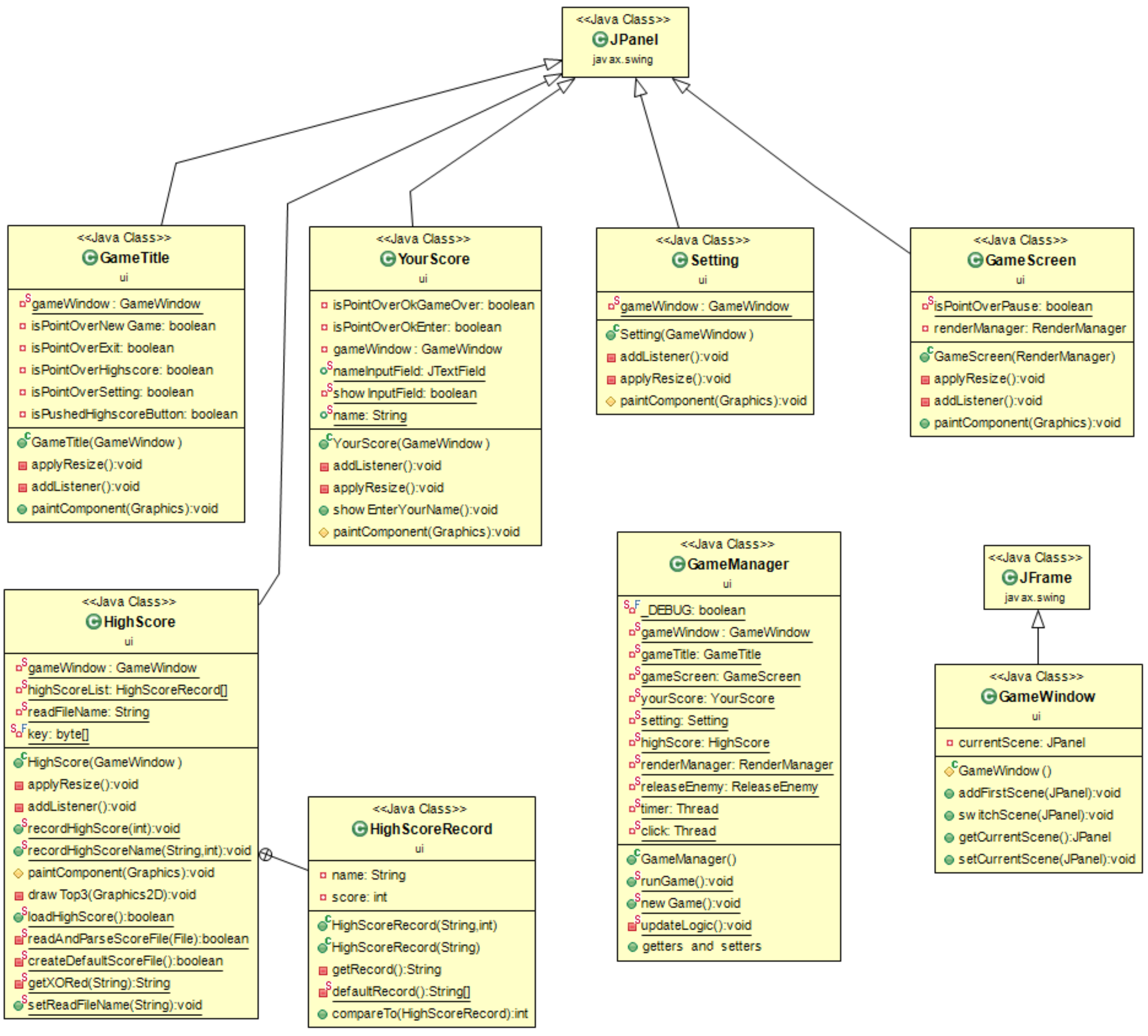


Figure 9 Structure of classes in Package "ui"

IMPLEMENTING THE ALIENOID GAME

1. PACKAGE 'logic'

1.1 Class "logic.Enemy"

This is an abstract class, which represents all of the enemies in the field and the method called when the enemies do actions.

Field

- ***int life, speed, incRocket, score;*** The life point, walking speed of enemy, the increasing volumn of the rocket that this enemy can increase, and score that get after killing this enemy.
- ***boolean isDead, visible;*** isDead is "true" when the alien was dead**,** isVisible is true when the alien is visible (not yet dead).
- ***Position post;*** The position of the alien.
- ***int z;*** return the z-axis position
- ***int wave, timeToRelease, randomRocPost;*** The releasing alien wave, time to release this alien, the random position of alien while building the rocket.
- ***boolean isWalkR, isClimbR, isDown;*** The status of alien that show this direction <walk to left(0)/right(1) side>, the ladder that the alien will climb <left(0)/right(1)>, and The level which alien is standing <upper level = true, lower level = false>.

Constructor

- ***Enemy(int life, int speed, int incRocket, int z, int wave, int time_release, Position post,boolean isWalkR,boolean isClimbR,int score);***
  It initializes the value of the alien**.**

Method

- ***synchronized int getScore();*** It returns score of this alien.
- ***synchronized void setScore(int score);*** It sets score of this alien.
- ***boolean isDead();*** It returns true when the alien is dead.
- ***boolean isVisible();*** It returns true when the alien is visible (not yet dead).
- ***int getZ();*** It returns index of object in z-axis.
- ***void move(int mode);***
  - Change the position of the enemy (move the enemy) with 'speed' unit.
  - Every tick the enemy will move ('speed') unit when it walk, and it will move ('speed')/2 unit when it climb the ladder.
  - mode=1 : Walk Left mode=2 : Walk right
    mode=3 : Climb up
- ***abstract void animationAction(Graphics2D g2d, int mode);***
  - Drawing the animation of alien actions. This method will be called from paintComponent() method
  - mode=1 : Walk Left mode=2 : Walk right
    mode=3 : Climb up mode=4 : Die
    mode=5 : Build the spaceship

1.2 Class "logic.GreenAlien"

Green alien is a small and fast enemy but can build low unit. It's has 1 life point, which can be killed in 1 click.

Field

- ***static final int LIFE=1, SPEED=2, INCROCKET=100, SCORE=2;*** The default value of 'life', 'speed', 'incRocket', 'score' variables.

- ***static final int delayWalk***=6 ,***delayClimb***=8, ***delayFill***=30, ***countDelayWalk***=0, ***countFrameWalk***=0, ***countDelayClimb***=0, ***countFrameClimb***=0, ***countDelayFill***=0;
  The number of frames (sub images) and an interval (delay) time for showing each sub image.

Constructor

- ***GreenAlien(Position post, int z, int time_release, int wave, boolean isWalkR, boolean isClimbR);***

  It initializes the value of the alien.

Method

- ***Getter & Setter;***
- ***void animationAction(Graphics2D g2d, int mode);*** Implement the 'animationAction' from 'Enemy' class.

1.3 Class "logic.PinkAlien"

Pink alien is a big and slow enemy but can build huge unit. It's has 2 life point, which can be killed in 2 click.

Field

- ***static final int LIFE=2, SPEED=1, INCROCKET=150, SCORE=5;***
- ***static final int delayWalk***=10, ***countDelayWalk***=0, ***countFrameWalk***=0, ***delayClimb***=14, ***countDelayClimb***=0, ***countFrameClimb***=0, ***delayFill***=30, ***countDelayFill***=0;
  The number of frames (sub images) and an interval (delay) time for showing each sub image.

Constructor

- ***PinkAlien(Position post, int z, int time_release, int wave, boolean isWalkR, boolean isClimbR);***

  It initializes the value of the alien.

Method

- ***Getter & Setter;***
- ***void animationAction(Graphics2D g2d, int mode);*** Implement the 'animationAction' from 'Enemy' class.

1.4 Class "logic.PlayerStatus"

Represent the player status and game status.

Field

- ***static boolean isPaused, isEnded, isStarted;*** The status of the game: pause, end, start.
- ***static boolean isUseHighScore;*** It's TRUE when the player in highscore scene.
- ***static boolean isGameOver;*** It's TRUE when the game is over.
- ***static boolean isEnterYourName;*** It's TRUE when the player in rank and in 'GameOver' scene.
- ***static boolean isOkOnEnterYourName;*** It's TRUE when the player in push 'OK' in 'GameOver' scene.
- ***static int score;*** It's a score of player.

Method

- ***Getter & Setter;***

1.5 Class "logic.Position"

Represent the x, y coordination.

Field

- ***int x, y;*** Contain the coordination (x,y)

1.6 Class "logic.ReleaseEnemy"

This class generates the enemies releasing to the game field.

Field

- ***ArrayList<IRenderable> genEnemies;*** Contain all of enemies in the game that will be releasing by the 'release_time'.
- **static final int WAVE;** The number of enemies waves in a game. The default value is 1000.
- **static final int WAIT_TIME_PER_WAVE;** The delay time between each wave. The default value is 50.
- **static final int WAIT_BEFORE_START_GAME;** The delay time before the first wave is releasing. The default value is 10.

Constructor

- ***ReleaseEnemy();*** Initial and generate the enemies randomly.

Method

- ***Getter & Setter;***
- ***void generateRandomEnemies();*** Generating the type, the born location, and the releasing time of each enemy. At first, the number of enemies per wave has a small number, then it will be increasing respectively.

1.7 Class "logic.Rocket"

Rocket class store the progression of the rocket construction. The rocket has 2 layers: yellow layers and outer layer. The alien has to build the inner layer (yellow layer) first, then the outer layer. The rocket is finish after these two level are built.

Field

- ***static int lifeInnerLevel;*** The height of the inner rocket that the alien built.
- ***static int lifeOuterLevel;*** The height of the outer rocket that the alien built.
- ***static boolean innerFull;*** Its value is TRUE when lifeInnerLevel equals to LIFE. Default value is FALSE.
- ***static final int fullRocketLife;*** The full life of the rocket.

Method

- ***Getter & Setter;***
- **static void animationFillRocket(Graphics2D g2d, int fillCount);** The animation of the alien when built rocket.
- **static void rocketFill(Graphics2D g2d);** The animation of the rocket that was built.

1.8 Class "logic. ScoreParsingException"

Field

- ***int errorType;*** The type of the occurred error.

Constructor

- ***ScoreParsingException(int errorType);*** It initializes the error type.

Method

- ***String getMessage();*** It returns an exception message. If it is errorType=0, the return text is "No record score". If it is errorType=1, the return text is "Wrong record format".

1.9 Class "logic. Skill"

Field

- ***static final int damagePerClick;*** It's the damage per click (the value is always 1).
- ***static boolean isUsedSkill2;*** It's TRUE when the player uses skill2.
- ***static int fillSkillClick, fillSkillLight;*** The delay of tick for cooldown skill.

  static boolean resetNextTickClick, resetNextTickLight; It's TRUE when the player used skill last tick.
- ***static int skillType;*** It's a type of skill <1=first type skill , 2=second type skill>

Method

- ***Getter & Setter;***
- static boolean checkSkill(Enemy alien, int x, int width, int y, int height);
- static void animationChooseSkill(Graphics2D g2d);
- static void animationFillSkillClick(Graphics2D g2d);
- static void animationFillSkillLight(Graphics2D g2d);
- static void changeSkill();
- static void updateSkill2();

1.10 Class "logic.SkillLightUpdate"

Method

- ***void run();***

1.11 Class "logic.Time"

Field

- ***static int time;*** The current time unit. (1 time unit = 100 ms)

Method

- ***Getter & Setter;***
- ***void run();*** Start the game timer.

2. PACKAGE 'render'

Need to add a table of contents or a bibliography? No sweat.

2.1 Class "render. AudioUtility"

This class is used for manage the sounds.

Field

- ***static AudioClip backgroundSound;*** It plays while game is running,
- ***static AudioClip gameOverSound;*** It plays when game is over.
- ***static AudioClip freezeBGSound;*** It plays when the player use skill2.
- ***static AudioClip pushButtonSound;*** It plays when the player push button.
- ***static AudioClip clickSound;*** It plays when the player click.
- ***static AudioClip dieGreenSound;*** It plays when the alien die.
- ***static AudioClip alienBuildSound;*** It plays when the rocket was built.

Method

- ***static AudioClip getSound(String directory);*** Load sound from a specify directory with ClassLoader and return the sound in the format of AudioClip.
- ***static void playBackgroundSound(AudioClip sound, boolean loopMode, boolean stopCurrentBGM);*** This method plays background sound ('sound') loop if 'loopMode' is true, and need to stop the background music that are playing or not.
- ***static void playMusicSound(AudioClip sound);*** This method plays sound effect.
- ***static void stopBackgroundSound();***This method stops background music.

2.2 Class "render. ConfigurableOption"

Field

- ***static final int screenWidth=800;***
- ***static final int screenHeight=600;***
- ***static final boolean enableFXSound=true;***
- ***static final boolean enableMusicSound=true;***

2.3 Class "render. DrawingUtility"

This class loads and stores all images in the game. This class can accessed from every classes.

Field

- ***static final BufferedImage bgGameTitle;*** It is a title scene's background.
- ***static final BufferedImage gameOver, newHighScore, newHighScoreEnterName;*** It is a game over scene's background that show when end game.
- ***static final BufferedImage highscoreBg;*** It is a highscore scene's background.

- ***static final BufferedImage skillClick, skillLight, skillFreezeBg;*** It is a skill's image.
- ***static final BufferedImage emptyRocket, fullRocket;*** It is a rocket's image.
- ***static final BufferedImage greenSprite, pinkSprite;*** It is a sprite sheet of alien.
- ***static final BufferedImage startPushed, exitPushed, highscorePushed, settingPushed, pausePushed, okPushed;*** It is a button's image.
- ***static final BufferedImage settingBgOn, settingBgOff;*** It is a setting's image.

Method

- ***static BufferedImage getImage(String directory);*** Load image from a specify directory with ClassLoader and return the image in the format of BufferImage.

2.4 Class "render. InputUtility"

This class is used for recording all of the statuses from mouse clicking and differentiate between "click the button" or "press and hold the button".

Field

- ***static boolean isLeftDown, isRightDown ;*** Status while pressing mouse.
- ***static boolean isLeftClickedLastTick, isRightClickedLastTick;*** Status for pressing mouse last tick.
- ***static boolean isLeftReleased, isRightReleased;*** Status while releasing mouse.
- ***static Position posClicked ;*** The current clicked mouse position.
- ***static Position posReleased;*** The current released mouse position.

Method

- ***getters & setters;*** Completely and properly add all getters & setters.
- ***static void postUpdate();*** This method is used for updating input status after "end of tick".

2.5 Interface "render. IRenderable"

This is the interface for the object which can be rendered.

Method

- ***int getZ();*** It returns index of object in z-axis.
- ***void draw(Graphics2D g2d);*** Draw the elements
- ***boolean isDead();*** It returns true when the alien is dead.
- ***boolean isVisible();*** It returns true when the alien is visible (not yet dead).

2.6 Class "render. RenderManager"

This class stores, controls, and rendering the IRenderable objects.

Field

- ***ArrayList<IRenderable> enemies;*** All of IRenderable object (All of enemies) need to be rendered.

Constructor

- ***RenderManager();***

Method

- ***void add(IRenderable enemy);*** Adding enemy (IRenderable object) for rendering.

- ***void drawAll(Graphics2D g2d);*** Drawing all IRenderable objects.
- ***void update();*** Update the status in every ticks.

3. PACKAGE 'ui'

3.1 Class "ui. GameManager"

This class is game manager that consists of many parts: creating game pages, starting the game, game loop, start the timer threads.

Field

- ***static GameWindow gameWindow;*** It's use for switch scene.
- ***static GameTitle gameTitle;*** The panel for title scene.
- ***static GameScreen gameScreen;*** The panel for game screen.
- ***static YourScore yourScore;*** The panel for the score when end game.
- ***static Setting setting;*** The panel for setting scene.
- ***static HighScore highScore;*** The panel for highscore scene.
- ***static RenderManager renderManager;*** The render manager.
- ***static ReleaseEnemy releaseEnemy;*** The releasing enemies logic.
- ***static Thread timer, click;*** Timer Thread, recharging skill cooldown timer Thread.

Method

- ***static void runGame();*** Start the game.
- ***static void newGame();*** Start a new game (change the panel to GameScreen).
- ***static void updateLogic();*** Update every game logic status in every ticks.

3.2 Class "ui. GameScreen"

GameScreen is the main page of this game consists of 'Pause' button, 'Score bar', 'Skill' recharging status, and game interface.

Field

- ***static boolean isPointOverPause;*** It's TRUE when the player holds cursor over the 'PAUSE' button.
- ***RenderManager renderManager;*** It's use for draw renderable object.

Constructor

- ***GameScreen(RenderManager renderMager);*** Initialize the value and resize screen and add listener.

Method

- ***void applyResize();*** Specify screen size as defined in ConfigurableOption.
- ***void addListener();*** It is responsible for capturing action events. This game records all of the inputs.
- ***void paintComponent(Graphics g);*** It is a default method of JComponent. It is called automatically and must be overridden to draw graphic objects.

3.3 Class "ui. GameTitle"

GameTitle is the title page/ menu page. This page links to 'Highscore' page, 'GameScreen' page, and 'Setting' page.

Field

- ***static GameWindow gameWindow;*** It's use for switch scene.
- ***boolean isPointOverNewGame;*** It's TRUE when the player holds cursor over the 'New Game' button.
- ***boolean isPointOverExit;*** It's TRUE when the player holds cursor over the 'Exit' button.

- ***boolean isPointOverHighscore;*** It's TRUE when the player holds cursor over the 'Highscore' button.
- ***boolean isPointOverSetting;*** It's TRUE when the player holds cursor over the 'Setting' button.
- ***boolean isPushedHighscoreButton;*** It's TRUE when the player push the 'Highscore' button.

Constructor

- ***GameTitle(GameWindow gameWindow);*** Initialize the value and resize screen and add listener.

Method

- ***void applyResize();*** Specify screen size as defined in ConfigurableOption.
- ***void addListener();*** It is responsible for capturing action events. This game records all of the inputs.
- ***void paintComponent(Graphics g);*** It is a default method of JComponent. It is called automatically and must be overridden to draw graphic objects.

3.4 Class "ui. GameWindow"

GameWindow is the main window of the game that can switch between the pages.

Field

- ***JPanel currentScene;*** It's a current scene.

Constructor

- ***GameWindow();*** Resize screen and set the operation of JFrame.

Method

- *void addFirstScene(JPanel scene);* This Method use for specify the first scene.
- *void switchScene(JPanel scene);* This method uses for switch scene.
- *JPanel getCurrentScene();* It's getter of currentScene.
- *void setCurrentScene(JPanel currentScene);* It's setter of currentScene.

3.5 Class "ui. HighScore"

This class manages the user records with the top 3 highest score of the game in a file. By default, it initializes the "default score" in the beginning of the program usage (no scores recorded yet).

Field

- *static HighScoreRecord[] highScoreRecord;* An array storing the top 3 highest scores players that is loaded from the scores' file. The beginning value is null.
- *static final byte[] key;* A key to encrypt player's data.
- *static String readFileName = "highscore";* The filename storing the top 3 highest scores.
- *static GameWindow gameWindow;* It's use for switch scene.

Constructor

- *HighScore(GameWindow gameWindow);* Initialize the value and resize screen and add listener.

Method

- *void applyResize();* Specify screen size as defined in ConfigurableOption.
- *void addListener();* It is responsible for capturing action events. This game records all of the inputs.

- ***void paintComponent(Graphics g);*** It is a default method of JComponent. It is called automatically and must be overridden to draw graphic objects.
- ***void setReadFileName(String name);*** A setter method for the variable "readFileName".
- ***static void recordHighScore(int score***); When the game is over, this method is automatically called to record the player's score.
- ***static void recordHighScoreName(String name, int score);*** The method recordHighScore will call this method if the player is in rank.
- ***static void drawTop3();*** It displays the data of the top 3 highest score in the message dialog.
- ***static boolean loadHighScore();*** This method tries to load data from the "highscore" file and return the status whether or not it is success (file existed). If it fails, delete the file and then create the default file.
- ***static boolean readAndParseScoreFile(File f);*** This method read from the "highscore" file. Then, it parses and assigns value to the array highScoreRecord. Note that it is called by the loadHighScore() method.
- ***static boolean createDefaultScoreFile();*** This method creates the default score file (highscore) if the has not been created yet.
- ***static String getXORed(String in);*** This method aims to encrypt and decrypt message by use XOR operation.

3.6 Class "ui. HighScore.HighScoreRecord"

This class is a static inner class only used within the HighScore class. It represents each player's record. Because there is a comparison among player's records, the Comparable method must be implemented.

Field

- ***String name;*** A player's name
- ***int score;*** A player's score.

Constructor

- ***HighScoreRecord(String name, int score);*** It assigns values to fields based on parameters.
- ***HighScoreRecord(String record);*** The input of this method is a string in the format of "Player_Name:Scores".

Method

- ***String getRecord();*** It returns the player's info in the format "Player_Name:Scores".
- ***String[] defaultRecord();*** It returns an array of players' data as default records of the top 3 players.
- ***int compareTo(HighScoreRecord o);*** This method belongs to the Comparable interface. It is used for comparing to player's data. The player with higher score must come before in the rank.

3.7 Class "ui. Setting"

The player can change the configuration in this 'Setting' page. This page consists of effect sound and background music switches.

Field

- ***static GameWindow gameWindow;*** It's use for switch scene.

Constructor

- ***Setting(GameWindow gameWindow)***; Initialize the value and resize screen and add listener.

Method

- ***void applyResize();*** Specify screen size as defined in ConfigurableOption.
- ***void addListener();*** It is responsible for capturing action events. This game records all of the inputs.

- ***void paintComponent(Graphics g);*** It is a default method of JComponent. It is called automatically and must be overridden to draw graphic objects.

3.8 Class "ui. YourScore"

YourScore is the page displayed after the player get the new high score and show the boxes for getting the player name.

Field

- ***boolean isPointOverOkGameOver, isPointOverOkEnter;*** It's TRUE when the player holds cursor over the 'OK' button.
- ***GameWindow gameWindow;*** It's use for switch scene.
- ***static JTextField nameInputField;*** It's a field to input the winner player's name.
- ***static boolean showInputField;*** It's TRUE when you are in rank.
- ***static String name;*** The winner player's name.

Method

- ***void applyResize();*** Specify screen size as defined in ConfigurableOption.
- ***void addListener();*** It is responsible for capturing action events. This game records all of the inputs.
- ***void paintComponent(Graphics g);*** It is a default method of JComponent. It is called automatically and must be overridden to draw graphic objects.

4. CLASS 'Main'

Class "Main"

In the main method, it starts running the game.

```
import ui.GameManager;

public class Main {
	public static void main(String[] args) {
		GameManager.runGame();
	}
}
```