Project: Key of Clairvoyance

What is this?

This is a game for person who interested in ability to do things simultaneously in which sometimes, it's important and in need for some people. In this game we have two mode of playing, first – SINGLE MODE, second – ENDLESS MODE. In game play, we have two screens of games. Left screen consists of key A, S, W and D that are falling down as the time goes by, you have to press the key that is the same to falling block so that the block will be destroyed and you will get score. Otherwise, if you let the falling block get through the screen, your score will be decreased. Right screen consists of circle figures that have number in it range from 1 to 4. In order to get score, you have to click those figures ascending in sequence from 1 to 4. If you click the figure that is inconsistent with the sequence, your score will be decreased. In SINGLE MODE, you have 3 minutes to score as many as you can by which everything works as above. But in ENDLESS MODE you will have 30 seconds time remaining and you have to score as many as you can until the time runs out which if you play the game correctly the time remaining will be increased, otherwise if you try to type the key that aren't falling down or click the figures not in sequence your time will be decreased.

Figure 1 - MainPanel

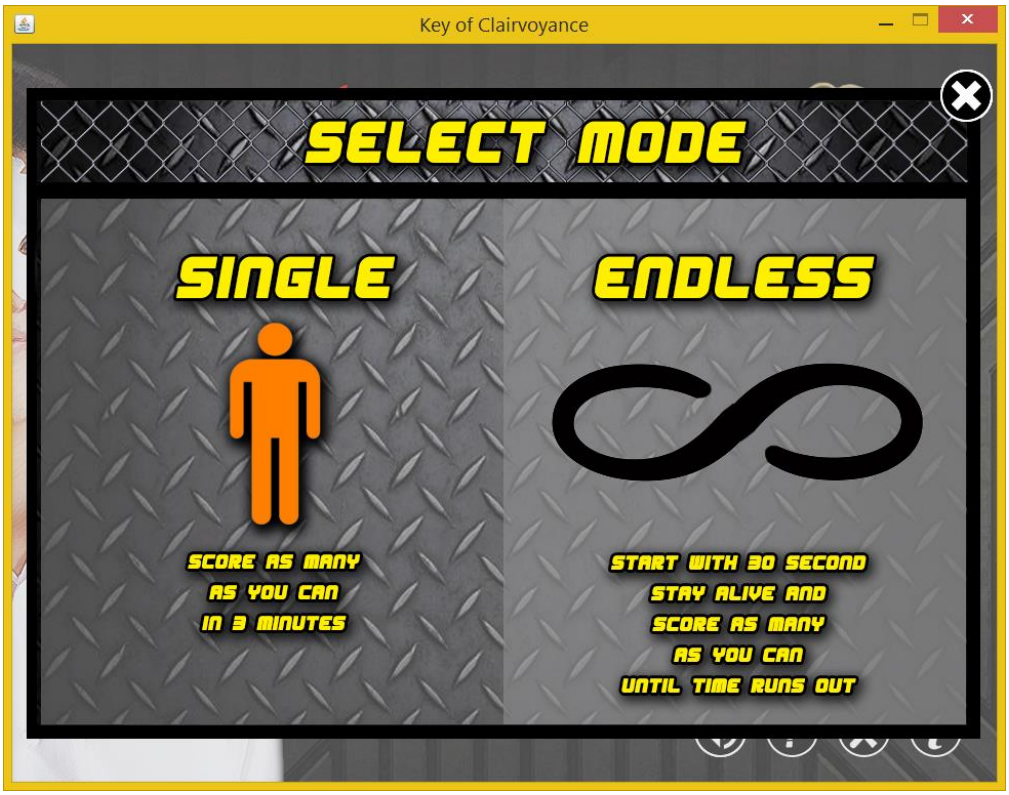


Figure 2 - ModePanel

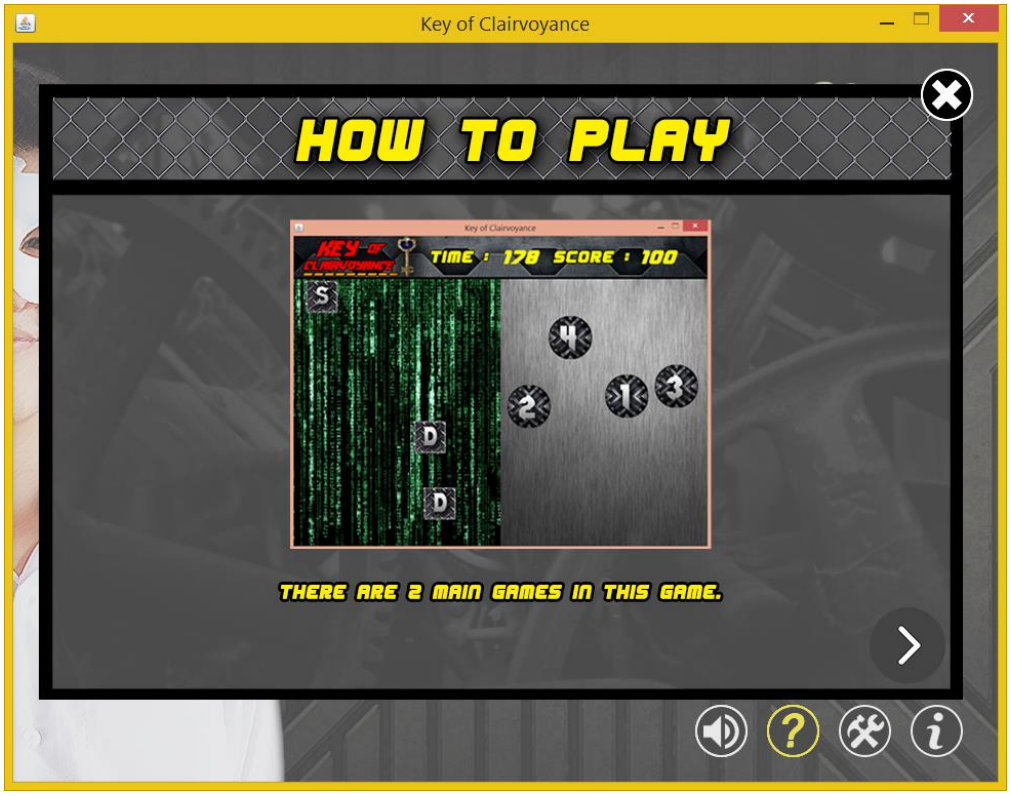


Figure 3 - HowToPlayPanel

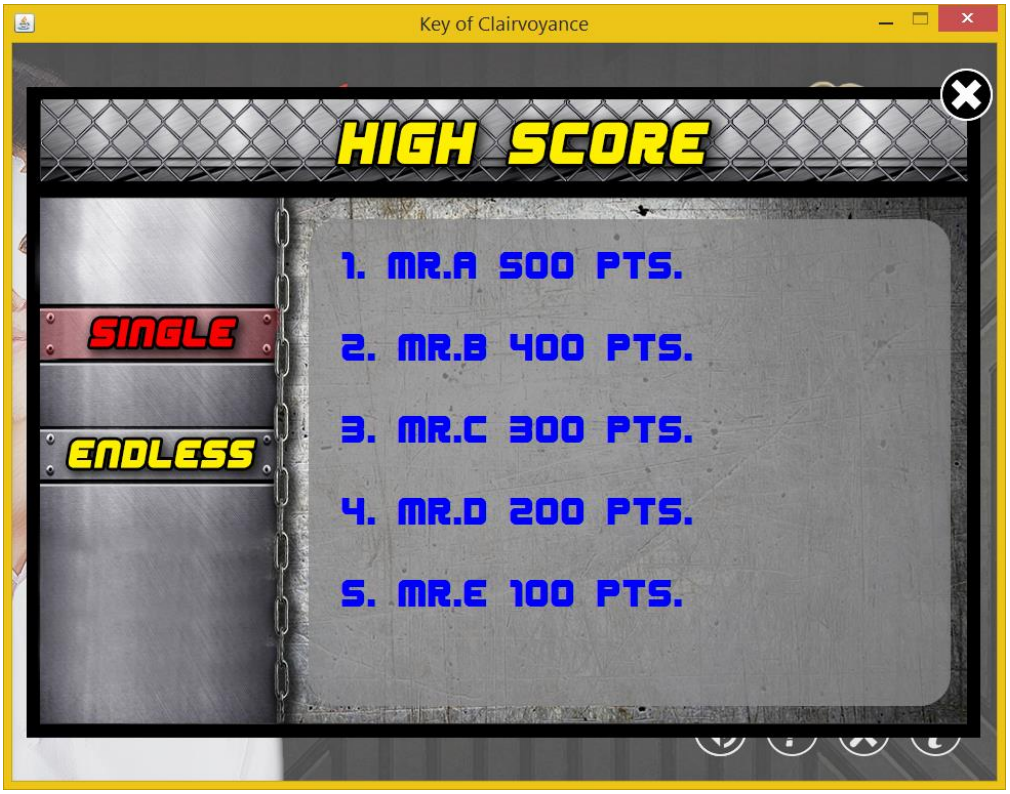


Figure 4 - HighScorePanel

Figure 5 - SettingPanel

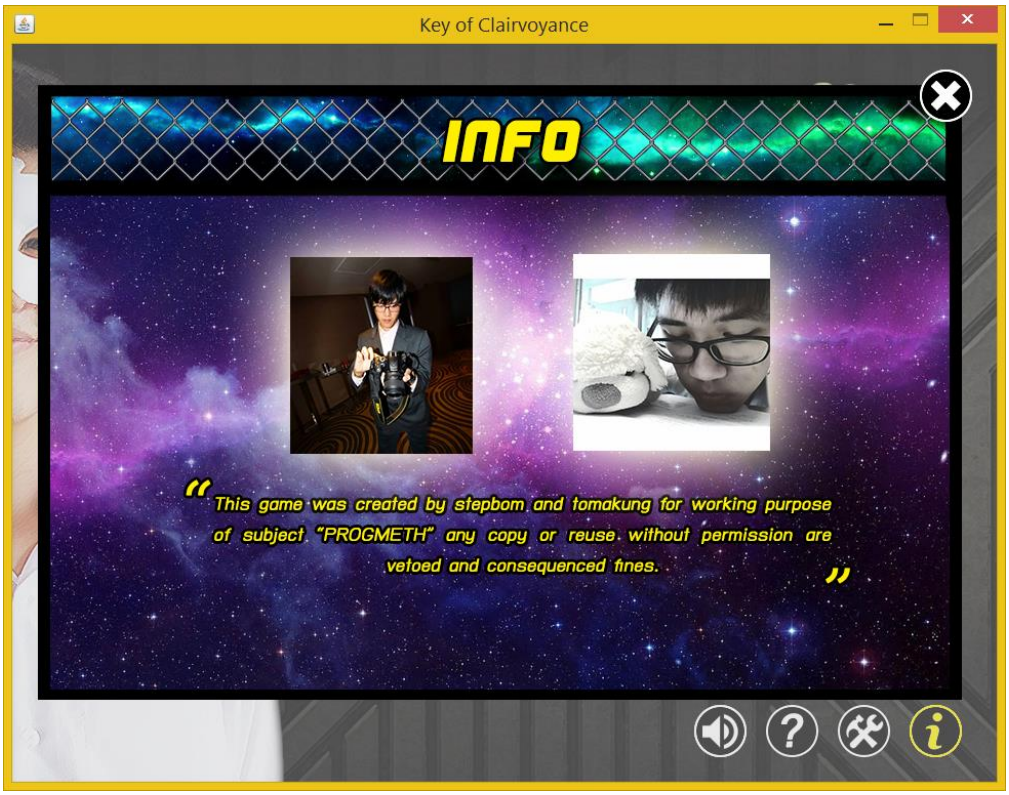


Figure 6 - InfoPanel

Figure 7 - StoryPanel

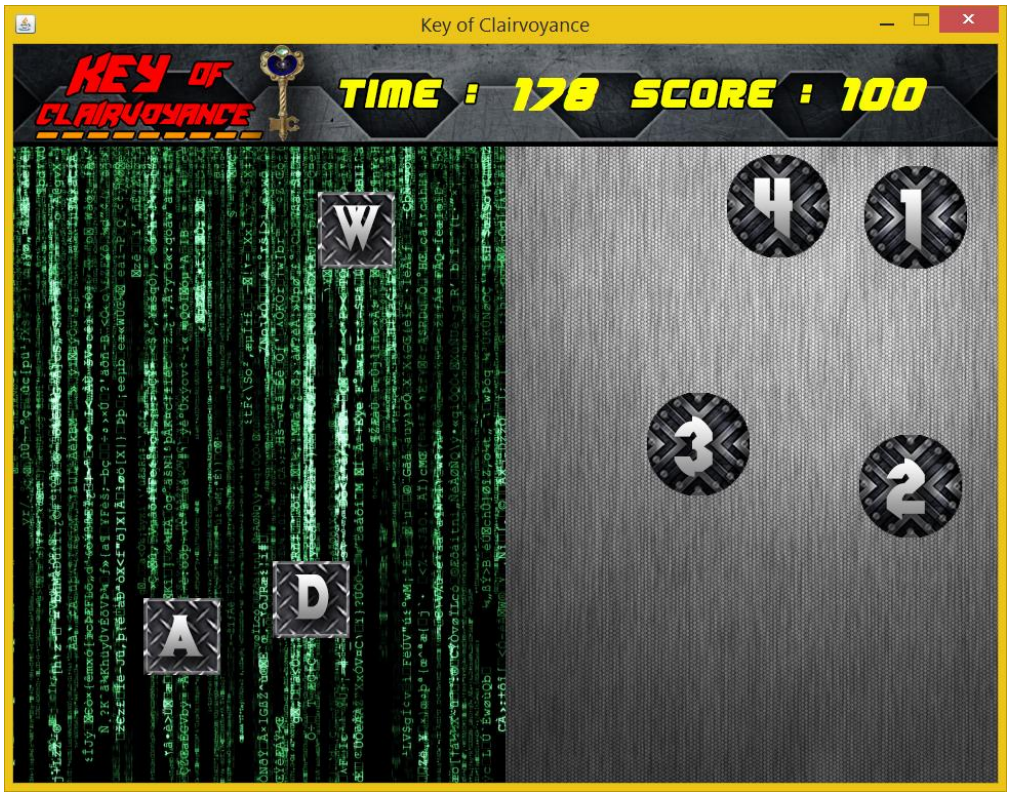


Figure 8 – GameScreen(Single Mode)

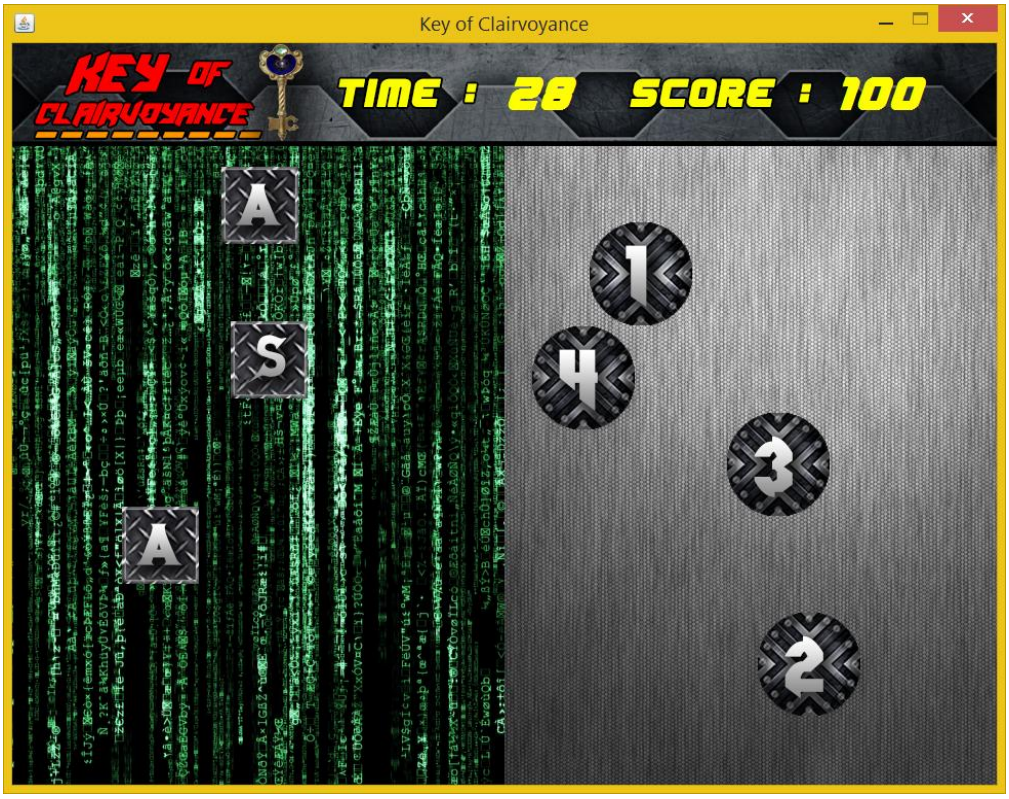
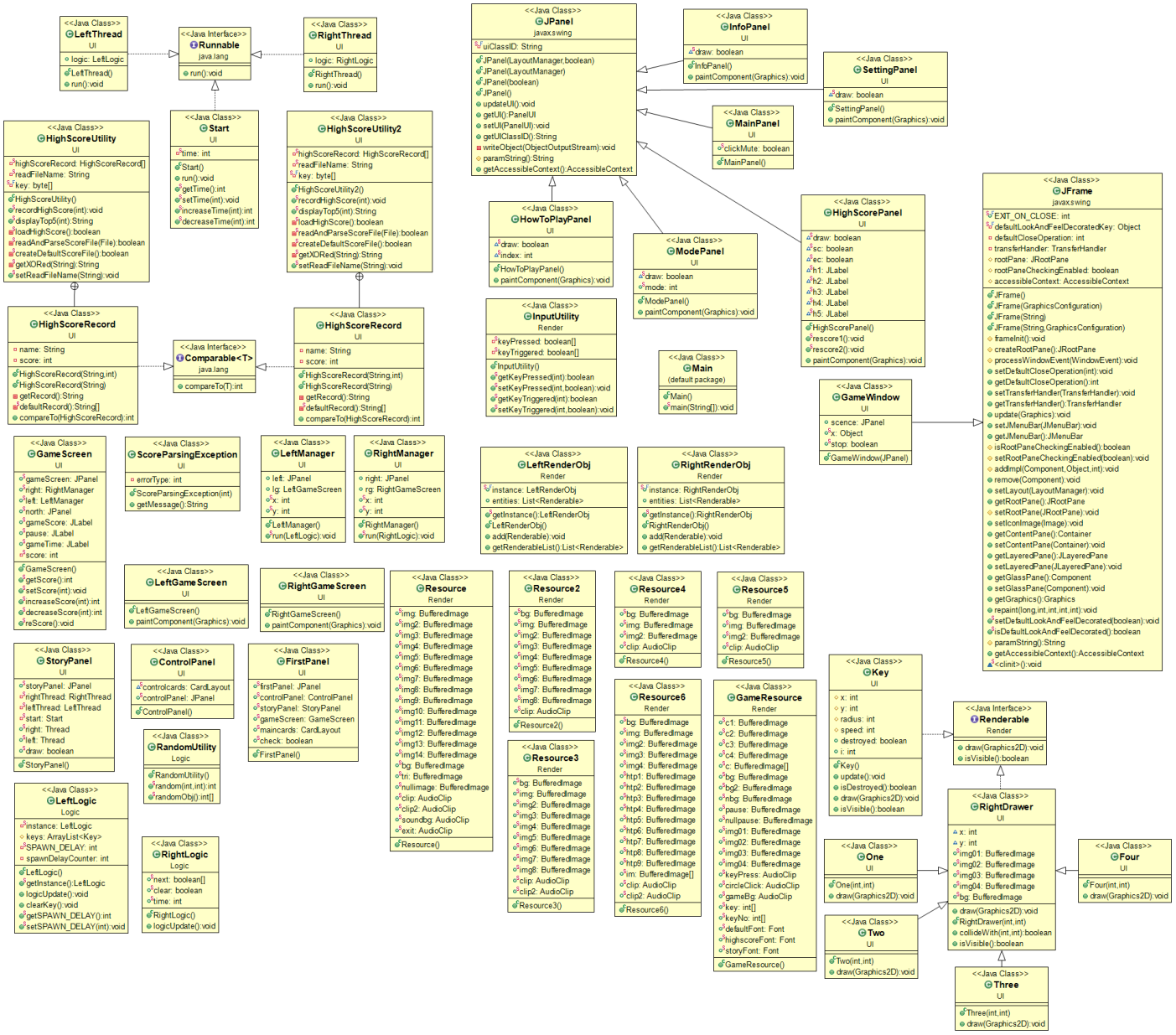


Figure 9 - GameScreen(Endless Mode)


LeftThread
RightThread
Runnable
JPanel
InfoPanel
SettingPanel
MainPanel
Start
HighScoreUtility
HighScoreUtility2
HowToPlayPanel
ModePanel
HighScorePanel
JFrame
HighScoreRecord
Comparable<T>
InputUtility
Main
GameWindow
GameScreen
ScoreParsingException
LeftManager
RightManager
LeftRenderObj
RightRenderObj
LeftGameScreen
RightGameScreen
Resource
Resource2
Resource4
Resource5
StoryPanel
ControlPanel
FirstPanel
Resource6
GameResource
Key
Renderable
RandomUtility
RightDrawer
LeftLogic
RightLogic
One
Two
Three
Four

UML Diagram

1. Package: default package

1.1 Class Main

- The main run by using EventQueue.invokelater method.
- Create new FirstPanel.
- Create new GameWindow(FirstPanel.firstPanel).

2. Package: Render

2.1 Class InputUtility

2.1.1 Field

- **Boolean[ ] keyPressed;** status if key is pressed or not.
- **Boolean[ ] keyTriggered;** status if key is triggered or not.

2.1.2 Method

- **Boolean getKeyPressed(int key);** if key is in range of 0 to 256 then return value of keyPressed[key].
- **Boolean setKeyPressed(int key, Boolean pressed);** if key is in range of 0 to 256 then set value of keyPressed[key] to pressed.
- **Boolean getKeyTriggered(int key);** if key is in range of 0 to 256 then return value of keyTriggered[key].
- **Boolean setKeyTriggered(int key, Boolean pressed);** if key is in range of 0 to 256 then set value of keyTriggered[key] to pressed.

2.2 Class Renderable (interface)

2.2.1 Method

- **Void Draw(Graphic2D g);** abstract method for drawing.
- **Boolean isVisible();** abstract method for checking whether it's visible or not.

2.3 Class RightRenderObj

2.3.1 Field

- **RightRenderObj instance;** this is the current instance.
- **List<Renderable> entities;** this is the list of render object.

2.3.2 Constructor

- **RenderableObj();** create new entities.

2.3.3 Method

- **RightRenderableObj getInstance();** return instance.
- **Void add(Renderable entity);** add render object to entities.
- **List<Renderable> getRenderableList();** return entities.

2.4 Class LeftRenderObj

2.4.1 Field

- **LefttRenderObj instance;** this is the current instance.
- **List<Renderable> entities;** this is the list of render object.

2.4.2 Constructor

- **RenderableObj();** create new entities.

2.4.3 Method

- **LeftRenderableObj getInstance();** return instance.

- **Void add(Renderable entity);** add render object to entities.
- **List<Renderable> getRenderableList();** return entities.

2.5 Class Resource

This class contains image and sound resources of the **MainPanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip().

2.6 Class Resource2

This class contains image and sound resources of the **ModePanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip().

2.7 Class Resource3

This class contains image and sound resources of the **HighScorePanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip().

2.8 Class Resource4

This class contains image and sound resources of the **SettingPanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip().

2.9 Class Resource5

This class contains image and sound resources of the **InfoPanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip().

2.10 Class Resource6

This class contains image and sound resources of the **HowToPlayPanel**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip(). This Resource also contains array of BufferedImage for easier change of picture in HowToPlayPanel.

2.11 Class GameResource

This class contains image, sound and font resources of the **GameScreen**. All images and sounds are initialized by using static block which implement classloader to load those contents to the static variable by ImageIO.read() and Applet.newAudioClip(). But when initialize font we need to create GraphicsEnvironment ge and then initialize it as GraphicsEnvironmentgetLocalGraphicsEnvironment(). Then create inputStream and using classloader to getResourceAsStream after that use the method ge.registerFont(Font.createFont(Font.TRUETYPE_FONT,inputStream)) to register this font into this game. This Resource also contains array of BufferedImage for easier change of picture of key in left game screen. And also array of Boolean keyNo to check the quantity of A, W, S or D key that isn't destroyed.

3. Package: UI

3.1 Class GameWindow (extends JFrame)

3.1.1 Field

- **JPanel scence;** this is the JPanel of the JFrame.
- **Object x;** this is the locked object for synchronizing.
- **Boolean stop;** this is to check if the game is pause or not.

3.1.2 Constructor

- **GameWindow(JPanel scence);** create JFrame set Preferred Size to 960 x 720, terminated when click close button, setResizable to false, setFocusable to true along with requestFocus(). Then add scence to the filed scence. After that, add key listener to this JFrame to detect A, W, S and D key, play keyPress sound when pressed and has detection of enter key to pause the game by setting stop to true and when press enter key again set stop to false notify all the sleep Thread. Finally, make frame visible.

3.2 Class FirstPanel

3.2.1 Field

- **JPanel firstPanel;** this is firstPanel to make card layout as a maincards.
- **ControlPanel controlPanel;** this is controlPanel to make card layout as a controlcards.
- **GameScreen gameScreen;** this is gameScreen when playing game.
- **StoryPanel storyPanel;** this is storyPanel when before playing game.

- **CardLayout maincards;** this is maincards for firstPanel.
- **Boolean check;** to check if the game is played or not.

3.2.2 Constructor

- **FirstPanel();** set layout to card layout which is maincards then add controlPanel, storyPanel and gameScreen to this panel respectively.

3.3 Class ControlPanel

3.3.1 Field

- **CardLayout controlcards;** this is controlcards for JPanel controlPanel.
- **JPanel controlPanel;** this is controlPanel JPanel to make card layout as controlcards.

3.3.2 Constructor

- **ControlPanel();** set layout of controlPanel to card layout which is controlcards then create new MainPanel, ModePanel, SettingPanel, HighScorePanel, InfoPanel and HowToPlayPanel then add those panels into controlPanel.

3.4 Class MainPanel (extends JPanel)

3.4.1 Field

- **Boolean clickMute;** create boolean for checking whether the background sound is mute or not by default value is false.

3.4.2 Constructor

- **MainPanel();** set layout of this panel to null then create JLabels that contain startgame button, background, highscore button, info button, mute/unmute button, setting button and how to play button image. After that, play the soundbg. Then add mouse listener to each of them to change to the panel they are designated for and create some animations and sounds when mouse is entered or exited.

3.5 Class ModePanel (extends JPanel)

3.5.1 Field

- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.
- **Int mode;** to save which mode that player is going to play.

3.5.2 Constructor

- **ModePanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels that contain background, single-mode, endless-mode, exit button image. Then add mouse listener to single-mode and endless-mode image to change the panel to storyPanel and set mode to 1 if single-mode is clicked and 2 if endless-mode is clicked and set the configuration of the game up to the mode then create some animations and sounds when mouse is entered or exited. Then add JLabel that is exit button image and add mouse listener to return to the

MainPanel.mainPanel and create animations when mouse entered or exited.

3.5.3 Method

- **Void paintComponent(Graphics g);** draw low opacity background of this panel.

3.6 Class SettingPanel

3.6.1 Field

- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.

3.6.2 Constructor

- **SettingPanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels that contain background and size-selecter image. Then add mouse listener to size-selecter image to change to change the size of the panel (but this game has only one size so there's no change here) and create some animations and sounds when mouse is entered or exited. Then add JLabel that is exit button image and add mouse listener to return to the MainPanel.mainPanel and create animations when mouse entered or exited.

3.6.3 Method

- **Void paintComponent(Graphics g);** draw low opacity background of this panel.

3.7 Class HighScorePanel

3.7.1 Field

- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.
- **Boolean sc;** to check whether the single highscore button image is clicked or not.
- **Boolean ec;** to check whether the endless highscore button image is clicked or not.
- **JLabel h1;** for showing highscore no 1;
- **JLabel h2;** for showing highscore no 2;
- **JLabel h3;** for showing highscore no 3;
- **JLabel h4;** for showing highscore no 4;
- **JLabel h5;** for showing highscore no 5;

3.7.2 Constructor

- **HighScorepanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels that contain background image, single select image, endless select image, and highscore 1 to 5. And set font to GameResource.highscoreFont to all JLabel from h1 to h5. Then add mouse listener to single and endless select image to change the JLabel highscore to display the mode that is selected and create some animations and sounds when mouse is entered or exited. Then add JLabel that is exit button image and add mouse listener to return to the MainPanel.mainPanel and create animations when mouse entered or exited.

3.7.3 Method

- **Void paintComponent(Graphics g);** draw low opacity background of this panel.
- **Void rescore1();** update the JLabel h1 to h5 to the latest highscore of single mode.
- **Void rescore2();** update the JLabel h1 to h5 to the latest highscore of endless mode.

3.8 Class HighScoreUtitlity

3.8.1 Field

- **HighScoreRecord[ ] highScoreRecord;** array for saving highscore record of single mode.
- **String readFile;** the file name that is going to be read of single mode.
- **Byte [ ] key;** the key to encoded highscore text file to unchangeable text.

3.8.2 Method

- **Void recordHighScore(int score);** check whether loadHighScore is false or highScoreRecord is null and then show Error message. Then check the score whether it's higher than any highScoreRecord or not. If the score isn't higher, show Game Over and your score. But if the score is higher, sort the highScoreRecord and show message to input player name and congratulate including your rank in highScoreRecord. After that rewrite the new highScoreRecord to the highscore file, if can't save then show the error message.

- **String displayTop5(int rank);** display the player name and the point of that rank. But if can't load highscore or highScoreRecord = null return null.
- **Boolean loadHighScore();** create new file by readFileName, if can readAndParseScoreFile then return true. If there's no highscore, create default highscore. If can't read highscore, delete the file and create default highscore file and readAndParseScoreFile again and return the value of readAndParseScoreFile. If can't createDefaultScoreFile, return false.
- **Boolean readAndParseScoreFile(File f);** create new highScoreRecord of size 5 then read the data in highscore file f and then store it in highScoreRecord array and sort the array if there's ScoreParsingException show the error message and set highScoreRecord to ERROR_RECORD and return false.
- **boolean createDefaultScoreFile();** create default highscorefile by using HighScoreRecord.defaultRecord then encoded with getXORed, write a file and return true. But if there's a IOException set the highScoreRecord to null and return false.
- **String getXORed(String in);** both encoded/decoded the data in and return the string that finished encoded/decoded.
- **void setReadFileName(String name);** set the readFileName to name.

3.9 Class HighScoreRecord (inner class of HighScoreUtitlity implement Comparable<HighScoreRecord>)

3.9.1 Field

- **String name;** name of the player equal “ ”.
- **Int score;** score of the player equal 0.

3.9.2 Constructor

- **HighScoreRecord(String name, int score);** setting this name equals name and this score equals score
- **HighScoreRecord(String record);** throws ScoreParsingException. And checking the record whether it’s in correct format or not. If its format isn’t correct throw ScoreParsingException(1) but if can’t parse String to Integer of the score then throw ScoreParsingException(0).

3.9.3 Method

- **String getRecord();** return name.trim() + “ ” + score.
- **String[] defaultRecord();** return default record of single mode highscore.
- **int compareTo(HighScoreRecord o);** to compare the score between this HighScoreRecord and HighScoreRecord o. return 1 if this score is lower, return -1 if this score is higher and if both score are equal return 0.

3.10 Class HighScoreUtitlity2

3.10.1 Field

- **HighScoreRecord[ ] highScoreRecord;** array for saving highscore record of endless mode.
- **String readFile;** the file name that is going to be read of endless mode.
- **Byte [ ] key;** the key to encoded highscore2 text file to unchangeable text.

3.10.2 Method

- **Void recordHighScore(int score);** check whether loadHighScore is false or highScoreRecord is null and then show Error message. Then check the score whether it's higher than any highScoreRecord or not. If the score isn't higher, show Game Over and your score. But if the score is higher, sort the highScoreRecord and show message to input player name and congratulate including your rank in highScoreRecord. After that rewrite the new highScoreRecord to the highscore2 file, if can't save then show the error message.
- **String displayTop5(int rank);** display the player name and the point of that rank. But if can't load highscore2 or highScoreRecord = null return null.
- **Boolean loadHighScore();** create new file by readFileName, if can readAndParseScoreFile then return true. If there's no highscore2, create default highscore. If can't read highscore2, delete the file and create default highscore file and readAndParseScoreFile again and return the value of

readAndParseScoreFile. If can't createDefaultScoreFile, return false.

- **Boolean readAndParseScoreFile(File f);** create new highScoreRecord of size 5 then read the data in highscore2 file f and then store it in highScoreRecord array and sort the array if there's ScoreParsingException show the error message and set highScoreRecord to ERROR_RECORD and return false.
- **boolean createDefaultScoreFile();** create default highscorefile by using HighScoreRecord.defaultRecord then encoded with getXORed, write a file and return true. But if there's a IOException set the highScoreRecord to null and return false.
- **String getXORed(String in);** both encoded/decoded the data in and return the string that finished encoded/decoded.
- **void setReadFileName(String name);** set the readFileName to name.

3.11 Class HighScoreRecord (inner class of HighScoreUtitlity2 implement Comparable<HighScoreRecord>)

3.11.1 Field

- **String name;** name of the player equal " ".
- **Int score;** score of the player equal 0.

3.11.2 Constructor

- **HighScore ScoreRecord(String name, int score);** setting this name equals name and this score equals score

- **HighScoreRecord(String record);** throws ScoreParsingException. And checking the record whether it's in correct format or not. If its format isn't correct throw ScoreParsingException(1) but if can't parse String to Integer of the score then throw ScoreParsingException(0).

3.11.3 Method

- **String getRecord();** return name.trim() + " " + score.
- **String[] defaultRecord();** return default record of endless mode highscore.
- **int compareTo(HighScoreRecord o);** to compare the score between this HighScoreRecord and HighScoreRecord o. return 1 if this score is lower, return -1 if this score is higher ands if both score are equal return 0.

3.12 Class ScoreParsingException

3.12.1 Field

- **Int errorType;** the errorType of this Exception.

3.12.2 Constructor

- **ScoreParsingException(int errorType);** set this errorType to errorType.

3.12.3 Method

- **String getMessage();** if error Type equals 0, return error message "No record score", otherwise, return error message "Wrong record format".

3.13 Class InfoPanel (extends JPanel)

3.13.1 Field

- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.

3.13.2 Constructor

- **InfoPanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels that contain background image which is info of the producer of this game. Then add mouse listener to previous and next button image to change the index of how to play and create some sounds when mouse is clicked (the previous can click only when it's not the first page and the next button can click only when it's not the last page). Then add JLabel that is exit button and add mouse listener to return to the MainPanel.mainPanel and create animations when mouse entered or exited.

3.13.3 Method

- **Void paintComponent(Graphics g);** draw low opacity background of this panel.

3.14 Class HowToPlayPanel

3.14.1 Field

- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.
- **Boolean index;** to represent the index of the BufferedImage of the array of how to play images.

3.14.2 Constructor

- **HowToPlayPanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels that contain background, how to play pictures, previous button and next button image. Then add JLabel that is exit button image and add mouse listener to return to the MainPanel.mainPanel and create animations when mouse entered or exited.

3.14.3 Method

- **Void paintComponent(Graphics g);** draw low opacity background of this panel.

3.15 Class StoryPanel

3.15.1 Field

- **JPanel storyPanel;** this is storyPanel before GameScreen.gameScreen Panel.
- **RightThread rightThread;** create runnable rightThread for right game screen.
- **LeftThread leftThread;** create runnable leftThread for left game screen.
- **Start start;** create runnable start which is time Thread.
- **Thread right;** create right Thread field.
- **Thread left;** create left Thread field.
- **Boolean draw;** Boolean to check whether the background has drawn or not in order to avoid glitch that repaint in the background.

3.15.2 Constructor

- **StoryPanel();** set draw to true, set layout to null, create low opacity background (have alpha value after r, g, b value), then create JLabels startLabel and set font to GameResource.storyFont and add mouse listener to this label to change this panel to GameScreen.gameScreen panel, then make boolean of FirstPanel to true and then start left, right and start Thread after that play the gamebg sound. Then add JLabel that is exit button image and add mouse listener to return to the mainPanel and create animations when mouse entered or exited.

3.16 Class GameScreen

3.16.1 Field

- **JPanel gameScreen;** create JPanel gameScreen.
- **RightManager right;** create RightManager right = null.
- **LeftManager left;** create LeftManager right = null.
- **JPanel north;** create JPanel north.
- **JLabel gameScore;** create JPanel gameScore to display gamescore.
- **JLabel pause;** create JPanel pause to display pause.
- **JLabel gameTime;** create JPanel gameTime to display time.

3.16.2 Constructor

- **GameScreen();** create new gameScreen, set layout to null, set preferred size to 960 x 720. Then create new JPanel north, new RightManager right and LeftManager left. Set the size of north panel to 960 x 100. Create new JLabel gameScore and set font to

GameResource.defaultFont and its location correlate to the background. Then create new JLabel pause and gameBg. Then set size of left.left and right.right to its preferred size.

3.16.3 Method

- **Void getScore();** return score value.
- **Void setScore(int score);** set score value to score.
- **Int increaseScore(int i);** increase score value by i.
- **Int decreaseScore(int i);** decrease score value by i.
- **Void rescore();** set gameScore to score.

3.17 Class RightThread (extends Runnable)

3.17.1 Field

- **RightLogic logic;** create RightLogic logic.

3.17.2 Method

- **Void run();** run the method run of RightManager by parameter logic.

3.18 Class RightManager

3.18.1 Field

- **JPanel right;** create new JPanel right.
- **RightGameScreen rg;** create rightGameScreen rg.
- **Int x;** set default x position to 900.
- **Int y;** set default y position to 900.

3.18.2 Constructor

- **RightManager();** throws InterruptedException. Then set the JPanel right Layout to null, set Preferred Size to 480 x 620 and set location to (480,100). Then create new RightGameScreen and create JLabel for background from rightBg image. Then add rg and JLabel to the right JPanel. After that, add mouse listener to detect the mouse pressed and set the position of x, y to the position when mouse pressed but when mouse released set x, y out of the right Panel.

3.18.3 Method

- **Void run(RightLogic logic);** while true and check if the time of class Start is equal or less than zero remove all the RightRenderObj. If Boolean check of FirstPanel is true sleep for 10 ms and then call logic's logic update method and repaint the GameScreen panel.

3.19 Class RightGameScreen (extends JComponent)

3.19.1 Constructor

- **RightGameScreen();** create array of PositionObj by using RandomUtility.randomObj(); then create RightRenderObj which is one, two , three and four and set its position by PositionObj array. Then set this component preferred size to 480 x 620.

3.19.2 Method

- **Void paintComponent(Graphics g);** Cast Graphics g to Graphics2D then check if Boolean check of FirstPanel is true then synchronized the entities of RightRenderObj in this block run for loop to draw all of its entities only when it's visible.

3.20 Class RightDrawer (extends Renderable)

3.20.1 Field

- **Int x;** set default position x of the object.
- **Int y;** set default position y of the object.
- **BufferImage img01;** create BufferedImage of figure one.
- **BufferImage img02;** create BufferedImage of figure two.
- **BufferImage img03;** create BufferedImage of figure three.
- **BufferImage img04;** create BufferedImage of figure four.
- **BufferImage bg;** create BufferedImage of Right Background.

3.20.2 Constructor

- **RightDrawer(int x, int y);** set filed x and y to value of x and y.

3.20.3 Method

- **Void draw(Graphics2D g);** this method use in its child class.
- **Boolean collideWith(int xx, int yy);** to check if the mouse click of position (xx, yy) is in the figure or not by Math.hypot() if it's clicked return true otherwise return false;
- **Boolean isVisible();** always return true.

3.21 Class One (extends RightDrawer)

3.21.1 Constructor

- **One(int x, int y);** super(x, y) to call parent class and set img01 to number1 image.

3.21.2 Method

- **Void draw(Graphics2D g2d);** use method drawImage(BufferedImage img01, int x, int y, ImageObserver null) to draw at img01 at x, y

3.22 Class Two (extends RightDrawer)

3.22.1 Constructor

- **Two(int x, int y);** super(x, y) to call parent class and set img02 to number2 image.

3.22.2 Method

- **Void draw(Graphics2D g2d);** use method drawImage(BufferedImage img02, int x, int y, ImageObserver null) to draw at img02 at x, y

3.23 Class Three (extends RightDrawer)

3.23.1 Constructor

- **Three(int x, int y);** super(x, y) to call parent class and set img03 to number3 image.

3.23.2 Method

- **Void draw(Graphics2D g2d);** use method drawImage(BufferedImage img03, int x, int y, ImageObserver null) to draw at img03 at x, y

3.24 Class Four (extends RightDrawer)

3.24.1 Constructor

- **Four(int x, int y);** super(x, y) to call parent class and set img04 to number4 image.

3.24.2 Method

- **Void draw(Graphics2D g2d);** use method drawImage(BufferedImage img04, int x, int y, ImageObserver null) to draw at img04 at x, y

3.25 Class LeftThread (extends Runnable)

3.25.1 Field

- **LeftLogic logic;** create RightLogic logic.

3.25.2 Method

- **Void run();** run the method run of RightManager by parameter logic.

3.26 Class LeftManager

3.26.1 Field

- **JPanel left;** create new JPanel left.
- **LeftGameScreen lg;** create lg LeftGameScreen.
- **Int x;** create x default position = 900.
- **Int y;** create y default position = 900.

3.26.2 Constructor

- **LeftManager();** throws InterruptedException, setFocusable to true, set layout to null, set Preferred Size to 480 x 620, set

Location to (0, 100), then create new lg LeftGameScreen and setPreferredSize(lg.getPrefererredSize()). Then create JLable background to make set gameScreen background. After that add lg and background to the leftPanel.

3.26.3 Method

- **Void run(LeftLogic logic);** while true and check if the time of class Start is equal or less than zero remove all the LeftRenderObj. If Boolean check of FirstPanel is true sleep for 10 ms and then call logic's logic update method and repaint the GameScreen panel.

3.27 Class LeftGameScreen (extends JComponent)

3.27.1 Constructor

- **LeftGameScreen();** set this component preferred size to 480 x 620.

3.27.2 Method

- **Void paintComponent(Graphics g);** cast Graphics g to Graphics2D then check if Boolean check of FirstPanel is true then synchronized the entities of LeftRenderObj in this block run for loop to draw all of its entities only when it's visible.

3.28 Class Key

3.28.1 Field

- **Int x;** this key x position.
- **Int y;** this key y position.
- **Int radius;** this key radius.

- **Int speed;** this key speed.
- **Boolean destroyed;** true when this key is destroyed.
- **Int i;** key value from a, s, d and w to the key.

3.28.2 Constructor

- **Key ();** use RandomUtility.random(0, 380) to random x position, set y = -100, use RandomUtility.random(3,6) to random speed, use RandomUtility.random(0,4) to random i, increase LeftResource.keyNo[i] to increase number of key that are available, set radius to 100 and destroyed = falsed.

3.28.3 Method

- **Void update();** increase y position by this key speed and if y > 600 destroyed = true.
- **Boolean isDestroyed();** return destroyed.
- **Void paintComponent(Graphics g);** use method drawImage(BufferedImage LeftResource.c[i], null, int x, int y) to draw key that has key value i at x, y
- **Boolean isVisible();** always return true.

3.29 Class Start

3.29.1 Field

- **Int time;** game time.

3.29.2 Method

- **Void run();** first synchronized GameWindow.x by if GameWindow.stop is true then wait(Pause the game) then wait for 1000 ms then decrease time by 1 and set GameScreen.gameTime to time but if time equal or less than zero, stop the gameBg sound and set GameScreen.gameTime to

zero and record the score up to the mode that selected. Then change the panel to MainPanel.

- **Int getTime();** return time.
- **Void setTime(int time);** set this time = time.
- **Int increaseTime(int i);** return time by time plus i
- **Int decreaseTime(int i);** return time by time minus i but if time <= 0 so time = 0

4. Package: Logic

4.1 Class RightLogic

4.1.1 Field

- **Boolean int[ ] next;** to check if the player click the figure in sequence or not by initialize it with false of size 4.
- **Boolean clear;** to refresh the figure 1 to 4 position if player can't click the figure in sequence in time.
- **Int time;** time of refreshing those figures initialize with 0.

4.1.2 Method

- **Void logicUpdate();** this method has to check whether the game is pause by synchronized with GameWindow.x by checking the Boolean GameWindow.stop and then use GameWindow.x.wait(). Then increase the time but if the size of entities of RightRenderObj is zero refresh the figures. But if player can't click the figures in time so set clear to true and decrease player's score by five and set time to zero but in Endless Mode every time you fail to complete the sequence the LeftLogic.SPANW_DELAY is decrease everytime by 2. So if clear is

true, remove all the RightRenderObj, set next[i] to false, random the position and create new one to four Figures. Finally check the sequence if player click the right figure next[figure] will be true and you will get one score(in Endless mode you also get one second time) and if the next[3] is true (player finish the sequence), player will get five score(in endless mode, will get more 5 seconds). Note that always use GameScreen.rescore() and GameScreen.gameTime.setText() to update the score and game time every time you increase or decrease score or time.

4.2 Class LeftLogic

4.2.1 Field

- **LeftLogic instance;** this is the current instance.
- **ArrayList<Key> keys;** ArrayList of keys that are on the LeftManager.left Panel.
- **Int SPAWN_DELAY;** spawn delay of the key that is falling down.
- **Int spawnDelayCouter;** the counter for the spawn delay.

4.2.2 Method

- **LeftLogic getInstance();** return this instance.
- **Void logicUpdate();** this method has to check whether the game is pause by synchronized with GameWindow.x by checking the Boolean GameWindow.stop and then use GameWindow.x.wait(). Then check that the key that player presses are the same with falling or not. If player press the key that hasn't fallen down, score will be decrease. But, if the player presses the right key, score will be increased. Then increase the spawnDelayCounter by 1 and check if spawnDelay is equal SPAWN_DELAY, add more key

to LeftRenderObj's entities and reset the spawnDelayCounter to 0.

- **Void clearKey();** remove all key both entities of LeftRenderObj and this class keys ArrayList. Then set GameResource.keyNo[i] (i from 0 to 3) to zero.
- **Int getSPAWN_DELAY;** return SPAWN_DELAY.
- **Void setSPAWN_DELAY(int delay);** to set SPAWN_DELAY equals delay but if delay is equal or less than 20 set SPAWN_DELAY equals 20.

4.3 Class RandomUtility

4.3.1 Method

- **Int random(int s, int e);** this method is to random the number in range from s to e.
- **Int [ ] randomObj();** this method is to random the figure 1 to 4 of right game screen to not overlap each other.

Extra note about this game

- Thread

In this game we use about 5 threads which are normal Thread, EventQueue Thread, Right Game Screen Thread, Left Game Screen Thread and Start Thread(time thread).

We use 2 synchronized object in our game.

First is GameWindow.x. This object is designated for wait and notify method. By using Enter key to triggered the Boolean stop. Then when left and right Thread are running (using method logicupdate in while(true)), they will notice that the boolean stop is true and use GameWindow.x.wait() until the player press the Enter key again to notifyAll() the Thread that are sleeping.

Second one is entities in RightRenderObj class. This object is designated to make the entities can access by only one Thread at a time which can result in good ways that this entities will be safe every time we run for loop to remove some entities.

Exception

In this game we have about 3 exists exception which is IOException(use in static block of every Resource class to load a BufferedImage by using classloader and use in HighScoreUtility and HighScoreUtility2 class when read a file), URISyntaxException(use in static block of every Resource class when load an audioclip by using class loader) and FontFormatException(use in static block of Resource class to load Font by using classloader).

Moreover, this game has an own exception which is ScoreParsingException. This exception is to detect whether there is no record score, or the record is wrong format in both HighScoreUtility and HighScoreUtility2 class. When this exception detects the error, it will return message to tell the condition of what's not right of the program.