Tetris on Android

Team

Nir Boneh (nir.boneh@colorado.edu)
Shirong Bai (shirong.bai@colorado.edu)
Lili Ji (lili.ji@colorado.edu)

Project Summary

We three developed a classic game of Tetris on the android system, taking advantage of what we learned in class.

1. Features implemented

- User open the app can see the main menu, shows name of the game, and also see four buttons, “Play”, “HighScores”, “Help” and “Settings”, as in Figure 1.

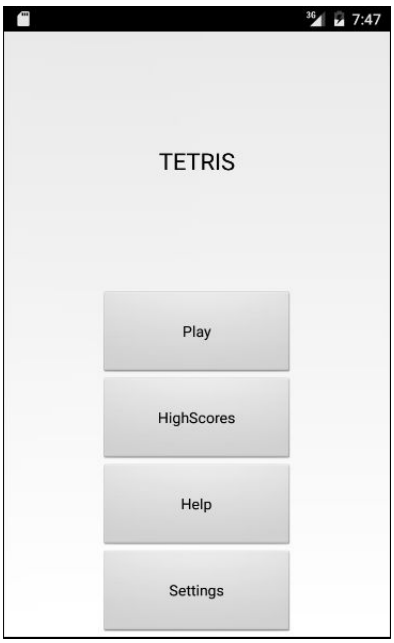


Figure 1.

Figure 2.

Figure 3.

- User click “HighScores” button, high-scores window pops out, shows the “Local” and “Onine” high scores with user name on the left, following by the scores on the right of the screen, in descending order, as in Figure 2.
- User click “Help” button, help screen pops out, which is a image viewer, user can the help information, like the scoring rules, the button functionalities, and layout, etc. like in Figure 3.

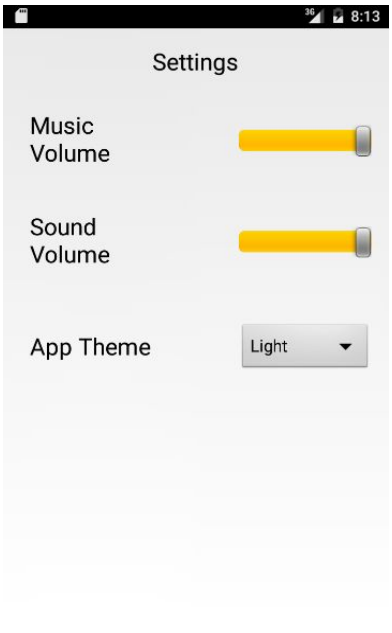


Figure 4.

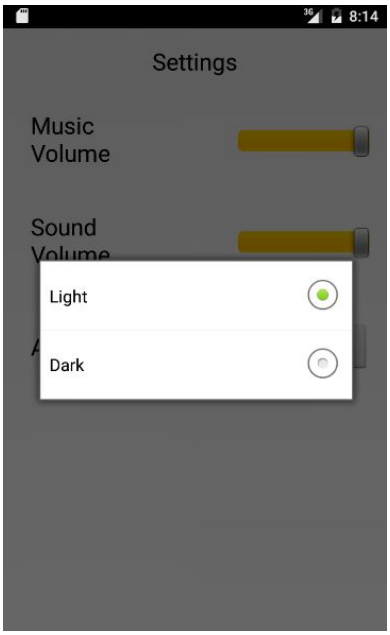


Figure 5.

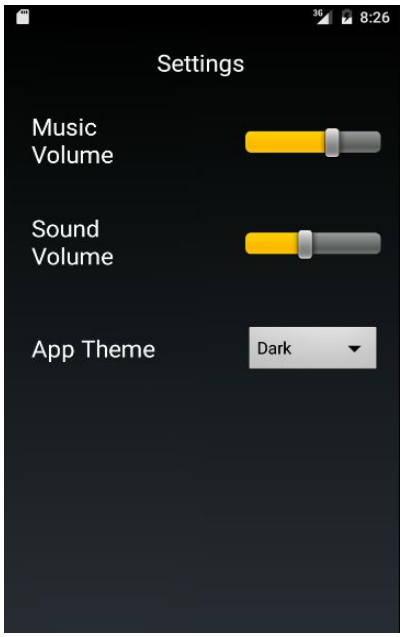


Figure 6.

- User click “Settings”, setting window pops out, where user can set the “Music Volume”, “Sound Voluem” and the App Theme. (See Figure 4.) We have two themes availble, one is light theme, the other is dark theme, as in Figure 5 and Figure 6.

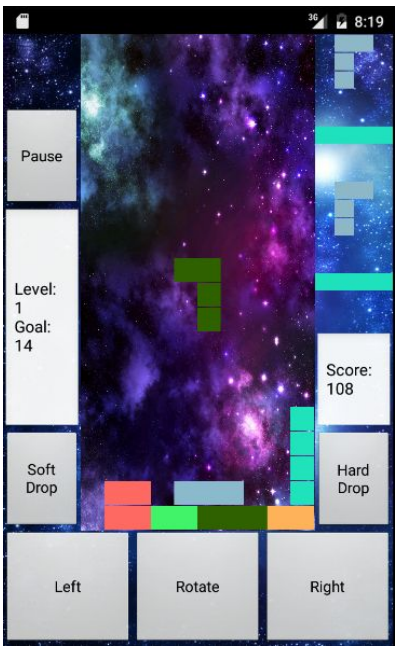


Figure 7.

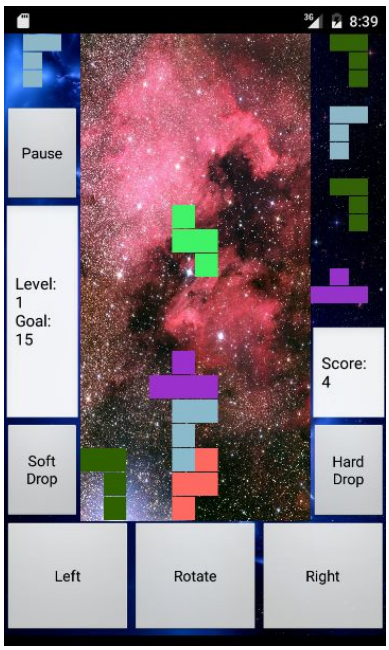


Figure 8.

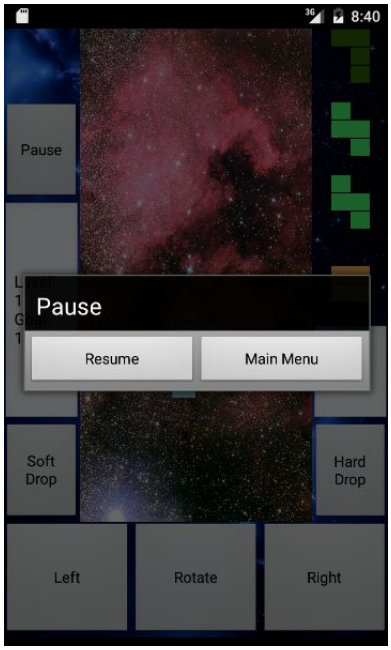


Figure 9.

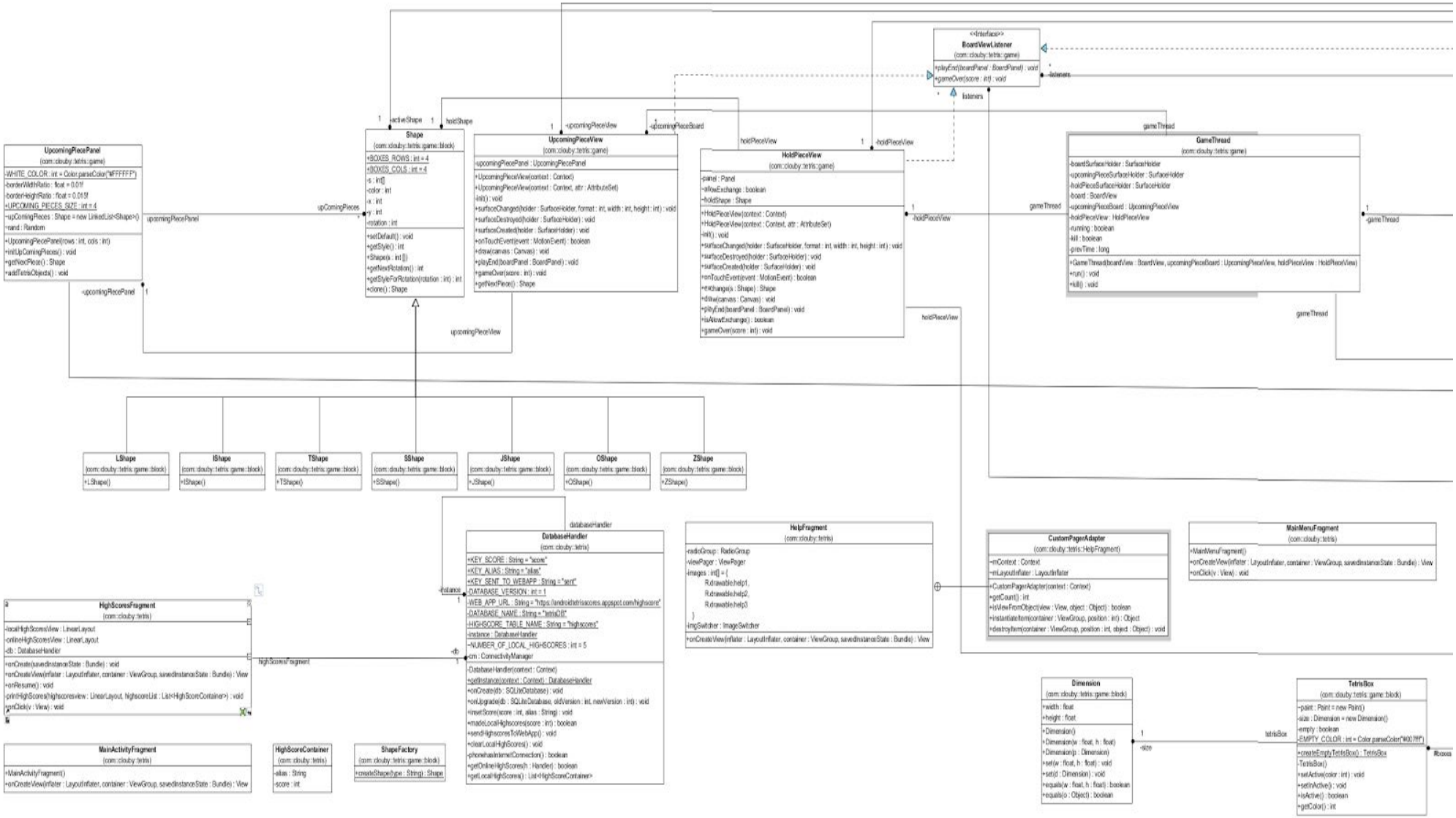
- Use click the “Play” button, the game window pops
- In the game window, Figure 7, the middle big area holds the game view, which has a game panel bascically implements canvas, we draw the background and tetris pieces on it, as in Figure 7, different tetris shape has different colors.
    - The left top corner is the piece holder, user can use it to hold one tetris shape.
    - The right top corner is the next four upcoming tetris shape.

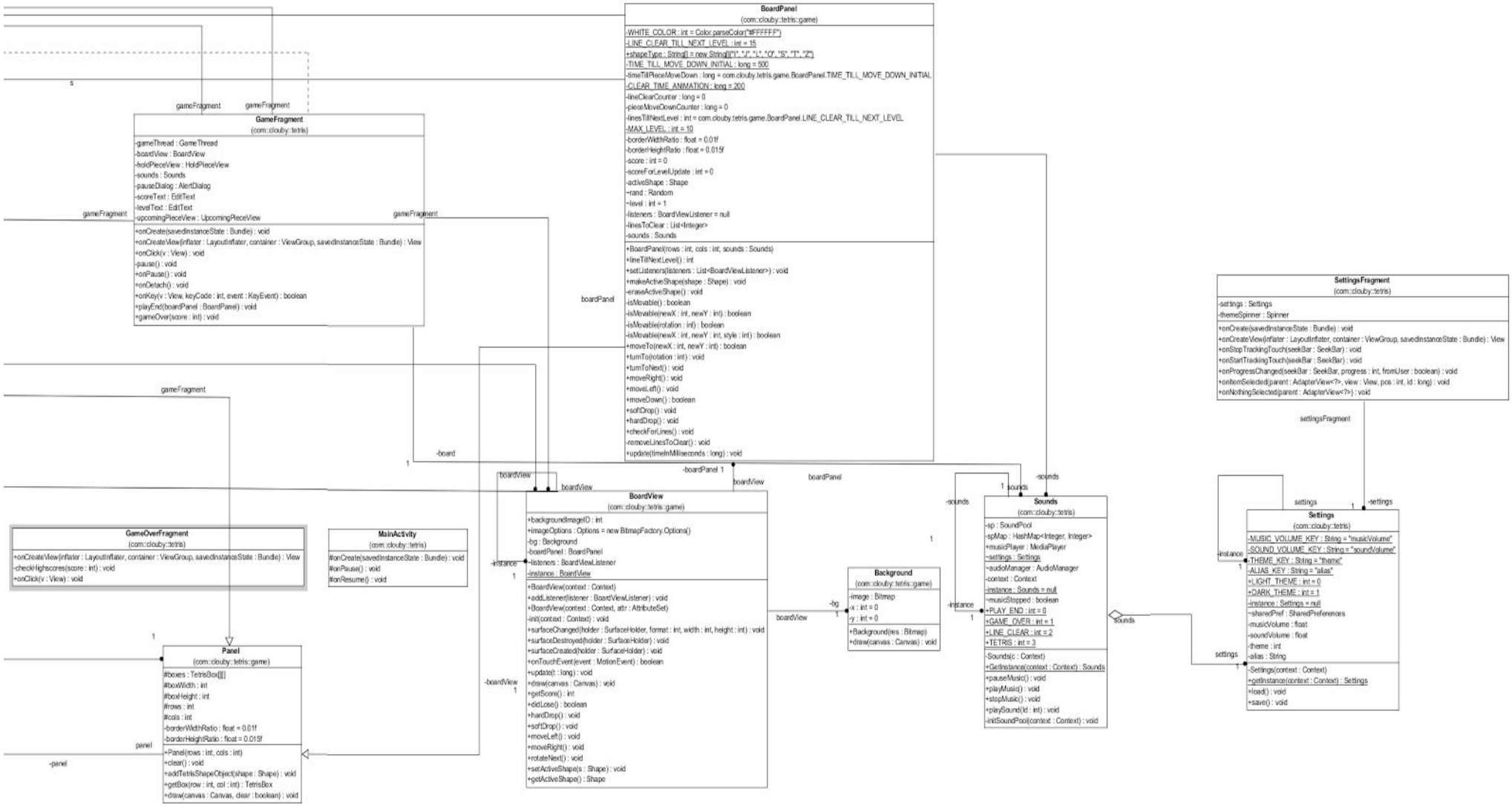
    - If user click “Pause” button, the game will be paused, a pause window pops out, ask the user to resume the game or quit and back to main menu, as in Figure 9.
    - The level panel shows the current level, the goal show how many lines need to reach next level.
    - The score panel shows the current score
    - Click “Left” button will move the tetris piece one box left
    - Click “Right” button will move the tetris piece one box right
    - Click “Soft Drop” button will drop the tetris quickly
    - Click “Hard Drop” button will drop the tetris immediately
    - Click “Rotate” button will rotate the tetris following clockwise direction

- Basically, our goal is to apply what we learn from this class CSCI 5448 (Objected-Oriented Analysis & Design) to a real project. We started from learning basic java syntax, learning OOP features. After that, we did requirement analysis on our Tetris game, wrote use-case diagram, sequence diagram, state diagram and most importantly, designed class diagram. Then we started to develop the application based on the class diagram and we also refactored.

Final class digram:

our class diagram is complicated, we divided it into two parts.

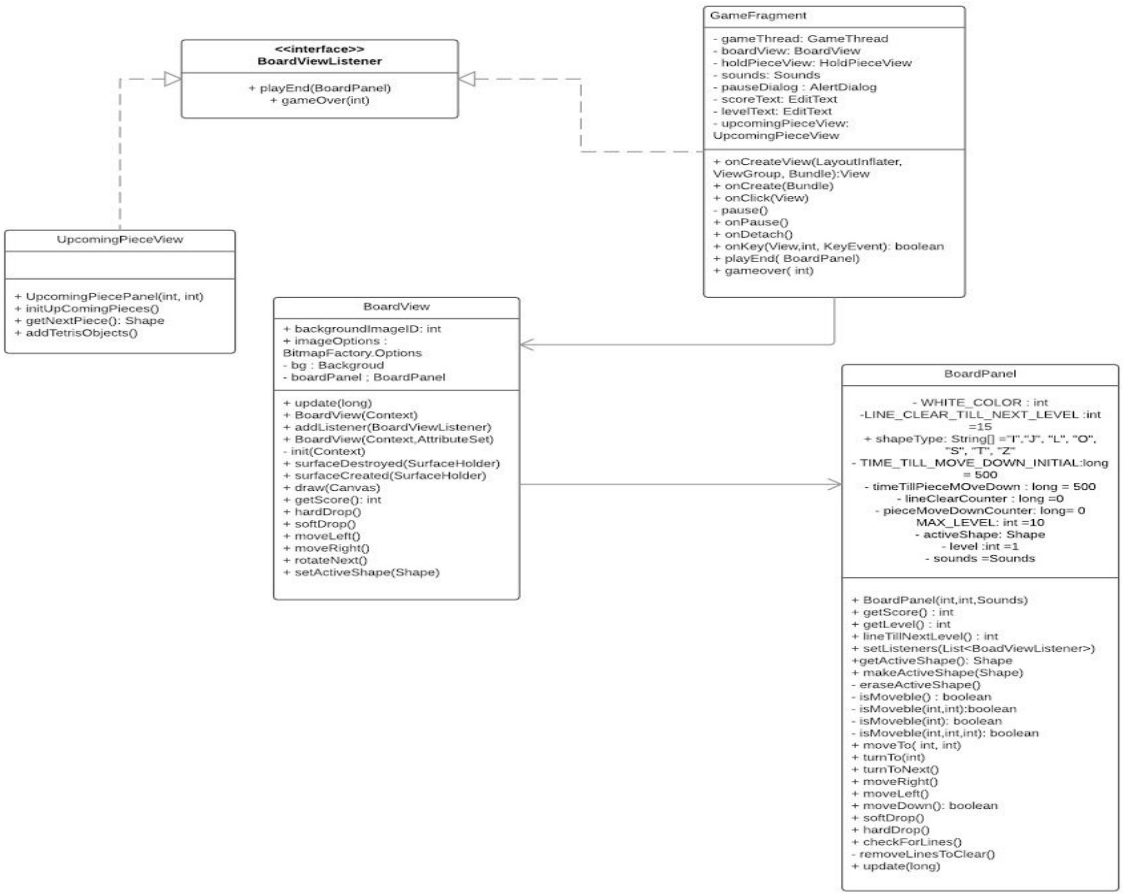

UpcomingPiecePanel
(com::clouby::tetris::game)
-WHITE_COLOR : int = Color.parseColor("#FFFFFF")
-borderWidthRatio : float = 0.01f
-borderHeightRatio : float = 0.01f
+UPCOMING_PIECES_SIZE : int = 4
-upComingPieces : Shape = new LinkedList<Shape>()
-rand : Random
+UpcomingPiecePanel(rows : int, cols : int)
+initUpComingPieces() : void
+getNextPiece() : Shape
+addTetrisObjects() : void
upcomingPiecePanel
upComingPieces
Shape
(com::clouby::tetris::game::block)
+BOXES_ROWS : int = 4
+BOXES_COLS : int = 4
-s : int[]
-color : int
-x : int
-y : int
-rotation : int
+setDefault() : void
+getStyle() : int
+Shape(s : int [])
+getNextRotation() : int
+getStyleForRotation(rotation : int) : int
+clone() : Shape
activeShape
holdShape
upcomingPieceView
upcomingPieceBoard
UpcomingPieceView
(com::clouby::tetris::game)
-upcomingPiecePanel : UpcomingPiecePanel
+UpcomingPieceView(context : Context)
+UpcomingPieceView(context : Context, attr : AttributeSet)
-init() : void
+surfaceChanged(holder : SurfaceHolder, format : int, width : int, height : int) : void
+surfaceDestroyed(holder : SurfaceHolder) : void
+surfaceCreated(holder : SurfaceHolder) : void
+onTouchEvent(event : MotionEvent) : boolean
+draw(canvas : Canvas) : void
+playEnd(boardPanel : BoardPanel) : void
+gameOver(score : int) : void
+getNextPiece() : Shape
upcomingPieceView
holdPieceView
HoldPieceView
(com::clouby::tetris::game)
-panel : Panel
-allowExchange : boolean
-holdShape : Shape
+HoldPieceView(context : Context)
+HoldPieceView(context : Context, attr : AttributeSet)
-init() : void
+surfaceChanged(holder : SurfaceHolder, format : int, width : int, height : int) : void
+surfaceDestroyed(holder : SurfaceHolder) : void
+surfaceCreated(holder : SurfaceHolder) : void
+onTouchEvent(event : MotionEvent) : boolean
+exchange(a : Shape) : Shape
+draw(canvas : Canvas) : void
+playEnd(boardPanel : BoardPanel) : void
+isAllowExchange() : boolean
+gameOver(score : int) : void
holdPieceView
<<Interface>>
BoardViewListener
(com::clouby::tetris::game)
+playEnd(boardPanel : BoardPanel) : void
+gameOver(score : int) : void
listeners
gameThread
GameThread
(com::clouby::tetris::game)
-boardSurfaceHolder : SurfaceHolder
-upcomingPieceSurfaceHolder : SurfaceHolder
-holdPieceSurfaceHolder : SurfaceHolder
-board : BoardView
-upcomingPieceBoard : UpcomingPieceView
-holdPieceView : HoldPieceView
-running : boolean
-kill : boolean
-prevTime : long
+GameThread(boardView : BoardView, upcomingPieceBoard : UpcomingPieceView, holdPieceView : HoldPieceView)
+run() : void
+kill() : void
LShape
(com::clouby::tetris::game::block)
+LShape()
IShape
(com::clouby::tetris::game::block)
+IShape()
TShape
(com::clouby::tetris::game::block)
+TShape()
SShape
(com::clouby::tetris::game::block)
+SShape()
JShape
(com::clouby::tetris::game::block)
+JShape()
OShape
(com::clouby::tetris::game::block)
+OShape()
ZShape
(com::clouby::tetris::game::block)
+ZShape()
databaseHandler
DatabaseHandler
(com::clouby::tetris)
+KEY_SCORE : String = "score"
+KEY_ALIAS : String = "alias"
+KEY_SENT_TO_WEBAPP : String = "sent"
-DATABASE_VERSION : int = 1
-WEB_APP_URL : String = "https://android-tetris-scores.appspot.com/highscore"
-DATABASE_NAME : String = "tetrisDB"
-HIGHSCORE_TABLE_NAME : String = "highscores"
-instance : DatabaseHandler
~NUMBER_OF_LOCAL_HIGHSCORES : int = 5
-cm : ConnectivityManager
-DatabaseHandler(context : Context)
+getInstance(context : Context) : DatabaseHandler
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
+insetScore(score : int, alias : String) : void
+madeLocalHighscore(score : int) : boolean
+sendHighscoresToWebApp() : void
+clearLocalHighScores() : void
-phoneHasInternetConnection() : boolean
+getOnlineHighScores(h : Handler) : boolean
+getLocalHighScores() : List<HighScoreContainer>
instance
db
HelpFragment
(com::clouby::tetris)
-radioGroup : RadioGroup
-viewPager : ViewPager
-images : int[] = {
R.drawable.help1,
R.drawable.help2,
R.drawable.help3
}
-imgSwitcher : ImageSwitcher
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
CustomPagerAdapter
(com::clouby::tetris::HelpFragment)
-mContext : Context
-mLayoutInflater : LayoutInflater
+CustomPagerAdapter(context : Context)
+getCount() : int
+isViewFromObject(view : View, object : Object) : boolean
+instantiateItem(container : ViewGroup, position : int) : Object
+destroyItem(container : ViewGroup, position : int, object : Object) : void
MainMenuFragment
(com::clouby::tetris)
+MainMenuFragment()
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onClick(v : View) : void
HighScoresFragment
(com::clouby::tetris)
-localHighScoresView : LinearLayout
-onlineHighScoresView : LinearLayout
-db : DatabaseHandler
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onResume() : void
-printHighScores(highscoresview : LinearLayout, highscoreList : List<HighScoreContainer>) : void
+onClick(v : View) : void
highScoresFragment
MainActivityFragment
(com::clouby::tetris)
+MainActivityFragment()
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
HighScoreContainer
(com::clouby::tetris)
-alias : String
-score : int
ShapeFactory
(com::clouby::tetris::game::block)
+createShape(type : String) : Shape
Dimension
(com::clouby::tetris::game::block)
+width : float
+height : float
+Dimension()
+Dimension(w : float, h : float)
+Dimension(p : Dimension)
+set(w : float, h : float) : void
+set(d : Dimension) : void
+equals(w : float, h : float) : boolean
+equals(o : Object) : boolean
size
TetrisBox
(com::clouby::tetris::game::block)
-paint : Paint = new Paint()
-size : Dimension = new Dimension()
-empty : boolean
-EMPTY_COLOR : int = Color.parseColor("#001fff")
+createEmptyTetrisBox() : TetrisBox
-TetrisBox()
+setActive(color : int) : void
+setInActive() : void
+isActive() : boolean
+getColor() : int
tetrisBox


BoardPanel
(com::clouby::tetris::game)
-WHITE_COLOR : int = Color.parseColor("#FFFFFF")
-LINE_CLEAR_TILL_NEXT_LEVEL : int = 15
+shapeType : String[] = new String[]{"I", "J", "L", "O", "S", "T", "Z"}
-TIME_TILL_MOVE_DOWN_INITIAL : long = 500
-timeTillPieceMoveDown : long = com.clouby.tetris.game.BoardPanel.TIME_TILL_MOVE_DOWN_INITIAL
-CLEAR_TIME_ANIMATION : long = 200
-lineClearCounter : long = 0
-pieceMoveDownCounter : long = 0
-linesTillNextLevel : int = com.clouby.tetris.game.BoardPanel.LINE_CLEAR_TILL_NEXT_LEVEL
-MAX_LEVEL : int = 10
-borderWidthRatio : float = 0.01f
-borderHeightRatio : float = 0.015f
-score : int = 0
-scoreForLevelUpdate : int = 0
-activeShape : Shape
~rand : Random
~level : int = 1
-listeners : BoardViewListener = null
-linesToClear : List<Integer>
-sounds : Sounds
+BoardPanel(rows : int, cols : int, sounds : Sounds)
+lineTillNextLevel() : int
+setListeners(listeners : List<BoardViewListener>) : void
+makeActiveShape(shape : Shape) : void
-eraseActiveShape() : void
-isMovable() : boolean
-isMovable(newX : int, newY : int) : boolean
-isMovable(rotation : int) : boolean
-isMovable(newX : int, newY : int, style : int) : boolean
+moveTo(newX : int, newY : int) : boolean
+turnTo(rotation : int) : void
+turnToNext() : void
+moveRight() : void
+moveLeft() : void
+moveDown() : boolean
+softDrop() : void
+hardDrop() : void
+checkForLines() : void
-removeLinesToClear() : void
+update(timeInMilliseconds : long) : void
GameFragment
(com::clouby::tetris)
-gameThread : GameThread
-boardView : BoardView
-holdPieceView : HoldPieceView
-sounds : Sounds
-pauseDialog : AlertDialog
-scoreText : EditText
-levelText : EditText
-upcomingPieceView : UpcomingPieceView
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onClick(v : View) : void
-pause() : void
+onPause() : void
+onDetach() : void
+onKey(v : View, keyCode : int, event : KeyEvent) : boolean
+playEnd(boardPanel : BoardPanel) : void
+gameOver(score : int) : void
BoardView
(com::clouby::tetris::game)
+backgroundImageID : int
+imageOptions : Options = new BitmapFactory.Options()
-bg : Background
-boardPanel : BoardPanel
-listeners : BoardViewListener
-instance : BoardView
+BoardView(context : Context)
+addListener(listener : BoardViewListener) : void
+BoardView(context : Context, attr : AttributeSet)
-init(context : Context) : void
+surfaceChanged(holder : SurfaceHolder, format : int, width : int, height : int) : void
+surfaceDestroyed(holder : SurfaceHolder) : void
+surfaceCreated(holder : SurfaceHolder) : void
+onTouchEvent(event : MotionEvent) : boolean
+update(t : long) : void
+draw(canvas : Canvas) : void
+getScore() : int
+didLose() : boolean
+hardDrop() : void
+softDrop() : void
+moveLeft() : void
+moveRight() : void
+rotateNext() : void
+setActiveShape(s : Shape) : void
+getActiveShape() : Shape
Panel
(com::clouby::tetris::game)
#boxes : TetrisBox[][]
#boxWidth : int
#boxHeight : int
#rows : int
#cols : int
-borderWidthRatio : float = 0.01f
-borderHeightRatio : float = 0.015f
+Panel(rows : int, cols : int)
+clear() : void
+addTetrisShapeObject(shape : Shape) : void
+getBox(row : int, col : int) : TetrisBox
+draw(canvas : Canvas, clear : boolean) : void
Background
(com::clouby::tetris::game)
-image : Bitmap
-x : int = 0
-y : int = 0
+Background(res : Bitmap)
+draw(canvas : Canvas) : void
Sounds
(com::clouby::tetris)
-sp : SoundPool
-spMap : HashMap<Integer, Integer>
+musicPlayer : MediaPlayer
~settings : Settings
~audioManager : AudioManager
-context : Context
-instance : Sounds = null
~musicStopped : boolean
+PLAY_END : int = 0
+GAME_OVER : int = 1
+LINE_CLEAR : int = 2
+TETRIS : int = 3
-Sounds(c : Context)
+GetInstance(context : Context) : Sounds
+pauseMusic() : void
+playMusic() : void
+stopMusic() : void
+playSound(id : int) : void
-initSoundPool(context : Context) : void
SettingsFragment
(com::clouby::tetris)
-settings : Settings
-themeSpinner : Spinner
+onCreate(savedInstanceState : Bundle) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onStopTrackingTouch(seekBar : SeekBar) : void
+onStartTrackingTouch(seekBar : SeekBar) : void
+onProgressChanged(seekBar : SeekBar, progress : int, fromUser : boolean) : void
+onItemSelected(parent : AdapterView<?>, view : View, pos : int, id : long) : void
+onNothingSelected(parent : AdapterView<?>) : void
Settings
(com::clouby::tetris)
-MUSIC_VOLUME_KEY : String = "musicVolume"
-SOUND_VOLUME_KEY : String = "soundVolume"
-THEME_KEY : String = "theme"
-ALIAS_KEY : String = "alias"
+LIGHT_THEME : int = 0
+DARK_THEME : int = 1
-instance : Settings = null
~sharedPref : SharedPreferences
-musicVolume : float
-soundVolume : float
-theme : int
-alias : String
-Settings(context : Context)
+getInstance(context : Context) : Settings
+load() : void
+save() : void
GameOverFragment
(com::clouby::tetris)
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
-checkHighscores(score : int) : void
+onClick(v : View) : void
MainActivity
(com::clouby::tetris)
#onCreate(savedInstanceState : Bundle) : void
#onPause() : void
#onResume() : void
gameFragment
boardPanel
-board
boardView
-boardPanel 1
-sounds
sounds
-instance
-bg
settingsFragment
settings
-settings
-panel
panel
-boardView

2. Design Patterns

We took advatage of three main desing patterns in the final prototype observer, factory, and signleton.

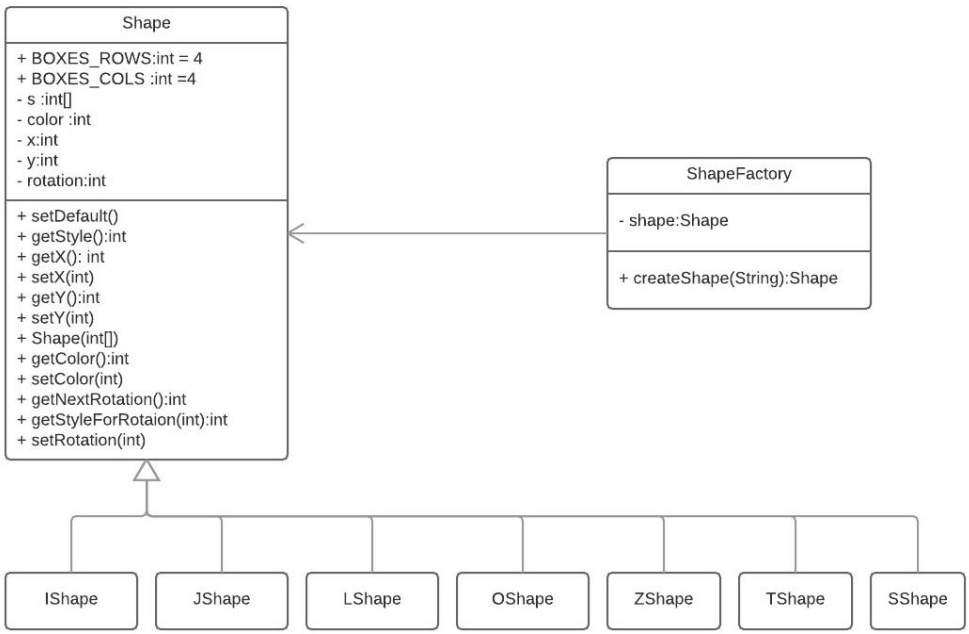
Observer design pattern:



We used observer pattern to listen for button clicks from the user. These are called listeners in android and they are built in. We also used the observer pattern in the database handler to go into a sepertate thread to get the online highscores from the internet and then send back a message to the highscore fragment to show the scores in a list. The biggest thing we used it for was to implement a BoardViewListener which two classes implemented inorder to listen to the events of playEnd and gameOver. The game fragment for example used the gameOver observer function in order to go to the game over fragment and used the playEnd fragment to reset the shapeholder view in order to allow exchange for shapes again. The upcomingpieceview used the playEnd observer function to send a new piece to be played by the boardPanel. We are aware that our implementation of the observer

pattern was defintly abit off from the design pattern used in class, but we did what made sense to us in order to provide events to diffrent instance classes and we derived the design pattern from the one used in class.

Factory design pattern:

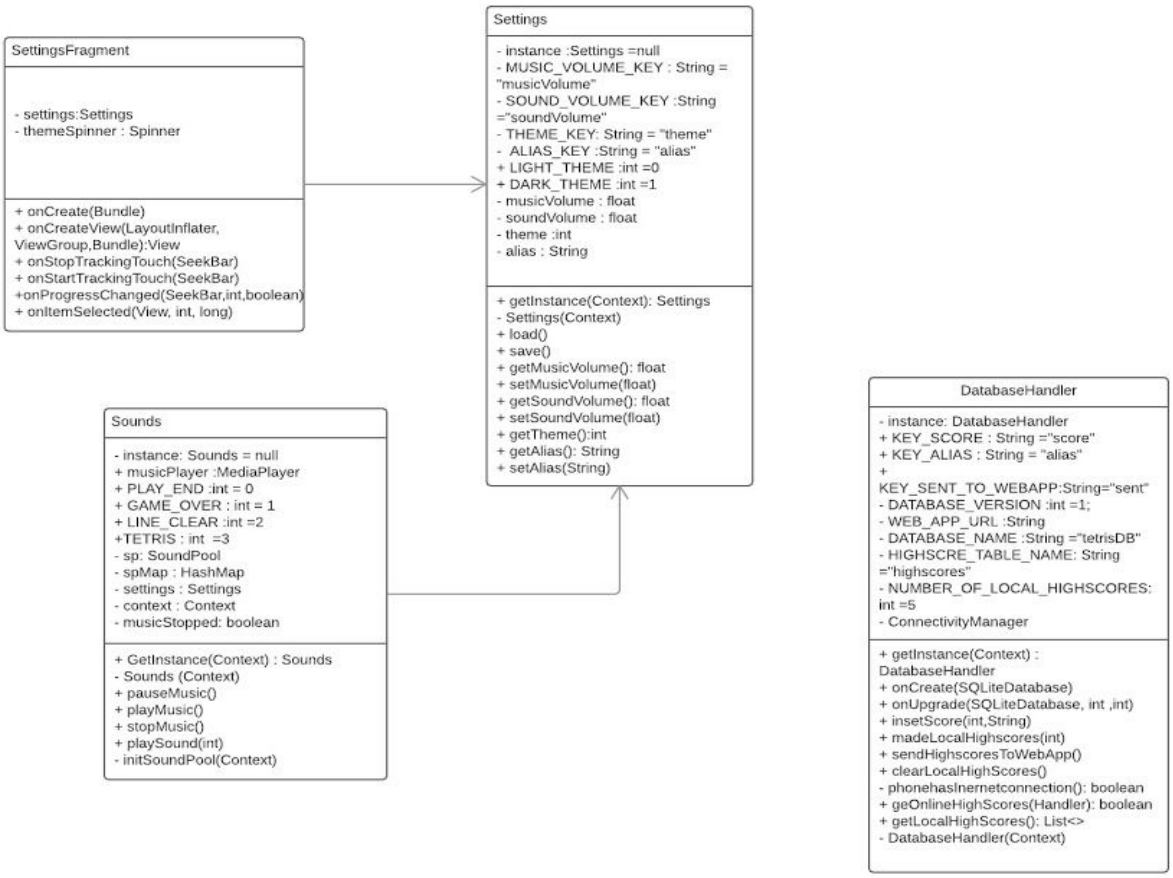



```
public UpcomingPiecePanel(int rows, int cols) {
    super(rows, cols);
    rand = new Random();
    initUpComingPieces();
    addTetrisObjects();
}

public void initUpComingPieces() {
    for (int i = 0; i < UPCOMING_PIECES_SIZE; ++i) {
        upComingPieces.add(ShapeFactory.createShape(BoardPanel.shapeType[rand.nextInt(BoardPanel.shapeType.length)]));
    }
}
```

```java
package com.clouby.tetris.game.block;

public class ShapeFactory {
    public static Shape createShape(String type) {
        Shape shape = null;

        if (type.equals("I")) {
            shape = new IShape();
        } else if (type.equals("J")) {
            shape = new JShape();
        } else if (type.equals("L")) {
            shape = new LShape();
        } else if (type.equals("O")) {
            shape = new OShape();
        } else if (type.equals("S")) {
            shape = new SShape();
        } else if (type.equals("T")) {
            shape = new TShape();
        } else if (type.equals("Z")) {
            shape = new ZShape();
        }

        return shape;
    }

}
```

We also used a factory pattern to generate shapes onto the boardPanel. The upcomingPiecePanel used the ShapeFactory inorder to generate random shapes for its queue and then send a piece over to BoardPanel from the queue every playEnd event.

Singleton design pattern:



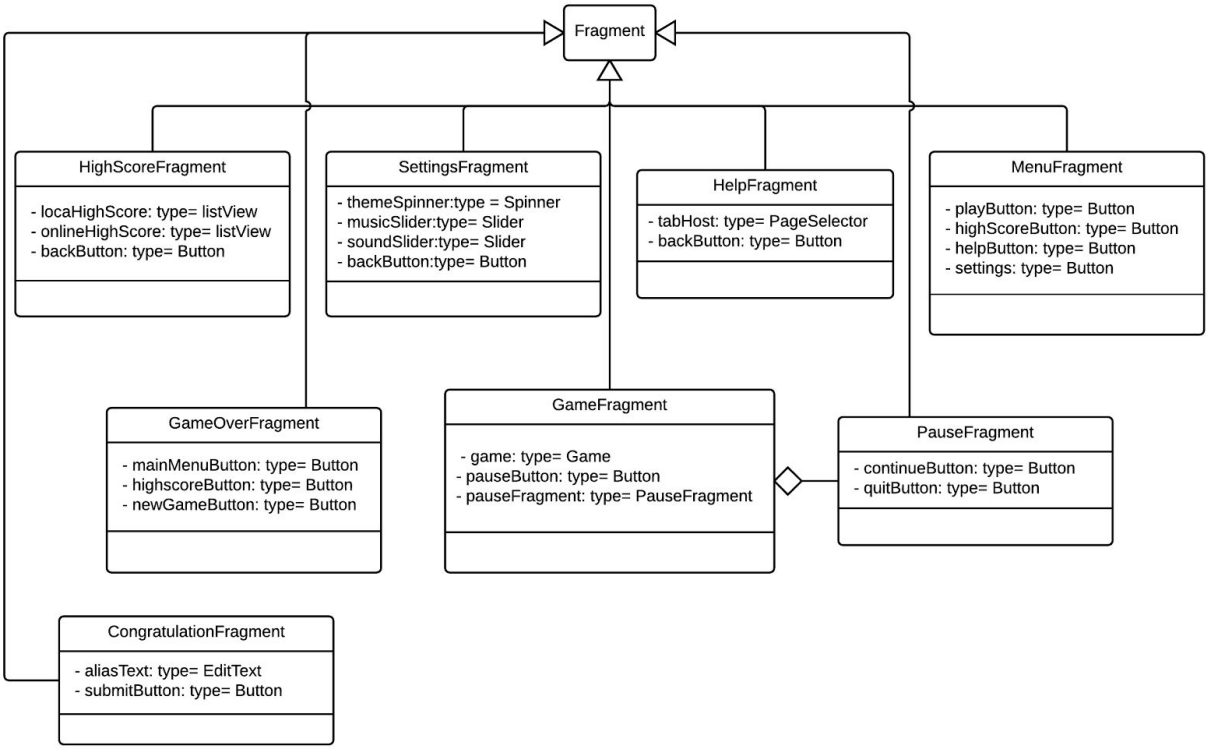

```
public static Sounds GetInstance(Context context) {
    if (instance == null) {
        instance = new Sounds(context);
    }
    return instance;

}
```

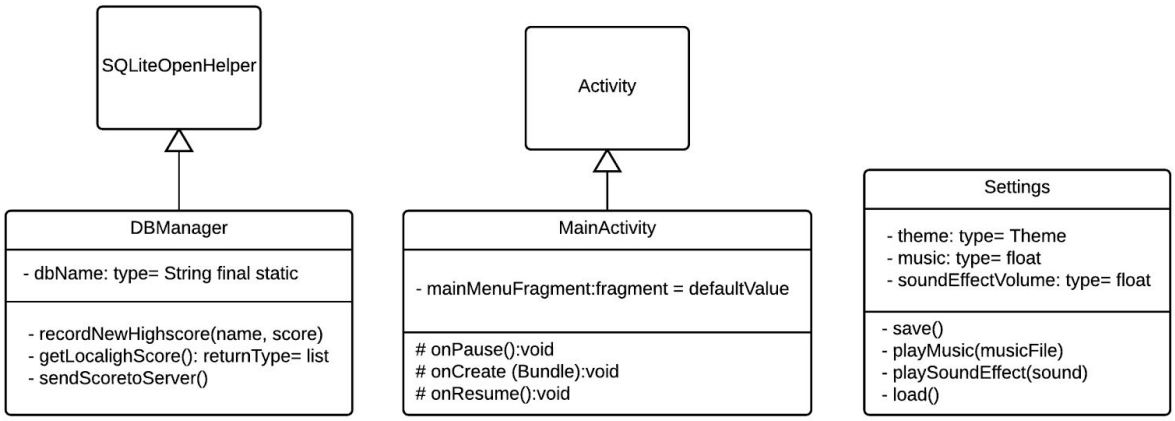
```
public static Settings getInstance(Context context) {
    if (instance == null) {
        instance = new Settings(context);
    }
    return instance;
}
```

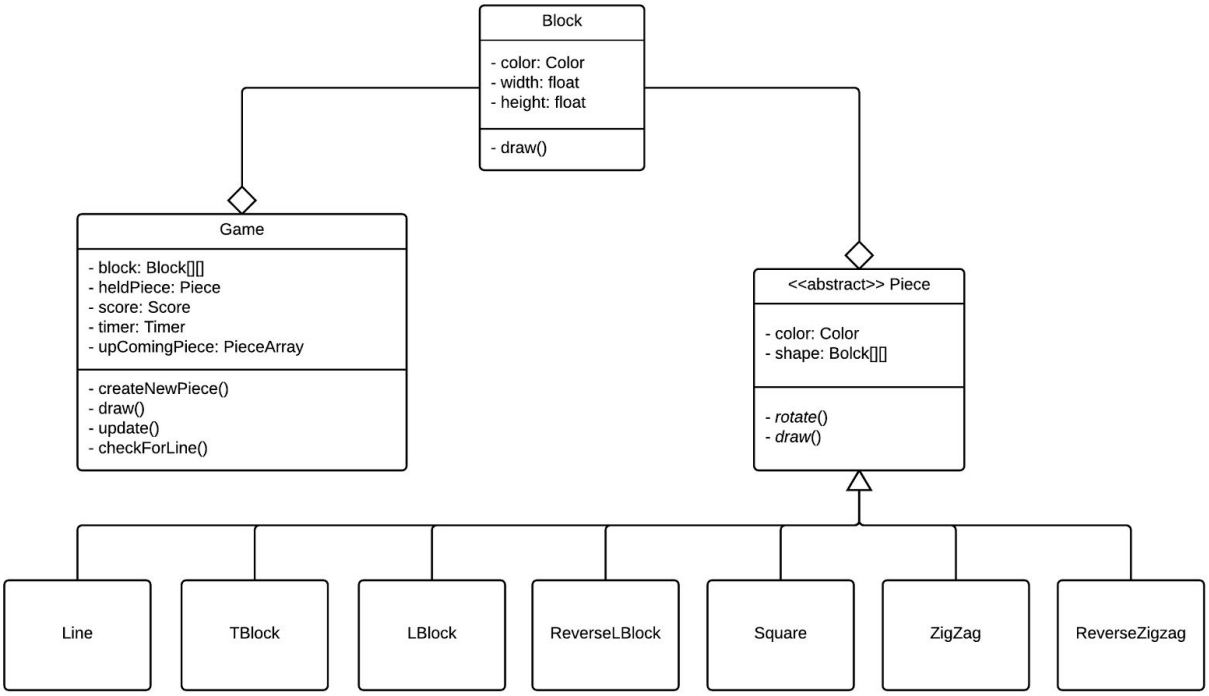
The last pattern we used is singleton. We used it for DatabaseHandler in order to insert new aliases and scores from the gameover fragment, but also to access the same data from the highscore fragment. We used it on Settings in order to access simple saved data from anywhere, such as the app theme for example. Which is used to changed the background picture of the app and needs to be accessed from the BoardView and MainActivity. The Settings Fragment is in chanrge of modifiying the Settings instance variables. It is also used to set the music volume and sound volume which is used by the Sounds class. The Sounds Class was also singleton in order to play music and sounds and was used in GameFragment and the BoardPanel.

3. Improvement

Part 2 Class Diagram:


Fragment
HighScoreFragment
- locaHighScore: type= listView
- onlineHighScore: type= listView
- backButton: type= Button
SettingsFragment
- themeSpinner:type = Spinner
- musicSlider:type= Slider
- soundSlider:type= Slider
- backButton:type= Button
HelpFragment
- tabHost: type= PageSelector
- backButton: type= Button
MenuFragment
- playButton: type= Button
- highScoreButton: type= Button
- helpButton: type= Button
- settings: type= Button
GameOverFragment
- mainMenuButton: type= Button
- highscoreButton: type= Button
- newGameButton: type= Button
GameFragment
- game: type= Game
- pauseButton: type= Button
- pauseFragment: type= PauseFragment
PauseFragment
- continueButton: type= Button
- quitButton: type= Button
CongratulationFragment
- aliasText: type= EditText
- submitButton: type= Button


SQLiteOpenHelper
Activity
DBManager
- dbName: type= String final static
- recordNewHighscore(name, score)
- getLocalighScore(): returnType= list
- sendScoretoServer()
MainActivity
- mainMenuFragment:fragment = defaultValue
onPause():void
onCreate (Bundle):void
onResume():void
Settings
- theme: type= Theme
- music: type= float
- soundEffectVolume: type= float
- save()
- playMusic(musicFile)
- playSoundEffect(sound)
- load()



As we can see our original class diagrams were not very well planned out. There were no actual design patterns within the original class diagrams so alot has changed. The basic fragment setup was mostly correct, however we did not include their listener functions which was bad. Also the Game design completely changed as we added a bunch of new classes including the observer pattern in the BoardViewListener and the factory pattern for the ShapeFactory for shapes in order to generate shapes. Overall part 2 was not too useful in our implementation of the game classes since our design was completley rearranged for it. We were however considering all those Shape classes which are Pieces in Part 2 which helped us out.We also thought out the 2d array of blocks in the GameClass in order to draw the game, but that ended up being used in multiple Panel classes such as BoardPanel, UpcomingPiecePanel, and the super class Panel as the TetrisBox[][]. The fragment setup was mostly useful in deciding how the layouts will be presented although we did remove congrutlations fragment which ended up being used as AlertDialog in GameOverFragment in order to enter the alias. We also seperated the Settings class into two classes one being the Sounds classes were as before we wanted Settings to playSounds and music, but we decided that did not make much since. We made the Settings class and DatabaseHandler class into a Singleton, we were planning to do that in part 2, but did not show it in the class diagram.

4. Lesson Learned

Since we did not due a great job on Part 2 we did fail to learn a few parts of analysing and designing a project. However, when we started developing and implementing the project we ended using the design patterns learned in class when we thought of a design pattern that made sense to use in a specific scenario which was incredibly useful. We personally did not find part 2 useful, we had a hard time envisioning in advance what will be the problems we run to when we haven't started coding and therefore we did not find designing the class diagrams before implementing the code as useful for us. It was useful in the sense that it gave us a rough estimate of what to start implementing. In our style of coding, we feel that not everything can be planned out since new and unexcpected problems can arise and sometimes while coding you might see a class design of the code that makes more sense to be implemented then the one envisioned in the create and design stages. So although we did not really find the planning phase too helpful, one thing that we found really valuable that we learned from the class were design patterns that we could use to solve common programming problems. As before we weren't aware of these patterns we might have choosen a less powerful solution to our implementation of the code, but as we added them in, it all sort of started to click as to why patterns such as Factory, Observer, Singleton, etc.. can be incredibly poweful and useful when used to solve problems. We also ended up using concurrency for our GameThread and for starting a new thread to get online high scores from the internet and this was also something useful that was elobrated in class.