Long Story

Long Story

Created by

Krit Kriengkasem 5831002521
Chawit U-Viengchai 5831011121

Overview

Long Story is the game that challenges players to match the objects on the game screen in the limited time. How many score you can get in just 60 seconds ?

Figure 1.1 is the main screen of this game. This screen displays when opening an application. On this screen, there are three buttons.

1. Play button : start playing the game
2. Exit button : close the game
3. About button : change to credit screen

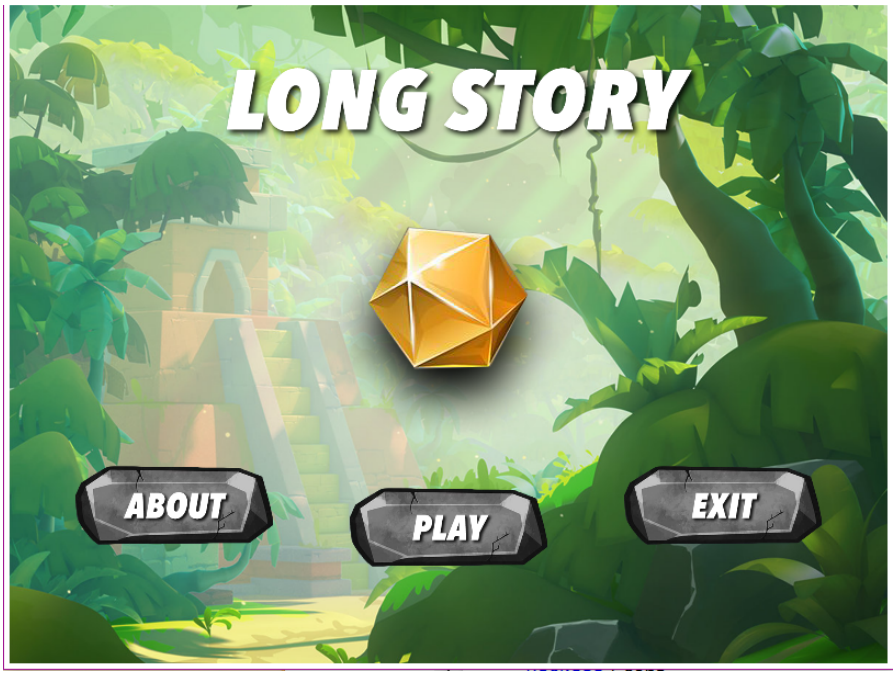


Figure 1.1: The main screen

Figure 1.2 is the credit screen of this game. On this screen, there is only one buttons at the bottom left of the credit screen.

Home button : change to main screen

Figure 1.2: The credit screen

Figure 1.3 is the game screen of this game. When players see the screen, the challenge has already started. The players play the game on this screen by clicking the cells on which are the center screen.

The group cells which can be destroy must be connection of at least three or more similar cells.

If players clicks the correct group cells, the score point will increase. Moreover, the combo bar will increase too. The more combo means the more point you get.

However, if players click wrong cells, the combo bar will reset to zero. Furthermore, the remaining time will decrease too.

There is only one buttons at top right of the game screen.

Pause button : pop up the pause screen in Figure 1.5

Figure 1.3: The game screen

Figure 1.4 is the status bar on the game screen that has

1. Score : the point that players get
2. Time : the remaining time
3. Combo : the combo bar

Figure 1.4: Status bar

Figure 1.5 is the pause screen. When this screen appears , the game screen will be paused. There are three buttons on this screen.

1. Retry button : start new game
2. Play button : continue playing the game
3. Home button : get back to the main screen

Figure 1.5: The pause screen

Figure 1.6 is game over screen. This screen will display when players have run out of time. There are two buttons on this screen.

1. Retry button : start new game
2. Home button : get back to the main screen

Figure 1.6: The end screen

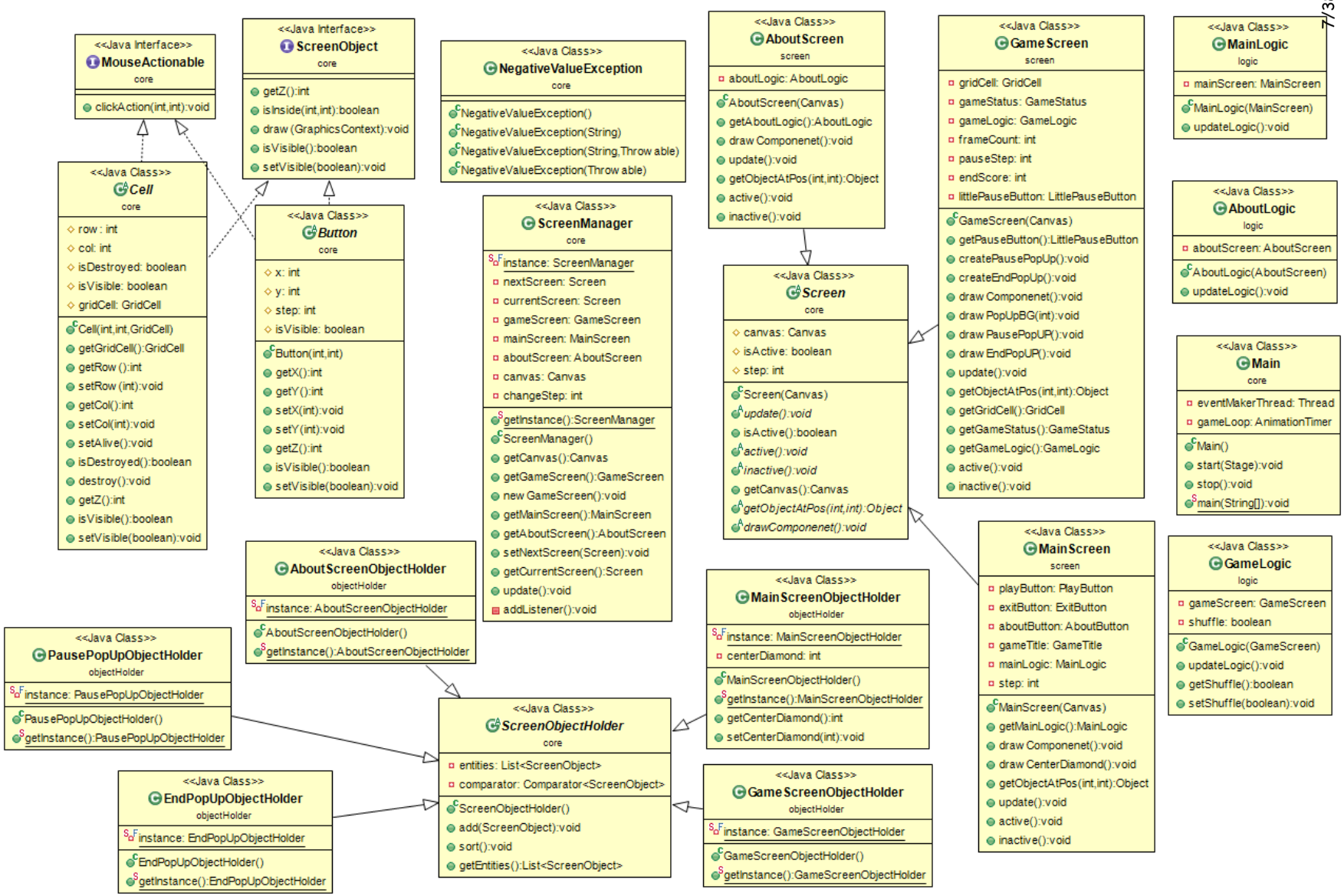
Figure 1.7 All cell pictures in the game.

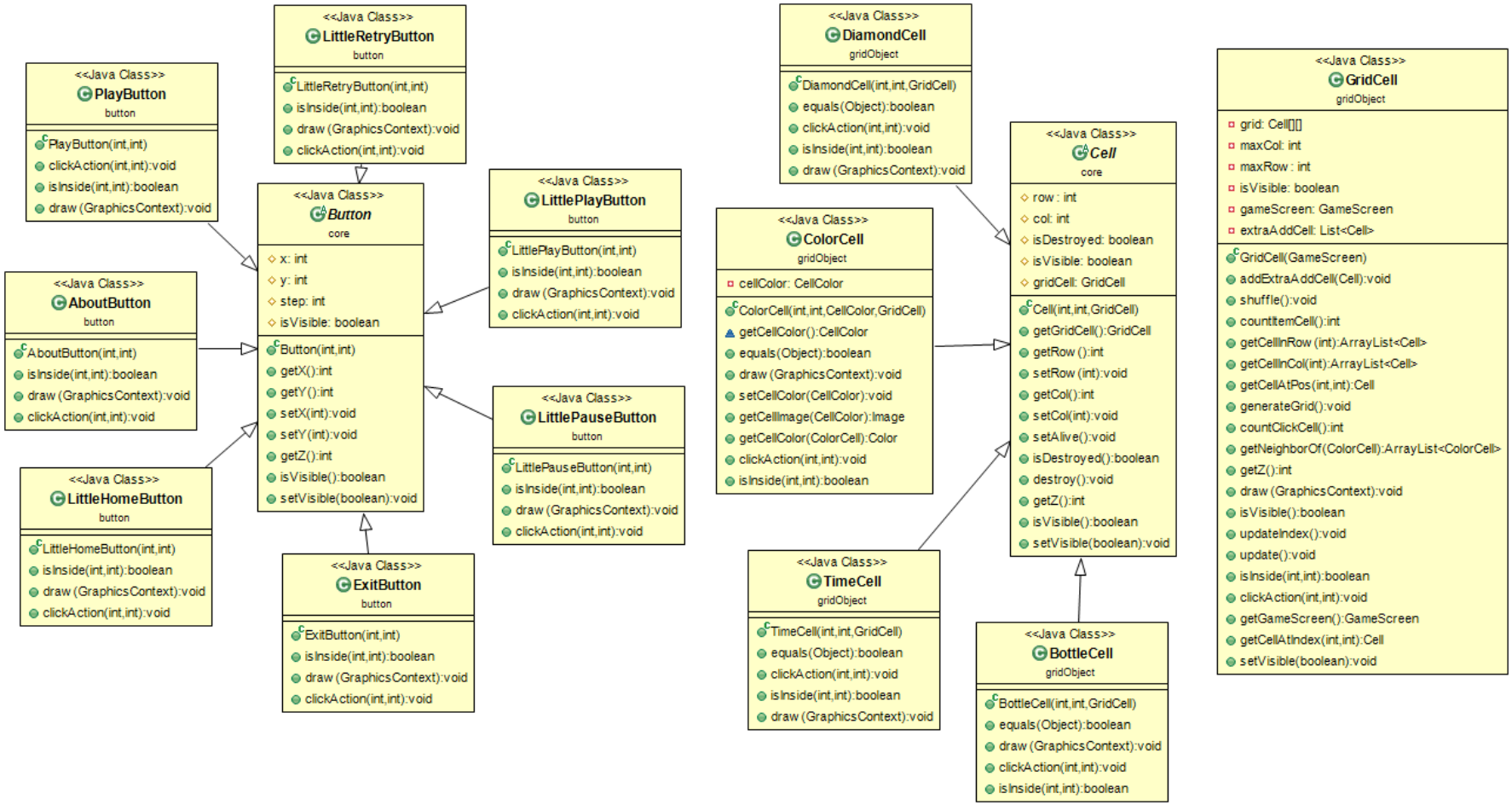
Figure 1.7: Cells

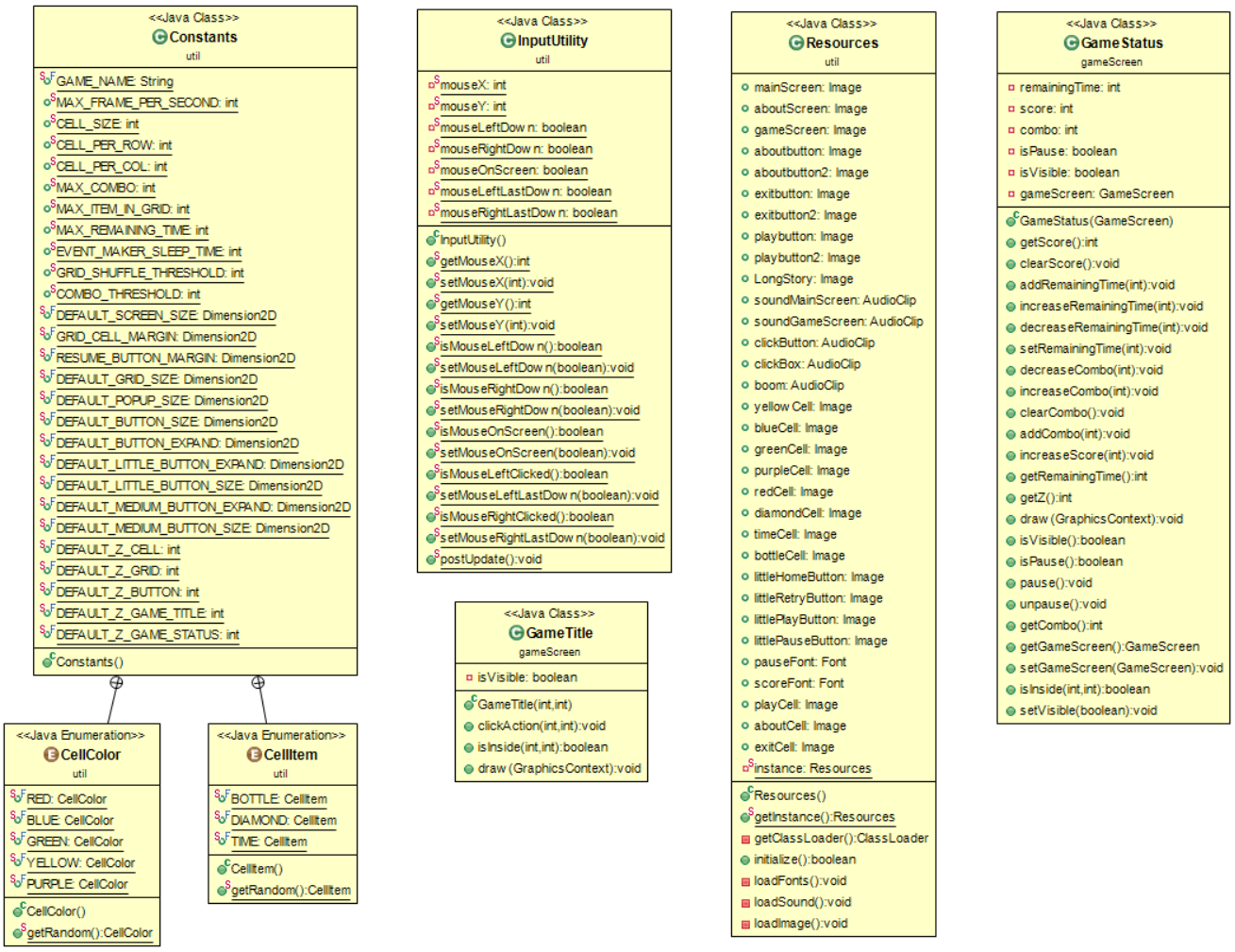
Figure 1.8 All item cell pictures in the game.

Figure 1.8: Item cells

Class Diagrams


<<Java Interface>>
MouseActionable
core
clickAction(int,int):void
<<Java Interface>>
ScreenObject
core
getZ():int
isInside(int,int):boolean
draw (GraphicsContext):void
isVisible():boolean
setVisible(boolean):void
<<Java Class>>
NegativeValueException
core
NegativeValueException()
NegativeValueException(String)
NegativeValueException(String,Throwable)
NegativeValueException(Throwable)
<<Java Class>>
AboutScreen
screen
aboutLogic: AboutLogic
AboutScreen(Canvas)
getAboutLogic():AboutLogic
draw Componenet():void
update():void
getObjectAtPos(int,int):Object
active():void
inactive():void
<<Java Class>>
GameScreen
screen
gridCell: GridCell
gameStatus: GameStatus
gameLogic: GameLogic
frameCount: int
pauseStep: int
endScore: int
littlePauseButton: LittlePauseButton
GameScreen(Canvas)
getPauseButton():LittlePauseButton
createPausePopUp():void
createEndPopUp():void
draw Componenet():void
draw PopUpBG(int):void
draw PausePopUP():void
draw EndPopUP():void
update():void
getObjectAtPos(int,int):Object
getGridCell():GridCell
getGameStatus():GameStatus
getGameLogic():GameLogic
active():void
inactive():void
<<Java Class>>
MainLogic
logic
mainScreen: MainScreen
MainLogic(MainScreen)
updateLogic():void
<<Java Class>>
Cell
core
row : int
col: int
isDestroyed: boolean
isVisible: boolean
gridCell: GridCell
Cell(int,int,GridCell)
getGridCell():GridCell
getRow ():int
setRow (int):void
getCol():int
setCol(int):void
setAlive():void
isDestroyed():boolean
destroy():void
getZ():int
isVisible():boolean
setVisible(boolean):void
<<Java Class>>
Button
core
x: int
y: int
step: int
isVisible: boolean
Button(int,int)
getX():int
getY():int
setX(int):void
setY(int):void
getZ():int
isVisible():boolean
setVisible(boolean):void
<<Java Class>>
ScreenManager
core
instance: ScreenManager
nextScreen: Screen
currentScreen: Screen
gameScreen: GameScreen
mainScreen: MainScreen
aboutScreen: AboutScreen
canvas: Canvas
changeStep: int
getInstance():ScreenManager
ScreenManager()
getCanvas():Canvas
getGameScreen():GameScreen
new GameScreen():void
getMainScreen():MainScreen
getAboutScreen():AboutScreen
setNextScreen(Screen):void
getCurrentScreen():Screen
update():void
addListener():void
<<Java Class>>
Screen
core
canvas: Canvas
isActive: boolean
step: int
Screen(Canvas)
update():void
isActive():boolean
active():void
inactive():void
getCanvas():Canvas
getObjectAtPos(int,int):Object
drawComponenet():void
<<Java Class>>
AboutLogic
logic
aboutScreen: AboutScreen
AboutLogic(AboutScreen)
updateLogic():void
<<Java Class>>
Main
core
eventMakerThread: Thread
gameLoop: AnimationTimer
Main()
start(Stage):void
stop():void
main(String[]):void
<<Java Class>>
MainScreen
screen
playButton: PlayButton
exitButton: ExitButton
aboutButton: AboutButton
gameTitle: GameTitle
mainLogic: MainLogic
step: int
MainScreen(Canvas)
getMainLogic():MainLogic
draw Componenet():void
draw CenterDiamond():void
getObjectAtPos(int,int):Object
update():void
active():void
inactive():void
<<Java Class>>
GameLogic
logic
gameScreen: GameScreen
shuffle: boolean
GameLogic(GameScreen)
updateLogic():void
getShuffle():boolean
setShuffle(boolean):void
<<Java Class>>
AboutScreenObjectHolder
objectHolder
instance: AboutScreenObjectHolder
AboutScreenObjectHolder()
getInstance():AboutScreenObjectHolder
<<Java Class>>
MainScreenObjectHolder
objectHolder
instance: MainScreenObjectHolder
centerDiamond: int
MainScreenObjectHolder()
getInstance():MainScreenObjectHolder
getCenterDiamond():int
setCenterDiamond(int):void
<<Java Class>>
PausePopUpObjectHolder
objectHolder
instance: PausePopUpObjectHolder
PausePopUpObjectHolder()
getInstance():PausePopUpObjectHolder
<<Java Class>>
ScreenObjectHolder
core
entities: List<ScreenObject>
comparator: Comparator<ScreenObject>
ScreenObjectHolder()
add(ScreenObject):void
sort():void
getEntities():List<ScreenObject>
<<Java Class>>
GameScreenObjectHolder
objectHolder
instance: GameScreenObjectHolder
GameScreenObjectHolder()
getInstance():GameScreenObjectHolder
<<Java Class>>
EndPopUpObjectHolder
objectHolder
instance: EndPopUpObjectHolder
EndPopUpObjectHolder()
getInstance():EndPopUpObjectHolder


<<Java Class>>
PlayButton
button
PlayButton(int,int)
clickAction(int,int):void
isInside(int,int):boolean
draw (GraphicsContext):void
<<Java Class>>
LittleRetryButton
button
LittleRetryButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
AboutButton
button
AboutButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
Button
core
x: int
y: int
step: int
isVisible: boolean
Button(int,int)
getX():int
getY():int
setX(int):void
setY(int):void
getZ():int
isVisible():boolean
setVisible(boolean):void
<<Java Class>>
LittlePlayButton
button
LittlePlayButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
LittlePauseButton
button
LittlePauseButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
LittleHomeButton
button
LittleHomeButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
ExitButton
button
ExitButton(int,int)
isInside(int,int):boolean
draw (GraphicsContext):void
clickAction(int,int):void
<<Java Class>>
DiamondCell
gridObject
DiamondCell(int,int,GridCell)
equals(Object):boolean
clickAction(int,int):void
isInside(int,int):boolean
draw (GraphicsContext):void
<<Java Class>>
ColorCell
gridObject
cellColor: CellColor
ColorCell(int,int,CellColor,GridCell)
getCellColor():CellColor
equals(Object):boolean
draw (GraphicsContext):void
setCellColor(CellColor):void
getCellImage(CellColor):Image
getCellColor(ColorCell):Color
clickAction(int,int):void
isInside(int,int):boolean
<<Java Class>>
Cell
core
row : int
col: int
isDestroyed: boolean
isVisible: boolean
gridCell: GridCell
Cell(int,int,GridCell)
getGridCell():GridCell
getRow ():int
setRow (int):void
getCol():int
setCol(int):void
setAlive():void
isDestroyed():boolean
destroy():void
getZ():int
isVisible():boolean
setVisible(boolean):void
<<Java Class>>
TimeCell
gridObject
TimeCell(int,int,GridCell)
equals(Object):boolean
clickAction(int,int):void
isInside(int,int):boolean
draw (GraphicsContext):void
<<Java Class>>
BottleCell
gridObject
BottleCell(int,int,GridCell)
equals(Object):boolean
draw (GraphicsContext):void
clickAction(int,int):void
isInside(int,int):boolean
<<Java Class>>
GridCell
gridObject
grid: Cell[][]
maxCol: int
maxRow : int
isVisible: boolean
gameScreen: GameScreen
extraAddCell: List<Cell>
GridCell(GameScreen)
addExtraAddCell(Cell):void
shuffle():void
countItemCell():int
getCellInRow (int):ArrayList<Cell>
getCellInCol(int):ArrayList<Cell>
getCellAtPos(int,int):Cell
generateGrid():void
countClickCell():int
getNeighborOf(ColorCell):ArrayList<ColorCell>
getZ():int
draw (GraphicsContext):void
isVisible():boolean
updateIndex():void
update():void
isInside(int,int):boolean
clickAction(int,int):void
getGameScreen():GameScreen
getCellAtIndex(int,int):Cell
setVisible(boolean):void


<<Java Class>>
Constants
util
GAME_NAME: String
MAX_FRAME_PER_SECOND: int
CELL_SIZE: int
CELL_PER_ROW: int
CELL_PER_COL: int
MAX_COMBO: int
MAX_ITEM_IN_GRID: int
MAX_REMAINING_TIME: int
EVENT_MAKER_SLEEP_TIME: int
GRID_SHUFFLE_THRESHOLD: int
COMBO_THRESHOLD: int
DEFAULT_SCREEN_SIZE: Dimension2D
GRID_CELL_MARGIN: Dimension2D
RESUME_BUTTON_MARGIN: Dimension2D
DEFAULT_GRID_SIZE: Dimension2D
DEFAULT_POPUP_SIZE: Dimension2D
DEFAULT_BUTTON_SIZE: Dimension2D
DEFAULT_BUTTON_EXPAND: Dimension2D
DEFAULT_LITTLE_BUTTON_EXPAND: Dimension2D
DEFAULT_LITTLE_BUTTON_SIZE: Dimension2D
DEFAULT_MEDIUM_BUTTON_EXPAND: Dimension2D
DEFAULT_MEDIUM_BUTTON_SIZE: Dimension2D
DEFAULT_Z_CELL: int
DEFAULT_Z_GRID: int
DEFAULT_Z_BUTTON: int
DEFAULT_Z_GAME_TITLE: int
DEFAULT_Z_GAME_STATUS: int
Constants()
<<Java Enumeration>>
CellColor
util
RED: CellColor
BLUE: CellColor
GREEN: CellColor
YELLOW: CellColor
PURPLE: CellColor
CellColor()
getRandom():CellColor
<<Java Enumeration>>
CellItem
util
BOTTLE: CellItem
DIAMOND: CellItem
TIME: CellItem
CellItem()
getRandom():CellItem
<<Java Class>>
InputUtility
util
mouseX: int
mouseY: int
mouseLeftDown: boolean
mouseRightDown: boolean
mouseOnScreen: boolean
mouseLeftLastDown: boolean
mouseRightLastDown: boolean
InputUtility()
getMouseX():int
setMouseX(int):void
getMouseY():int
setMouseY(int):void
isMouseLeftDown():boolean
setMouseLeftDown(boolean):void
isMouseRightDown():boolean
setMouseRightDown(boolean):void
isMouseOnScreen():boolean
setMouseOnScreen(boolean):void
isMouseLeftClicked():boolean
setMouseLeftLastDown(boolean):void
isMouseRightClicked():boolean
setMouseRightLastDown(boolean):void
postUpdate():void
<<Java Class>>
GameTitle
gameScreen
isVisible: boolean
GameTitle(int,int)
clickAction(int,int):void
isInside(int,int):boolean
draw(GraphicsContext):void
<<Java Class>>
Resources
util
mainScreen: Image
aboutScreen: Image
gameScreen: Image
aboutbutton: Image
aboutbutton2: Image
exitbutton: Image
exitbutton2: Image
playbutton: Image
playbutton2: Image
LongStory: Image
soundMainScreen: AudioClip
soundGameScreen: AudioClip
clickButton: AudioClip
clickBox: AudioClip
boom: AudioClip
yellowCell: Image
blueCell: Image
greenCell: Image
purpleCell: Image
redCell: Image
diamondCell: Image
timeCell: Image
bottleCell: Image
littleHomeButton: Image
littleRetryButton: Image
littlePlayButton: Image
littlePauseButton: Image
pauseFont: Font
scoreFont: Font
playCell: Image
aboutCell: Image
exitCell: Image
instance: Resources
Resources()
getInstance():Resources
getClassLoader():ClassLoader
initialize():boolean
loadFonts():void
loadSound():void
loadImage():void
<<Java Class>>
GameStatus
gameScreen
remainingTime: int
score: int
combo: int
isPause: boolean
isVisible: boolean
gameScreen: GameScreen
GameStatus(GameScreen)
getScore():int
clearScore():void
addRemainingTime(int):void
increaseRemainingTime(int):void
decreaseRemainingTime(int):void
setRemainingTime(int):void
decreaseCombo(int):void
increaseCombo(int):void
clearCombo():void
addCombo(int):void
increaseScore(int):void
getRemainingTime():int
getZ():int
draw(GraphicsContext):void
isVisible():boolean
isPause():boolean
pause():void
unpause():void
getCombo():int
getGameScreen():GameScreen
setGameScreen(GameScreen):void
isInside(int,int):boolean
setVisible(boolean):void

Class detailed

Package : core

This package contains interfaces, abstract classes and main for launching an application

Interface : MouseActionable This is an interface, which uses for the classes that have actions from mouse event.	
Method	
+void clickAction(int x , int y) ;	The action when objects are clicked

Interface : ScreenObject Any class that can be drawn on screen must implement this interface.	
Method	
+int getZ() ;	Return the priority for drawing
+boolean isInside(int x, int y) ;	Return true, if a mouse position is inside object
+void draw(GraphicsContext gc) ;	Implement drawing
+boolean isVisible() ;	Return whether to draw this object or not
+void setVisible(boolean visible) ;	Set the isVisible value

Abstract Class : ScreenObjectHolder This is an abstract class for sharing ScreenObject objects.	
Field	
-List<ScreenObject> entities ;	List of ScreenObject objects
-Comparator<ScreenObject> comparator ;	A Comparator to sort entities list by z value
Constructor	
ScreenObjectHolder() ;	This constructor is used for initializing the value for each field as follows; - entities - comparator

Method	
+add(ScreenObject entity) ;	Add a ScreenObject object to entities and call sort() method
+void sort() ;	Sort all objects in entities by z value
+List<ScreenObject> getEntities() ;	Return entities

Abstract Class : Screen

This is an abstract class, which represents all screens in the game.

Field	
#Canvas canvas ;	Drawing pane
#boolean isActive ;	Flag for the screen
#int step ;	Use for implement animation effect in the screen
Constructor	
Screen(Canvas canvas) ;	This constructor is used for initializing the value for each field as follows; - Initialize the values for canvas - Default value for isActive is false - Initializing step is zero
Method	
+void update() ;	An abstract method for updating screen
+void drawComponent() ;	An abstract method for drawing objects
+void active() ;	An abstract method for set screen active
+void inactive() ;	An abstract method for set screen inactive
+boolean isActive() ;	The getter method for isActive
+Canvas getCanvas() ;	The getter method for canvas
+Object getObjectAtPos(int x , int y) ;	Return the object that on the top at mouse position

Abstract Class : Cell (implements Screen Object, MouseActionable) This is an abstract class, which represents all cells in the game.	
Field	
#int row , col ;	The position of cell in the GridCell
#GridCell gridCell ;	The array of cells
#boolean isVisible ;	Represent the visible status for this object
#boolean isDestroyed ;	Represent the destroy status for this object
Constructor	
Cell (int row , int col , GridCell gridCell) ;	This constructor is used for initializing the value for each field as follows; - Initialize all class fields. - Default value for isDestroy is false - Default value for isActive is true
Method	
Getters & setters	The getter and setter method for row, col and isVisible
+GridCell getGridCell() ;	Return gridCell
+int getZ() ;	Return the Constants.DEFAULT_Z_CELL
+boolean isDestroyed()	Return isDestroy
+void destroy() ;	Play boom sound and set isDestroy to true
+void setAlive() ;	Set isDestroy to false

Abstract Class : Button (implements Screen Object, MouseActionable) This is an abstract class, which represents all buttons in the game.	
Field	
#int x, y ;	The button's top-left position for drawing
#int step ;	Use for implement animation effect in the screen
#boolean isVisible ;	Represent the visible status for the button
Constructor	
Button(int x , int y) ;	Initialize x and y value and set step to zero
Method	

Getters & setters	The getter and setter methods for x, y and isVisible If it is not visible, set step to zero
+int getZ() ;	Return the constant z button

Class : ScreenManager This class manages all screens which one should be display on the screen.	
Field	
-ScreenManager instance ;	A singleton instance
-Screen currentScreen ;	The screen which displays on the screen at the present
-Screen nextScreen ;	The screen which displays after currentScreen finishing
-GameScreen gameScreen ;	A game screen as shown in Figure 1.3
-MainScreen mainScreen ;	A main screen as shown in Figure 1.1
-AboutScreen aboutScreen ;	A credit screen as shown in Figure 1.2
-Canvas canvas ;	Drawing pane for all
-int changeStep ;	Step of changing game screens
Constructor	
ScreenManager() ;	This constructor is used for initializing the value for each field as follows; - Initialize the values for canvas using width and height from Constants - Initialize the mainScreen, gameScreen and aboutScreen with canvas value - Set the nextScreen is mainScreen - addListener - Set changeSteo to zero
Method	
Getters & setters	The getter and setter methods for mainScreen, gameScreen, aboutScreen, Canvas, nextScreen and currentScreen (No setter methods for mainScreen, gameScreen, aboutScreen and canvas because they are managed only in ScreenManager)
+void newGameScreen() ;	Initialize the gameScreen
+static ScreenManager getInstance() ;	Return instance
+void update() ;	Responsible for managing the screen sequence.
+void addListener() ;	set MouseEvent to stackpane and canvas

Class : Main It is the main class to manage GUI of the game.	
Field	
-Thread eventMakerThread ;	This thread uses for - Count the number of possible group of cells can be exploded - Consider to shuffle the GridCell to continue playing the game - Accumulate the combo point and put some items cells by adding to GridCell when the point reaches to the threshold value. - Use sleep method to manage the frequency for updating
-AnimationTimer gameLoop ;	A thread for game loop
Method	
+void start() ;	The method overrides from Application - Loading all resources - Initial game logic and ui - Implement eventMaker thread
+void stop() ;	Stop the program when closing an application and ensure that eventMakerThread is stopped
+static void main() ;	Launch an application

Class : NegativeValueException (extends RuntimeException) It is an exception used when number value is less than zero. (use for some variable that cannot have negative value such as combo point, score point.)	
Constructor	
NegativeValueException() ;	Returns an exception message.
NegativeValueException(String string) ;	Return the string in the following format “"NegativeValueException : " + string ”
NegativeValueException(String string , Throwable throwable) ;	Return the string in the following format “"NegativeValueException : " + string + " with : " + throwable.toString() ”
NegativeValueException(Throwable throwable) ;	Return the string in the following format “"NegativeValueException with " + throwable.toString() ”

Package : Screen

This package contains the classes that concerns with screens.

Class : MainScreen(extends Screen) This class manages about the main screen.	
Field	
-PlayButton playButton ;	A play button
-ExitButton exitButton ;	An exit button
-AboutButton aboutButton ;	An about button
-GameTitle gameTitle ;	A game title
-MainLogic mainLogic;	A logic of the main screen
-int step ;	Use for implement animation effect in the screen
Constructor	
MainScreen (Canvas canvas) ;	This constructor is used for initializing the value for each field as follows; - Initialize the values for canvas - Clear objects in MainScreenObjectHolder - Initialize the mainLogic, aboutButton, playButton, exitButton, gameTitle and add all into MainScreenObjectHolder - Set default center diamond picture on the main screen
Method	
Getter	The getter method for mainLogic
+void drawComponent() ;	- Paint the white background on the canvas - Drawing mainScreen background on the canvas - Drawing all visible objects in the MainScreenObjectHolder entities list
+void update() ;	Update the progress of the main screen. - Call drawComponent() method - Update mainLogic - Objects transparent depend on step value
+Object getObjectAtPos(int x , int y) ;	Return the object that on the top at mouse position
+void drawCenterDiamond() ;	Drawing different diamond pictures that depend on what the mouse position is inside on the main screen
+void active() ;	- Make this screen active - Set step value to zero

	- Play MainScreen sound
+void inactive() ;	- Make this screen non active - Set visible value for all ScreenObject in MainScreenObjectHolder to false - Stop playing MainScreen sound

Class : GameScreen(extends Screen) This class manages about the game screen.	
Field	
-GridCell gridCell ;	The array of cells
-GameStatus gamestatus ;	A game status of the game
-GameLogic gameLogic ;	A logic of the game screen
-int frameCount ;	The number of frame
-int pauseStep ;	For draw pause popup animation
-int endScore ;	The score of player that show in end game popup (to implement score animation)
-LittlePauseButton littlePauseButton ;	A little pause button
Constructor	
GameScreen(Canvas canvas) ;	This constructor is used for initializing the value for each field as follows; - Initialize the values for canvas - Clear objects in MainScreenObjectHolder - Initialize the gridCell, gameStatus, gameLogic and littlePauseButton - Call method createPausePopUp() and createEndPopUp() - Initializing frameCount, pauseStep and endScore to be zero - Pause the gameStatus
Method	
Getters	The getter methods for gridCell, gameStatus, gameLogic and littlePauseButton
+void update();	Update the progress of the game screen - Call drawComponent() method - Determine the pauseStep for drawing EndPopUp or PausePopUp - Update gameLogic
+Object getObjectAtPos(int x , int y) ;	Return the object that on the top at mouse position

+void active() ;	- Make this screen active - Unpause the game screen - Play GameScreen sound
+void inactive() ;	- Make this screen non active - Stop playing GameScreen sound
+void createPausePopUp() ;	- Clear all objects in PausePopUpObjectHolder - Create LittleHomeButton, LittleRetryButton and LittlePlayButton - Add three button into PausePopUpObjectHolder
+createEndPopUp() ;	- Clear all objects in EndPopUpObjectHolder - Create LittleHomeButton and LittleRetryButton - Add two button into EndPopUpObjectHolder
+drawPopUpBG(int step) ;	Draw pop-up background with transparent effect
+drawPausePopUP() ;	- Drawing all visible objects in the PausePopUpScreenObjectHolder entities list - Drawing the text "GAME IS PAUSE"
+drawEndPopUP() ;	- Drawing all visible objects in the EndPopUpScreenObjectHolder entities list - Drawing the game over and score on the screen
+drawComponenet() ;	- Paint the white background on the canvas - Drawing gameScreen background - Drawing all visible objects in the GameScreenObjectHolder entities list

Class : AboutScreen(extends Screen) This class manages about the about screen.	
Field	
-AboutLogic aboutLogic ;	A logic of the about screen
Constructor	
AboutScreen(Canvas canvas) ;	This constructor is used for initializing the value for each field as follows; - Initialize the values for canvas - Clear objects in AboutScreenObjectHolder - Initialize the aboutLogic - Create LittleHomeButton and add into AboutScreenObjectHolder
Method	
Getter	The getter method for aboutLogic
+void drawComponent() ;	- Paint the white background on the canvas - Drawing aboutScreen background - Drawing all visible objects in the AboutScreenObjectHolder entities list
+void update() ;	Update the progress of the about screen - Call drawComponent() method - Update aboutLogic
+Object getObjectAtPos(int x , int y) ;	Return the object that on the top at mouse position
+void active() ;	Make this screen active
+void inactive() ;	Make this screen non active

Package : logic

This package contains the classes that concerns with logic.

Class : MainLogic This class is logic for update all objects in main screen.	
Field	
-MainScreen mainScreen ;	A main screen
Constructor	
MainLogic(MainScreen mainScreen) ;	Initialize a main screen
Method	
+void updateLogic() ;	If click on the object, Call clickAction() method

Class : GameLogic This class is logic for update all objects in game screen.	
Field	
-GameScreen gameScreen ;	A game screen
-boolean shuffle ;	Represent the shuffle status
Constructor	
GameLogic(GameScreen gameScreen) ;	Initialize a game screen and default shuffle value is false
Method	
+void updateLogic() ;	If click on the object, Call clickAction() method If shuffle is true - Call shuffle() method - Set shuffle to false Update the GridCell
+boolean getShuffle();	The getter methods for Shuffle
+synchronized void setShuffle(boolean shuffle);	The setter methods for Shuffle (This method is used in GridCell and eventMakerThread)

Class : AboutLogic This class is logic for update all objects in about screen.	
Field	
-AboutScreen aboutScreen ;	A about screen
Constructor	
AboutLogic(AboutScreen aboutScreen) ;	Initialize a about screen
Method	
+void updateLogic() ;	If click on the object, Call clickAction() method

Package : objectHolder

This package contains the different kinds of objectHolder.

Class : MainScreenObjectHolder (extends ScreenObjectHolder) This class represents a singleton which contains all main screen objects.	
Field	
-MainScreenObjectHolder instance ;	A singleton instance
-int centerDiamond ;	The sequence of diamond images
Constructor	
MainScreenObjectHolder() ;	Call the ScreenObjectHolder constructor
Method	
+static MainScreenObjectHolder getInstance() ;	Return instance
Getters & setters	The getter and setter methods for centerDiamond

Class : GameScreenObjectHolder (extends ScreenObjectHolder) This class represents a singleton which contains all game screen objects.	
Field	
-GameScreenObjectHolder instance ;	A singleton instance
Constructor	
GameScreenObjectHolder() ;	Call the ScreenObjectHolder constructor
Method	
+static GameScreenObjectHolder getInstance() ;	Return instance

Class : AboutScreenObjectHolder (extends ScreenObjectHolder) This class represents a singleton which contains all about screen objects.

Field	
-AboutScreenObjectHolder instance ;	A singleton instance
Constructor	
AboutScreenObjectHolder() ;	Call the ScreenObjectHolder constructor
Method	
+static AboutScreenObjectHolder getInstance() ;	Return instance

Class : PausePopUpObjectHolder (extends ScreenObjectHolder) This class represents a singleton which contains all pause pop up objects.	
Field	
-PausePopUpObjectHolder instance ;	A singleton instance
Constructor	
PausePopUpObjectHolder() ;	Call the ScreenObjectHolder constructor
Method	
+static PausePopUpObjectHolder getInstance() ;	Return instance

Class : EndPopUpObjectHolder (extends ScreenObjectHolder) This class represents a singleton which contains all end pop up objects.	
Field	
-EndPopUpObjectHolder instance ;	A singleton instance
Constructor	
EndPopUpObjectHolder() ;	Call the ScreenObjectHolder constructor
Method	
+static EndPopUpObjectHolder getInstance() ;	Return instance

Package : object.button

This package contains all buttons in the game

Class : PlayButton (extends Button) This class represents a play button in the game.	
Constructor	
PlayButton (int x, int y) ;	Create new a play button and initialize its position by given parameters. - Set isVisible to false
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and change the current screen to the gameScreen
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside PlayButton
+void draw(GraphicsContext gc) ;	If PlayButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the playButton2.png. - If isInside() is false, Draw the playButton.png.

Class : ExitButton (extends Button) This class represents an exit button in the game.	
Constructor	
ExitButton (int x, int y) ;	Create new an exit button and initialize its position by given parameters. - Set isVisible to false
Method	
+void clickAction(int x , int y) ;	Pop-up alert(Ask do you want to close the program) this method we must use Platfrom.runlater() to create alert dialog.
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside ExitButton
+void draw(GraphicsContext gc) ;	If ExitButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the playButton2.png. - If isInside() is false, Draw the playButton.png.

Class : AboutButton (extends Button) This class represents an about button in the game.	
Constructor	
AboutButton (int x, int y) ;	Create new an about button and initialize its position by given parameters. - Set isVisible to false
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and change the current screen to the aboutScreen
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside AboutButton
+void draw(GraphicsContext gc) ;	If AboutButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the aboutButton2.png. - If isInside() is false, Draw the aboutButton.png.

Class : LittleHomeButton (extends Button) This class represents a little home button in the game.	
Constructor	
LittleHomeButton (int x, int y) ;	Create new a little home button and initialize its position by given parameters. - Set isVisible to true
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and change the current screen to the mainScreen
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside LittleHomeButton
+void draw(GraphicsContext gc) ;	If LittleHomeButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the littleHomeButton(Big size) - If isInside() is false, Draw the littleHomeButton

Class : LittlePauseButton (extends Button) This class represents a little pause button in the game.	
Constructor	
LittlePauseButton (int x, int y) ;	Create new a little pause button and initialize its position by given parameters. - Set isVisible to true
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and pause the game screen
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside LittlePauseButton
+void draw(GraphicsContext gc) ;	If LittlePauseButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the littlePauseButton(Big size) - If isInside() is false, Draw the littlePauseButton

Class : LittlePlayButton (extends Button) This class represents a little play button in the game.	
Constructor	
LittlePlayButton (int x, int y) ;	Create new a little play button and initialize its position by given parameters. - Set isVisible to true
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and resume the game
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside LittlePlayButton
+void draw(GraphicsContext gc) ;	If LittlePlayButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the littlePlayButton(Big size) - If isInside() is false, Draw the littlePlayButton

Class : LittleRetryButton (extends Button) This class represents a little retry button in the game.	
Constructor	
LittleRetryButton (int x, int y) ;	Create new a little retry button and initialize its position by given parameters. - Set isVisible to true
Method	
+void clickAction(int x , int y) ;	Play the clickButton sound and restart the game on game screen
+boolean isInside(int posX , int posY);	Return true, if the mouse position is inside LittleRetryButton
+void draw(GraphicsContext gc) ;	If LittleRetryButton is visible - Objects transparent depend on step value - If isInside() is true, Draw the littleRetryButton png.(Big size) - If isInside() is false, Draw the littleRetryButton.png.

Package : object.gameScreen

This package contains game title and game status

Class : GameTitle (extends Button) This class represents the game title on the main screen.	
Field	
-boolean isVisible ;	Represent the visible status for this object
Constructor	
GameTitle (int x, int y) ;	Create new game title and initialize its position by given parameters. - Default value for isVisible is false
Method	
Getters & setters	The getter and setter methods for isVisible
+void clickAction(int x , int y) ;	Do nothing
+boolean isInside(int posX , int posY) ;	Return false
+void draw(GraphicsContext gc) ;	Draw the game title

Class : GameStatus (implements ScreenObject) This class represents all status in the game screen such as score, combo.	
Field	
-int remainingTime , score , combo ;	The status in the game
-boolean isPause , isVisible ;	Represent the pause and visible status
-GameScreen gameScreen ;	A game screen
Constructor	
GameStatus(GameScreen gameScreen) ;	Initialize a game screen and set parameter - Default value for remainingTime is Constants.MAX_REMAINING_TIME - Default value for isPause is false - Default value for isVisible is true - Clear score and combo to be zero
Method	

Getters & setters	The getter and setter methods for isVisible and gameScreen
+int getZ() ;	Return the Constants.DEFAULT_Z_GAME_STATUS
+boolean isInside(int x, int y) ;	Return false for every condition
+void draw(GraphicsContext gc) ;	This method use for draw , Score ,Combo bar , Time remaining bar
+int getScore() ;	Return score
+void clearScore() ;	Clear score to be zero
+synchronized void addRemainingTime(int addRemainingTime) ;	To add some value to a remainingTime variable and control its upperbound and lowerbound . Moreover use synchronized method to be safe with multithread. Remark this method will make more easier to implement other method about modify a value of remainingTime through this method
+void increaseRemainingTime(int increaseRemainingTime) ;	Increase value of remainingTime by addRemainingTime method with parameter "increaseReamingTime"
+void decreaseRemainingTime(int decreaseRemainingTime) ;	Decrease value of remainingTime by addRemainingTime method with parameter "-decreaseReamingTime"
+void setRemainingTime(int remainingTime) ;	Set value of remainingTime by addRemainingTime method with parameter "-x+remainingTime" where x is a current remaingTime (by getRemaingTime method)
+void decreaseCombo(int decreaseCombo) ;	Decrease value of combo by addCombo method with parameter "-decreaseCombo"
+void increaseCombo(int increaseCombo) ;	increase value of combo by addCombo method with parameter "increaseCombo"
+void clearCombo();	Set value of combo to be zero by addCombo method with parameter "-x" where x is a currentCombo (by getCombo method)
+synchronized void addCombo(int addCombo) ;	To add some value to combo variable and control its upperbound and lowerbound . Moreover use synchronized method to be safe with multithread. Remark this method will make more easier to implement other method about modify a value of combo through this method
+void increaseScore(int increaseScore) ;	Use to increase score
+int getRemainingTime() ;	Return remainingTime
+boolean isPause() ;	Return pause status
+void pause() ;	Set pause status to be true
+void unpause() ;	Set pause status to be false
+int getCombo();	Return combo

Package : object.gridObject

This package contains the classes that concerns with cells.

Class : GridCell (implements ScreenObject,MouseActionable) This class represents the array of cells.	
Field	
-Cell grid[][] ;	Cell in GridCell
-int maxCol , maxRow ;	The number of row and column for GRidCell
-boolean isVisible ;	Represent the visible status
-GameScreen gameScreen ;	A game screen
-List<Cell> extraAddCell ;	A synchronized ArrayList to adding special cell after game logic update in each game loop
Constructor	
GridCell(GameScreen gameScreen);	Initialize a game screen and set parameter - Default value for maxCol is Constants.CELL_PER_ROW - Default value for maxRow is Constants.CELL_PER_COL - Create new array of cell (grid[][]) - Default value for isVisible is true - Create a synchronized ArrayList extraAddCell - Call method generateGrid() to initialize each cell of grid
Method	
+void addExtraAddCell(Cell cell) ;	To add a cell to extraAddCell list
+void shuffle() ;	To shuffle all ColorCell in grid by destroy all of them and call method generateGrid() to create new cell in grid
+int countItemCell() ;	Return number of ItemCell in grid
+ArrayList<Cell> getCellInRow(int row) ;	Return a ArrayList of cell in grid which is stay in particular row
+ArrayList<Cell> getCellInCol(int col) ;	Return a ArrayList of cell in grid which is stay in specific column
+Cell getCellAtPos(int x , int y) ;	Return a cell which be on the position x , y in screen
+void generateGrid() ;	To create new ColorCell instead destroyed cell and empty into grid
+int countClickCell() ;	Return number of number of group of cell which can be destroy (a group of ColorCell that have same color and have at least 3 cell wit connection) - To balance the game a bigger group of colorCell and ItemCell will be

	counted with multiple of small group
+ArrayList<ColorCell> getNeighborOf(ColorCell cell) ;	Return a ArrayList of cell which have a same color and connected with it by bfs algorithm
+int getZ() ;	Return Constant.DEFAULT_GRID_CELL_Z
+void draw(GraphicsContext gc);	Use for draw a grid and highlight cell which mouse is point
+boolean isVisible() ;	Return visible status of grid
+void updateIndex() ;	Use to adjust new index of cell when some cell in grid is tranpose or destroyed Remark this method is important because bfs algorithm require actual index of cell
+void update() ;	To update the cell in grid This method consist of 3 part - Remove and shift down a cell above a empty or destroyed cell - Update index cell index in grid through updateIndex method and generate new cell by generateGrid method - Add a cell that queue in extraAddCell list
+boolean isInside(int x, int y) ;	Return true when position (x,y) is in grid otherwise return false
+void clickAction(int x, int y) ;	Find a cell in grid that stay in position x , y and let cell take its action
+GameScreen getGameScreen() ;	Return GameScreen that
+getCellAtIndex(int r , int c) ;	Return a cell that stay on index (r,c) in grid
+void setVisible(boolean visible) ;	Set visible status of grid

Class : ColorCell (extends Cell) This class represents the color cells on the game screen.	
Field	
-CellColor cellColor;	The color cell
Constructor	
ColorCell(int row, int col, CellColor cellColor , GridCell gridCell);	Assign values to fields based on parameters
Method	
Getter	The getter method for cellColor
+boolean equals(Object object) ;	Return true, if an object is the CellColor

+Image getCellImage(CellColor cellColor) ;	To get image that indicate a particular color type
Color getCellColor(ColorCell colorCell) ;	To get color that indicate a particular color type
+void draw(GraphicsContext gc) ;	To draw a image of colorCell
+void clickAction(int x , int y) ;	If this cell has at least two connected cells which have the same color with this cell - Destroy this cell and the same color connected cells
+boolean isInside(int x, int y) ;	Return true, if the mouse position is inside CellColor

Class : DiamondCell (extends Cell) This class represents the item cell that can destroy cells	
Constructor	
DiamondCell(int row, int col, GridCell gridCell) ;	Assign values to fields based on parameters and set isVisible to true
Method	
+boolean equals(Object object) ;	Return true, if an object is the DiamondCell
+void draw(GraphicsContext gc) ;	Draw DiamondCell
+void clickAction(int x , int y) ;	Random destroy cells that have the same row or column position with DiamondCell
+boolean isInside(int x, int y) ;	Return true, if the mouse position is inside DiamondCell

Class : BottleCell (extends Cell) This class represents the item cell that can change the color cell	
Constructor	
BottleCell(int row, int col, GridCell gridCell) ;	Assign values to fields based on parameters and set isVisible to true
Method	
+boolean equals(Object object) ;	Return true, if an object is the BottleCell
+void draw(GraphicsContext gc) ;	Draw BottleCell

+void clickAction(int x , int y) ;	- Random choose one cell from GridCell - Change cells around that cells to be same color
+boolean isInside(int x, int y) ;	Return true, if the mouse position is inside BottleCell

Class : TimeCell (extends Cell) This class represents the item cell that can increase remaining time	
Constructor	
TimeCell(int row, int col, GridCell gridCell) ;	Assign values to fields based on parameters and set isVisible to true
Method	
+boolean equals(Object object);	Return true, if an object is the TimeCell
+void draw(GraphicsContext gc) ;	Draw TimeCell
+void clickAction(int x , int y) ;	Increase remaining time
+boolean isInside(int x, int y) ;	Return true, if the mouse position is inside TimeCell

Package : util

This package contains all constants values and resources in the game. Moreover, it is recording and managing the status of input key and mouse.

Class : Constants This class stores configuration values data of the game screen and contain fields.	
Field	
+static final String GAME_NAME ;	The game name "Long Story"
+static int MAX_FRAME_PER_SECOND ;	Maximun frames per second
+static int CELL_SIZE ;	The size of cell
+static int CELL_PER_ROW ;	The number of cell per row
+static int CELL_PER_COL ;	The number of cell per column
+static int COMBO_THRESHOLD ;	If combo more than this value, Release item cells
+static int MAX_REMAINING_TIME ;	The maximum time in the game
+static int MAX_COMBO ;	The maximum combo in the game
+static int EVENT_MAKER_SLEEP_TIME ;	Sleep time for thread EventMaker
+static int GRID_SHUFFLE_THRESHOLD ;	if the possible cells that can be destroyed are less than this value,Shuffle all cells in GridCell
+static int MAX_ITEM_IN_GRID ;	The maximum item in the GridCell
+static final Dimension2D DEFAULT_SCREEN_SIZE ;	The screen size
+static final Dimension2D GRID_CELL_MARGIN ;	The position for drawing GridCell
+static final Dimension2D DEFAULT_GRID_SIZE ;	The GridCell size
+static final Dimension2D DEFAULT_POPUP_SIZE ;	The pop-up screen size
+static final Dimension2D DEFAULT_BUTTON_SIZE ;	The button size
+static final Dimension2D DEFAULT_BUTTON_EXPAND ;	The increasing size when buttons expand

+static final Dimension2D DEFAULT_LITTLE_BUTTON_EXPAND ;	The little button size
+static final Dimension2D DEFAULT_LITTLE_BUTTON_SIZE ;	The increasing size when buttons expand
+static final Dimension2D DEFAULT_MEDIUM_BUTTON_EXPAND ;	The play button size on the pause screen
+static final Dimension2D DEFAULT_MEDIUM_BUTTON_SIZE ;	The increasing size when play button on the pause screen expands
+static final int DEFAULT_Z_CELL ;	Priority for drawing cells
+static final int DEFAULT_Z_GAME_STATUS ;	Priority for drawing game status
+static final int DEFAULT_Z_GRID ;	Priority for drawing GridCell
Constructor	
Constants () ;	Do nothing

Enum : CellColor All color cells in the game	
Field	
+static final CellColor RED, BLUE, GREEN, YELLOW, PURPLE ;	All color cells on game screen
Constructor	
CellColor() ;	Initialize a color type of cells
Method	
+static CellColor getRandom() ;	Random color cells

Enum : CellItem All item cells in the game	
Field	
+static final CellItem BOTTLE, DIAMOND, TIME;	All item cells on game screen
Constructor	
CellItem() ;	Initialize a item type of cells
Method	
+static CellItem getRandom() ;	Random item cells

Class : InputUtility This class is used for recording and managing the status of the input key and mouse.	
Field	
-static int mouseX, mouseY;	The current mouse position
-static boolean mouseOnScreen;	Check whether or not mouse is on the screen
-static boolean mouseLeftDown, mouseRightDown;	Statuses while pressing mouse
-static boolean mouseLeftLastDown, mouseRightLastDown ;	Another statuses while pressing mouse, but these triggered variables are set to be true only first tick .
Method	
Getters & setters	The getter and setter method for all variables.
+void postUpdate();	This method is used for managing input in the triggering way.

Class : Resource All resources for using in the game	
Field	
Image mainScreen , aboutScreen , gameScreen ;	Background images
Image aboutbutton ,	About button images

aboutbutton2 ;	
Image playbutton , playbutton2 ;	Play button images
Image exitbutton , exitbutton2 ;	Exit button images
Image littleHomeButton , littleRetryButton , littlePlayButton , littlePauseButton ;	All little button images
Image LongStory ;	The gametitle image
AudioClip soundMainScreen, soundGameScreen , clickButton, clickBox , boom ;	All sounds in the game
Image yellowCell , blueCell , greenCell , purpleCell , redCell ;	All cell images
Image diamondCell , timeCell , bottleCell ;	Items images
Image playCell , aboutCell , exitCell ;	The different diamonds picture on the main screen
Font pauseFont , scoreFont ;	All fonts in the game
Constructor	
-Resources instance ;	A singleton instance
Method	
+boolean initialize() ;	Return true when load fonts, sounds and image completely
-ClassLoader getClassLoader() ;	Return ClassLoader
-void loadFonts() ;	Loaded all fonts
-void loadSound() ;	Loaded all sounds
-void loadImage() ;	Loaded all images
+static Resources getInstance() ;	Return instance

How to play

Step 1 : **Download and Open** the LongStory.java.
Step 2 : **Click the "Play" button on the main screen.** The game board will load filled with colorful cells. The players are given 60 seconds to match the colorful cells.
Step 3 : **Match 3 gems.** The simplest match players can make is a connection of three similar boxes. They don't need to be in one row or column. As long as they're connected, the player can match and clear them. Click or tap on the boxes to clear them and get scores.
Step 4 : Watch the clock. There is only 60 seconds per game. The timer is visible on screen so can see how much time you have left.
Step 5 : Item cells. Don't forget to pay attention to item cells since them can help you make score higher.
Step 6 : Once the time is up, the game will end and your score will be shown.

Thanks you