Brick Breaker Code Documentation

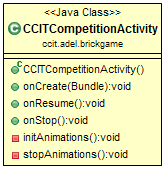
The game consists of 3 activities and 13 classes in total. Each activity has its own role and they are all connected together.

Activities:

* CCITCompetitionActivity
* GameActivity
* HighscoresActivity

1. CCITCompetitionActivity :

This is the main activity which launches with the application

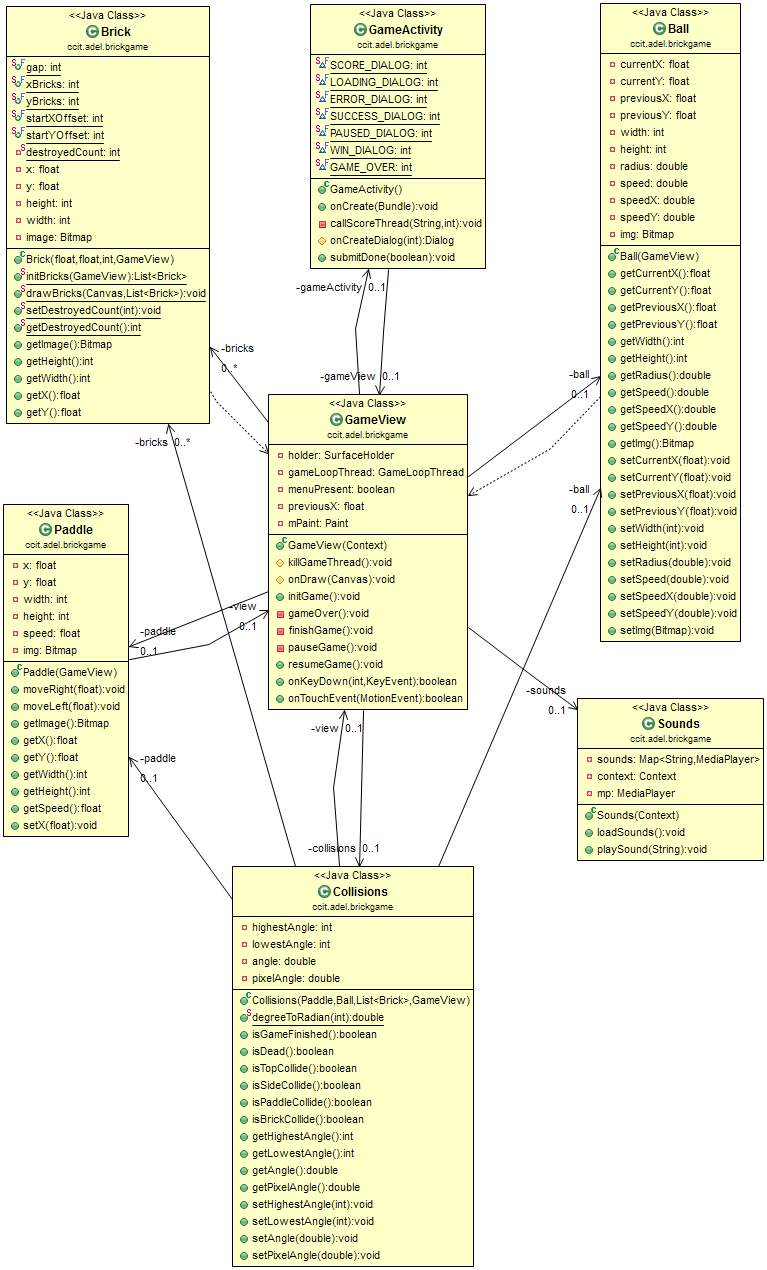


The function initAnimations initializes and starts the animations once the activity is launched

On the contrary the function stopAnimations stops them once the activity is replaced by another activity

1. GameActivity:

This is the activity that gets called when the user clicks play, its role is to control the game logic.

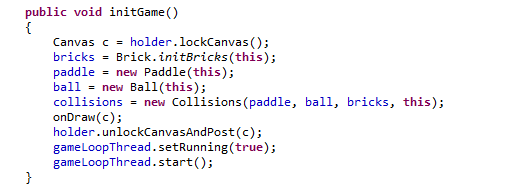


The previous image is the UML diagram of GameActivity with all correspondence classes.

* 1. GameView Class:

This class controls the objects of the game and draws them on the screen. An explanation of the important method of this class and their roles follows:

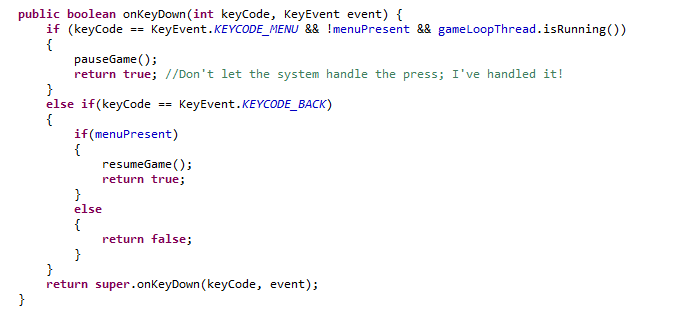
1) initGame:



This function initializes the game and all the required objects; it gets called from the constructor of the GameView class.

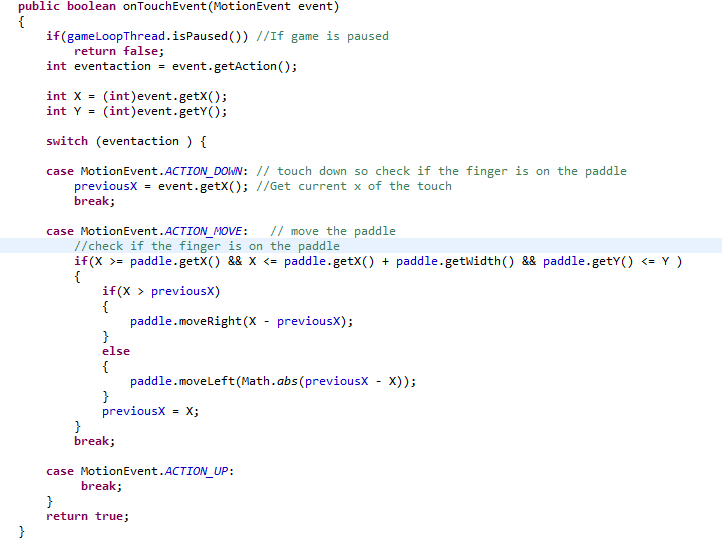
2) onKeyDown:

We check if the user pressed the menu key then we pause the game (if it wasn't already paused) otherwise we let the system handles the press.



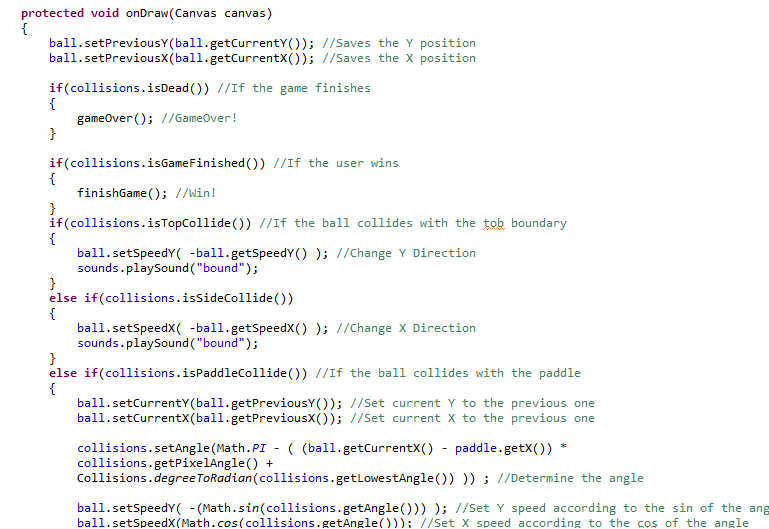
3) onTouchEvent:

This function overrides the method onTouchEvent in the parent class; inside the function we first check if the game is paused, if true then do nothing, else we add the required code to handle the touch event made by the user; we move the paddle relatively with the finger's move.

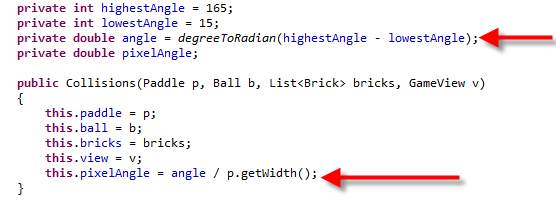


4) onDraw:

This method is the 'brain' of the game; it makes all the required checks and based on this it draws the appropriate frame. We explain some parts of the code of this method and the rest is clear … I hope.



The comments are really handy in our case as they explain almost everything; we check if the current frame causes lose to the user or if there is no more bricks, also we check if the ball collides with any boundary. The most important thing and actually where all the 'art' lies (:D) is when the ball hits the paddle; when you play the game you will notice that depending on the point where the ball hits the paddle; the angle of the ball's move is determined. To achieve this we need to make some calculations; first we get X of both paddle and ball and subtract them to get length between them; then, we multiply the length by a variable called pixelAngle. Following we explain the concept of this variable.



In Collisions class we define 4 variables:

* HighestAngle

This is the highest angle that a ball can move with after it hits the paddle;

* lowestAngle

This is the lowest angle that a ball can move with after it hits the paddle;

* angle

Get the total degrees of the angle (150) and convert them to radian

* pixelAngle

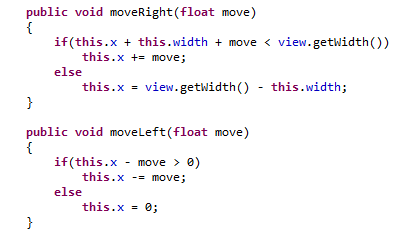
Here in the constructor we assign pixelAngle the value returned from dividing the angle in radian by the width of the paddle; hence, we can calculate the angle that the ball should move with after hitting the paddle on any point correctly.

* 1. Ball Class:

This class represents the ball object with all the necessary methods and attributes like X and Y position, height, and width.

* 1. Paddle Class:

This class represents the paddle object and contains all the necessary methods and attributes of it such as height, width and X, Y position; and moveRight, moveLeft methods that move the paddle in right and left directions.



* 1. Brick Class:

This class represents the brick object, the GameView creates an ArrayList of Brick class that holds as many bricks as required to show the CCIT logo.

* 1. Collisions Class:

This class contains all the methods that determine whether a collision state is present or not. GameView class checks in every frame whether a collision is present or not and hence draws the appropriate frame.

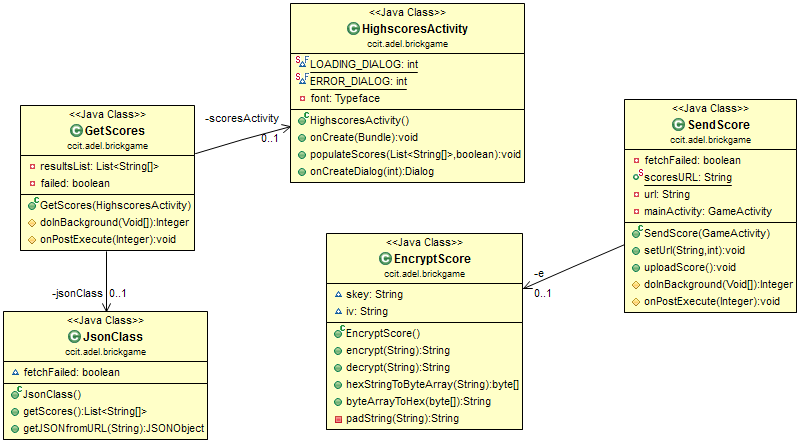
* 1. Sounds Class:

This class contains sound effects that get played when necessary.

g) GameLoopThread Class:

This is the game thread that calls the GameView every frame. This thread keeps executing an infinite loop till the end of the game

3- HighscoresActivity:



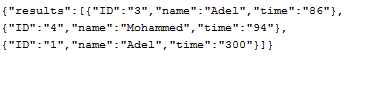
a) GetScores Class:

This class creates an object of JsonClass to retrieve the scores and parses them then it displays the final results. It extends the AsyncTask class which runs a task in the background to prevent blocking the UI main thread, this is the best approach since retrieving the scores might take a relatively long time and therefore blocking the application and making it unresponsive.

b) JsonClass Class:

This class creates an HTTP connection with the server and retrieves a JSON encoded string of the scores.

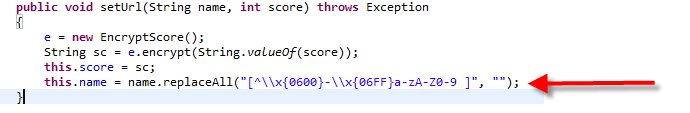
*A sample of JSON encoded string of scores:*



c) SendScore Class:

This class extends the AsyncTask class as well as the previous class for the same reason. In fact it's not a part of the HighScoresActivity as it gets called after the game is finished.

We use Regular Expression to limit the allowed characters; (Allowed characters are: Arabic alphabets [range 0600-06FF in Unicode], all English characters, digits and spaces)



d) EncryptScore Class:

This class plays a very important and critical role as it encrypts the score before sending it to the server, to avoid the risk of malicious users sending false scores to the server. It uses the AES encryption algorithm provided by JAVA's crypto library. A secret key of 128-bits is used to encrypt the scores. (*well, I guess it's not secret anymore xD*)