

Class Pad.java:

The Pad class stores the state of the pad: its size, position and whether it’s interacting with the walls. The class has a method that allows the pad to draw itself. It has methods that return the pad’s position, height and width. Also a method to set its width. The class is very OO, because it has no dependencies and methods are applied to itself, as well as fields being private, with appropriate setters and getters.

Class Ball.java:

The Ball class stores the state of the ball: its size, position and speed. The class has a method that allows the ball to draw itself. The ball is OO as methods are applied on itself, it has no dependencies, as well as having private fields, with appropriate setters and getters.

Class Block.java:

The Block class stores the state of the block: its size, position, health and colour and whether it is destroyed. The class has a method that allows the block to draw itself. It has methods that return the block’s position, size and health. It also has methods that set its position, size and health. Also a method, that changes the block’s colour if it gets damaged, i.e. hit by the ball. The class is very OO, because it has no dependencies and methods are applied to itself, as well as fields being private, with appropriate setters and getters.

Class ErrorDisplay.java:

This class is designed to display an error message to the user in case an error occurs in somewhere in the program. It either displays a default error message or the one passed by another class. It is very OO as it has no dependencies and methods are applied to itself.

Class IOUtil.java:

This class was taken from the LWJGL library. It is used for some I/O functionality, required by the TruetypeOversample class. It has a method for resizing the buffer and a method that reads some data and returns it as a ByteBuffer type. It is very OO, as it has no dependencies.

Class TruetypeOversample.java:

This class was taken from a github project, which is a Java port, which allowed us to use LWJGL’s STBTruetype to input text or graphical elements to a screen that is drawn using OpenGL. This class has methods that give us an option to load custom fonts and display the text we want.

Class ScoreStorage.java:

This class is used for storing the score in a text file on a hard drive. It has method that creates a score.txt file and writes the user’s name and score in it, if the file is already there it appends the name and score. It features a method to read the file or read the file line by line, this method is used when displayed the scores in the RecordsGUI class. It also has a method that enables us to clear the file in case we want to reset the records. Although the fields are private and the methods are applied to itself and do not have any dependencies, this class is not as OO as we could have made it. It has a predefined constructor, making it less flexible. The best way to do it would to have no constructor and all methods static, so as to be referenced from any class without creating an object

Class RecordGUI.java:

This class is employed to show users’ high scores. The constructor is there to simple call the initialization() method, which actually displays the GUI. It uses a JFrame to display the GUI. After creating an empty scoreboard, the class tries to populate it by scanning the “score.txt” file and soring the scores and names in descending order with the respect to the score. The class is able to throw an IOException, but most exceptions are handled within the class. It follows OO principles quite well, most fields are private, but it would be better if it had method separation, as everything is done in the initialization() method

Class AskName.java:

This class is used to ask the user for his/her name, once the game has finished. The class implements KeyListener. The constructor is there to simple call the initialization() method, which actually displays the GUI. It uses a JFrame to display the GUI. It asks the user to input a name into a JTextField. A KeyListener is added to both JFrame and the JTextField. In the beginning the program automatically focus on JFrame,soo that it will listen to the ‘Enter’ key to fire. But when you input something in the JTrxtField, it’ll no longer focus on the JFrame, as a result, pressing ‘Enter’ won’t work. So in the JTextField, once the player finishes inputting and press ‘Enter’, the listener is fired to re-focus on the JFrame to fire the other listener.  (the text, to perform the same action,) upon pressing “Enter”, as the button “Confirm” would. This button retrieves the text from the JTextField and attempts to write into a “score.txt” file. If the user has not put a name in but he still clicks ‘confirm’, he’s score will be stored anyway with a name ‘Naughty kid’. the score will be written with a name and the frame will be disposed of “Naughty Kid”. Another button “Cancel” sets the name to “Tourist” and then disposes of the frame. The class follows OO principles well, handles exceptions, but some of the methods could be taken out of the initialize() method. The class relies only on ScoreStorage.java class to write the scores and names to the files.

Class BallCollisionMechanics.java:

The class is designed to handle most of the game’s mechanics: how the ball interacts with walls, blocks and the pad. It takes almost all the objects that are displayed on the game screen as arguments. The class makes the whole program more OO, because it removes some functionality from the BlockBreakerMain.java class, but is itself quite dependent on other classes and might be difficult to extend, because of its complexity.

Class BlockBreakerApp.java:

This class has our main() method and is used to start the whole program. Its constructor is there just to start the initialize() method. The initialize() method uses a JFrame to display the game menu. It creates a JButton for the 3 option we have. “Start” calls the BlockBreakerMain(), which is our actual game passes it a SettingGUI type object. This is done in such a way because SettingGUI sets the speed of the ball(s). Buttons “Records” and “Setting” calling RecordGUI() and setting settingGUI to visible respectively.

Class ColourOpenGL.java

Class BlockBreakerMain.java

Class AudioFile.java