GAME DESIGN AND DEVELOPMENT

Assignment 2 (Game: Brick Breaker)

Designed & Developed By:

Mandeep Kaur (C0684262)

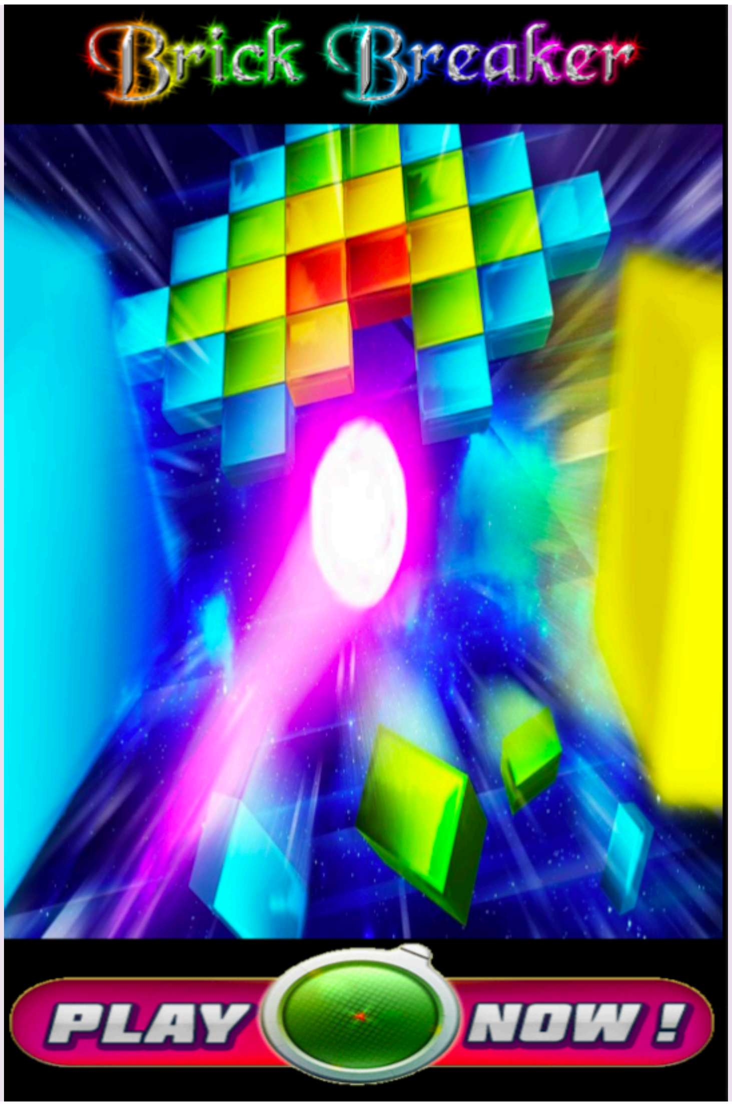
Sahil Thapar (C0688032)

Mohamed Fahad Shaikh (C0686094)

Brick Breaker

Welcome!

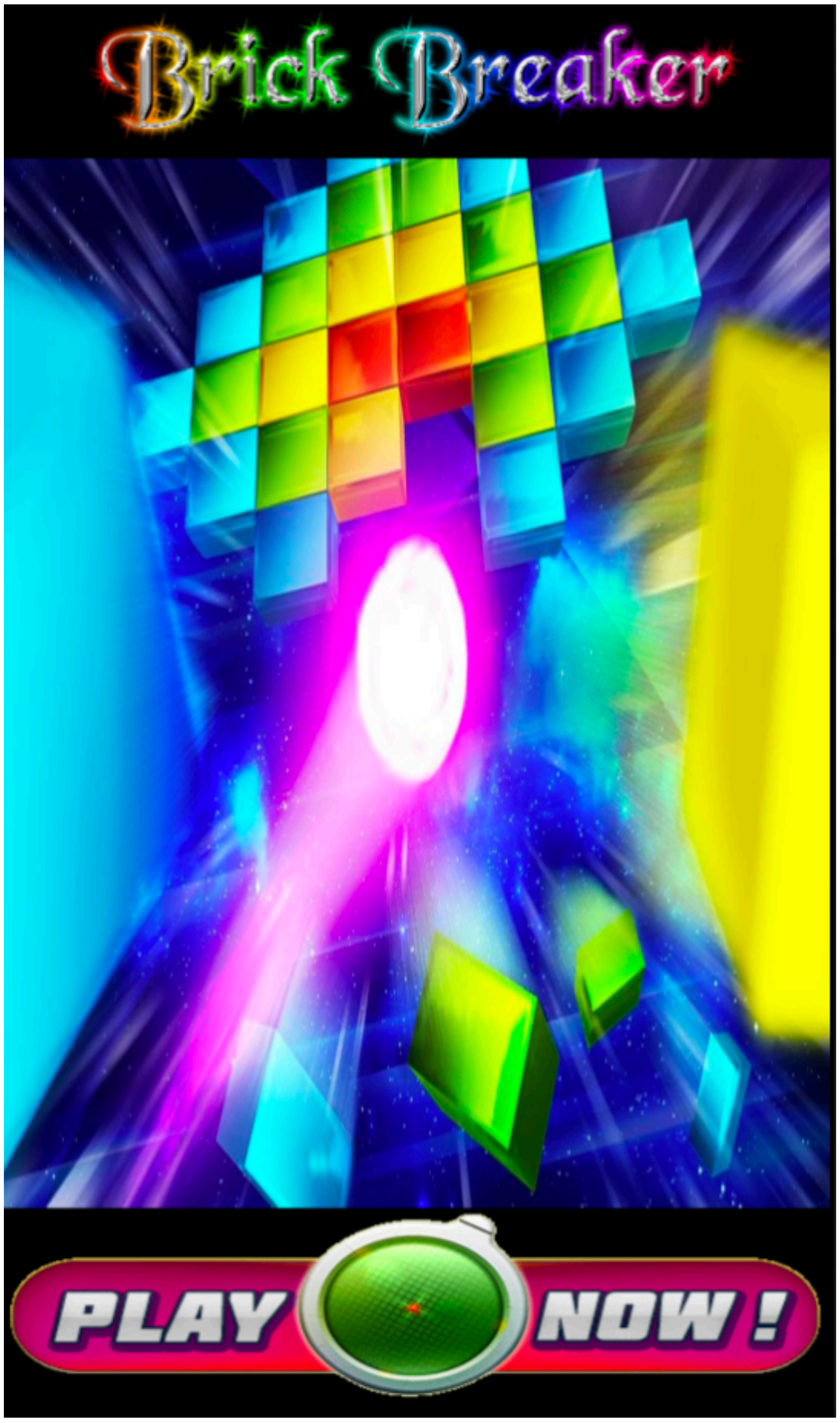
Brick Breaker is a fun and interactive game where you have to maintain ball on disk and make your way out by breaking those bricks.



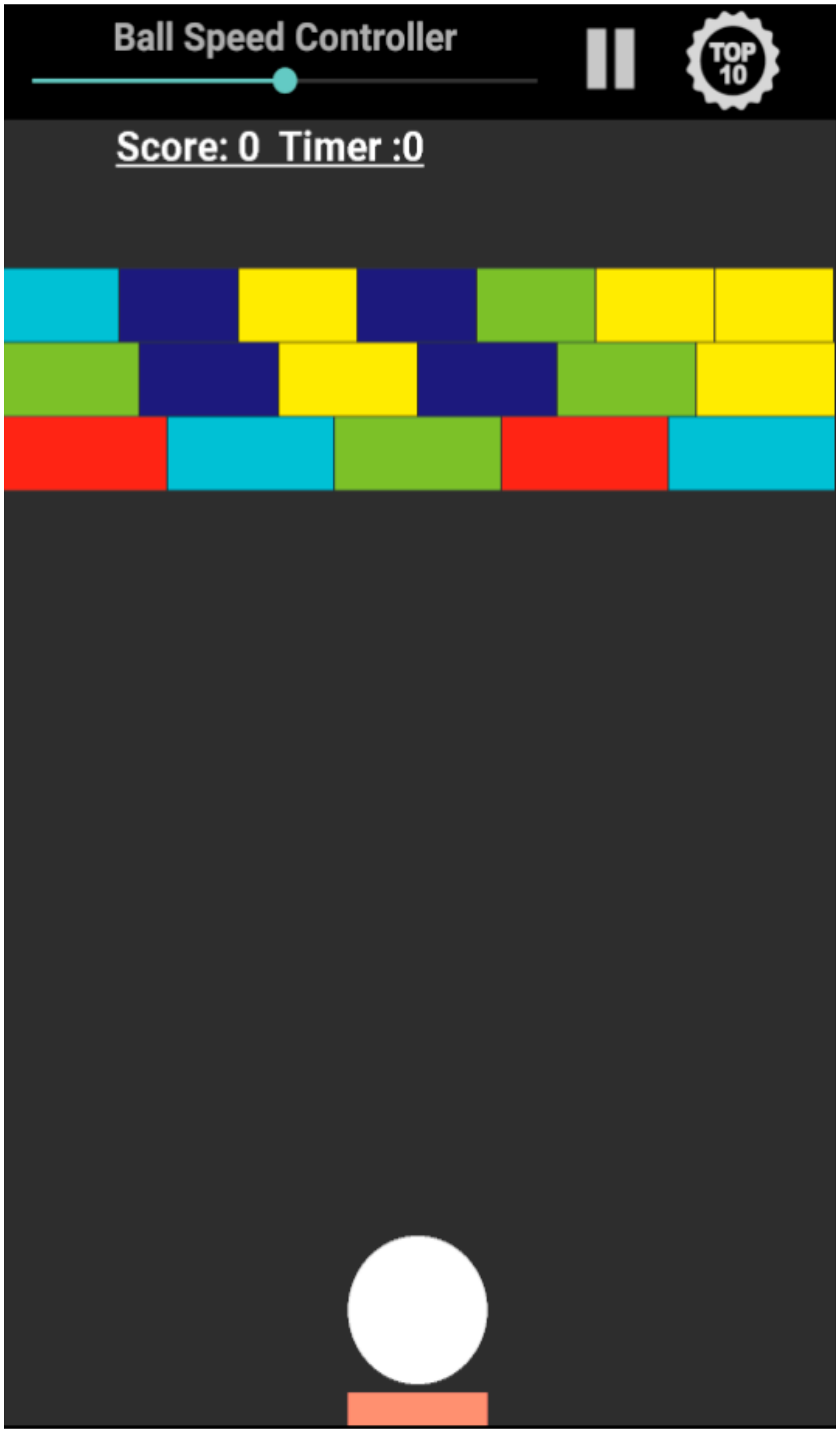
Project Link for code and .apk file:

https://drive.google.com/drive/folders/0B044vnhKJQIgeEFIaFp2SXlnWGM?usp=sharing

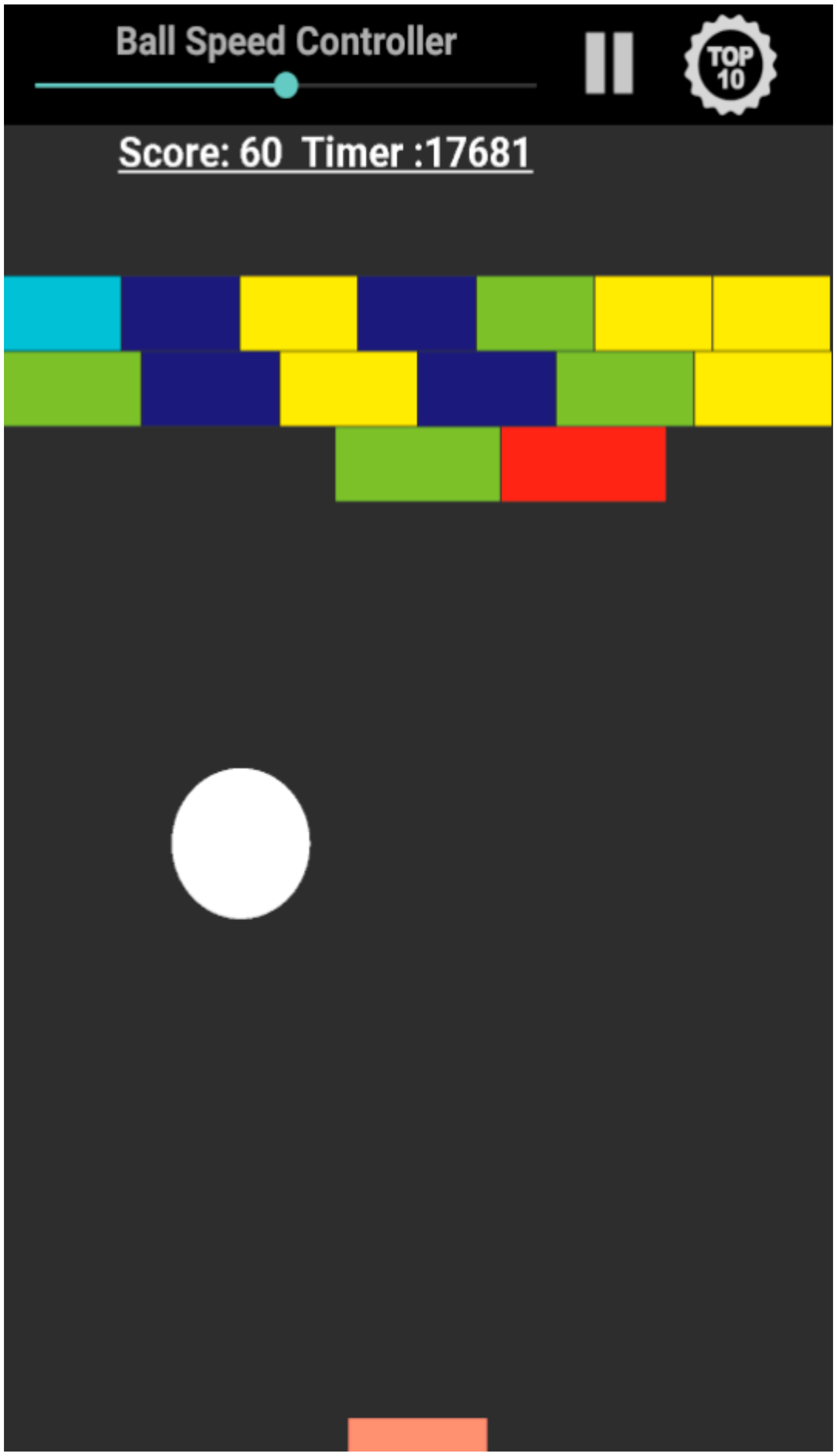
Screenshots of Game Play:



Main Screen:

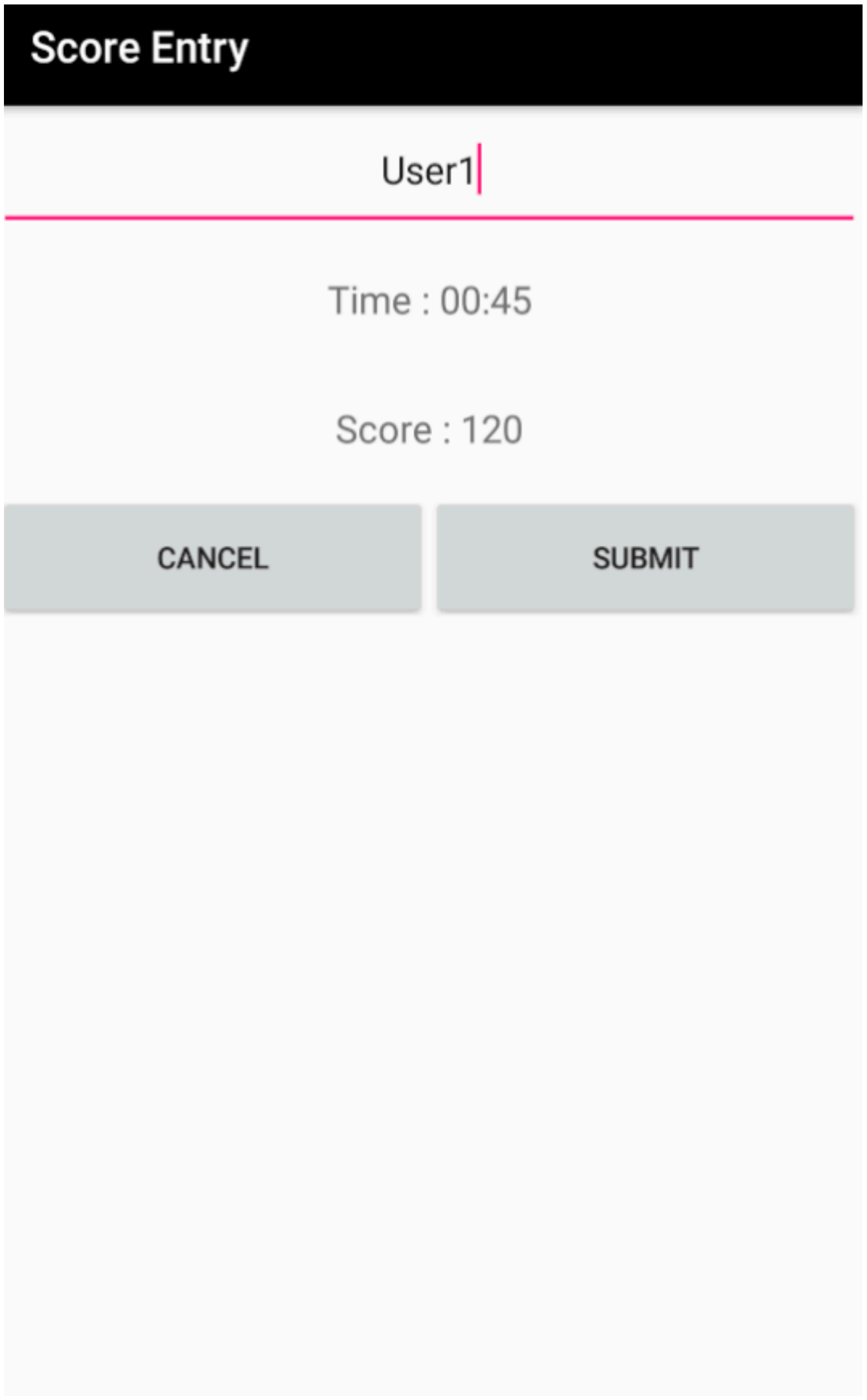


Main Screen:

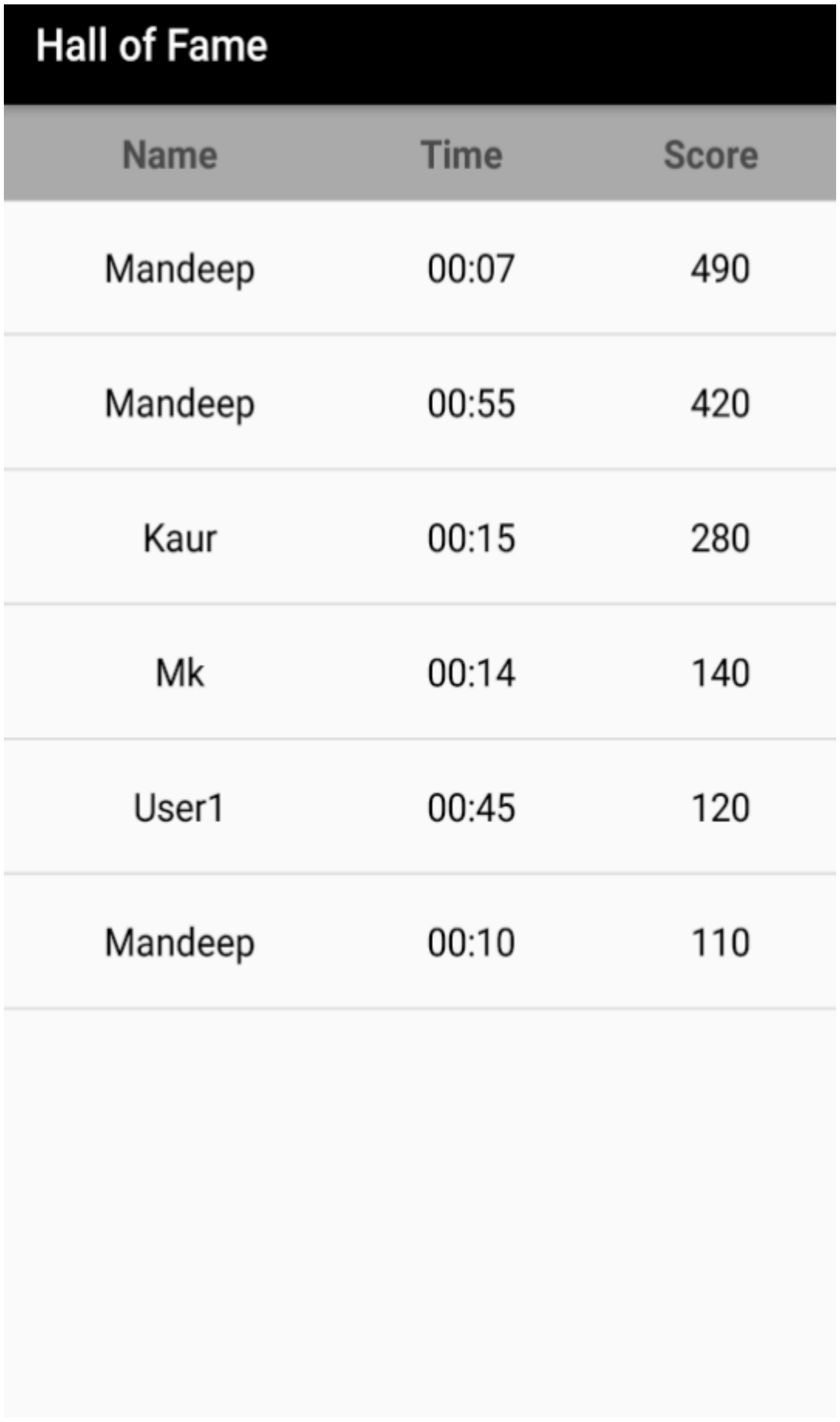


Score: 60 Timer :17681

Score Saving screen:



Top Scorer Screen:



Hall of Fame

Name	Time	Score
Mandeep	00:07	490
Mandeep	00:55	420
Kaur	00:15	280
Mk	00:14	140
User1	00:45	120
Mandeep	00:10	110

Code:

Activities

1. LauncherScreen.java

```java
package com.example.mandeepkaur.breakoutgame.activities;

import android.content.Intent;
import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

import com.example.mandeepkaur.breakoutgame.R;

import pl.droidsonroids.gif.GifTextView;

public class LauncherScreen extends AppCompatActivity {

    GifTextView start;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_launcher_screen);

        final MediaPlayer mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.theme);
        mediaPlayer.start();

        start = (GifTextView)findViewById(R.id.start);
        start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(LauncherScreen.this, BreakoutGame.class);
                mediaPlayer.stop();
                startActivity(intent);
            }
        });
    }
}
```

2. BreakoutGame.java

```java
package com.example.mandeepkaur.breakoutgame.activities;

import android.content.Context;
import android.content.Intent;
import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.RectF;
import android.graphics.Typeface;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.Display;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.widget.Button;



import android.widget.ImageView;
import android.widget.LinearLayout;
import android.view.ViewGroup.LayoutParams;
import android.widget.SeekBar;
import android.widget.Toast;

import com.example.mandeepkaur.breakoutgame.models.Ball;
import com.example.mandeepkaur.breakoutgame.models.Brick;
import com.example.mandeepkaur.breakoutgame.models.ScoreDataModel;
import com.example.mandeepkaur.breakoutgame.utilities.FileHandler;
import com.example.mandeepkaur.breakoutgame.models.Paddle;
import com.example.mandeepkaur.breakoutgame.R;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;

import static java.lang.Math.abs;

public class BreakoutGame extends AppCompatActivity {

    BreakoutView breakoutView;
    LayoutParams params;
    Button playButton, pauseButton;
    ImageView imageView;
    SeekBar ballSpeedSeekbar;
    public static long fps = 50;

    int firstTimeRun=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        breakoutView = new BreakoutView(this);
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);

        LayoutInflater inflater = (LayoutInflater)
```

```
View v = inflater.inflate(R.layout.custom_toolbar, layout, false);
layout.addView(v);
layout.addView(breakoutView);


params = layout.getLayoutParams();
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);
int screenHeight = metrics.heightPixels;
int screenWidth = metrics.widthPixels;

setContentView(layout);

ballSpeedSeekbar=(SeekBar)findViewById(R.id.seekBar);
imageView = (ImageView)findViewById(R.id.imageView);
playButton=(Button)findViewById(R.id.playButton);
pauseButton=(Button)findViewById(R.id.pauseButton);
playButton.setVisibility(View.INVISIBLE);


imageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        breakoutView.pause();
        startActivity(new Intent(getApplicationContext(), HallOfFame.class));
    }
});

playButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        playButton.setVisibility(View.INVISIBLE);
        pauseButton.setVisibility(View.VISIBLE);
        breakoutView.resume();
    }
});

pauseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pauseButton.setVisibility(View.INVISIBLE);
        playButton.setVisibility(View.VISIBLE);
        breakoutView.pause();
    }
});

ballSpeedSeekbar.setProgress(50);
ballSpeedSeekbar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    int progressChanged = 0;

    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        progressChanged = progress;
        fps = (99 - progressChanged);
    }

    public void onStartTrackingTouch(SeekBar seekBar) {
        // TODO Auto-generated method stub
        public void onStopTrackingTouch(SeekBar seekBar) {
            fps = (99 - progressChanged);
            Toast.makeText(getApplicationContext(), "Speed set to:" + progressChanged,
                    Toast.LENGTH_SHORT).show();
        }
    });

}

class BreakoutView extends SurfaceView implements Runnable,SensorEventListener {

    FileHandler fileHandler = new FileHandler(getApplicationContext());
    ScoreDataModel scoreDataModel = new ScoreDataModel();

    Thread gameThread = null;

    SurfaceHolder ourHolder;

    volatile boolean playing;


    boolean paused = true;

    Canvas canvas;
    Paint paint;

    private Object pauseLock;
    private boolean mPause;

    int screenX;
    int screenY;

    private SensorManager senSensorManager;
    private Sensor senAccelerometer;
    private long lastUpdate = -1;
    private float last_x, last_y, last_z;
    private static final int SHAKE_THRESHOLD = 800;

    int maxScore=0;

    Paddle paddle;

    Ball ball;

    long startTime;
    long timer;
    int firstTime=0;

    float initialTouch=0;

    Brick[] bricks = new Brick[200];
    int numBricks = 0;

    SoundPool soundPool;
    int beep1ID = -1;
    int beep2ID = -1;
    int loseLifeID = -1;
    int explodeID = -1;

    int score = 0;

    public BreakoutView(Context context) {

        super(context);

        ourHolder = getHolder();
        paint = new Paint();

        Display display = getWindowManager().getDefaultDisplay();
        Point size = new Point();
        display.getSize(size);

        screenX = size.x;
        screenY = size.y;

        paddle = new Paddle(screenX, screenY);

        ball = new Ball(screenX, screenY);

        pauseLock=new Object();
        mPause=false;

        soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC,0);

        try{
            AssetManager assetManager = context.getAssets();
            AssetFileDescriptor descriptor;

            descriptor = assetManager.openFd("beep1.ogg");
            beep1ID = soundPool.load(descriptor, 0);

            descriptor = assetManager.openFd("beep2.ogg");
            beep2ID = soundPool.load(descriptor, 0);

            descriptor = assetManager.openFd("beep3.ogg");
            beep3ID = soundPool.load(descriptor, 0);

            descriptor = assetManager.openFd("loseLife.ogg");
            loseLifeID = soundPool.load(descriptor, 0);

            descriptor = assetManager.openFd("explode.ogg");
            explodeID = soundPool.load(descriptor, 0);

        }catch(IOException e){
            Log.e("error", "failed to load sound files");
        }


        senSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        senAccelerometer = senSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        senSensorManager.registerListener(this, senAccelerometer ,
SensorManager.SENSOR_DELAY_NORMAL);
```

```
    createBricksAndRestart();

}

public int getScore(){
    return score;
}

public int getTime(){
    return (int)timer;
}

public void createBricksAndRestart(){

     ball.reset(screenX, screenY);
     paddle.reset(screenX,screenY);
     int brickWidth = screenX ;
     int brickHeight = screenX/12 ;

     startTime=0;
     timer=0;
     firstTime=0;
     maxScore=0;

     Random r=new Random();

     numBricks = 0;
     for(int column = 0; column < 8; column ++ ){
         for(int row = 2; row < 5; row ++ ){
             bricks[numBricks] = new Brick(row, column, brickWidth/(9-row), brickHeight);
             if(bricks[numBricks].getRect().left>0 && bricks[numBricks].getRect().right<screenX)
             {
                 int temp=0;

             while(temp==0)
             {
                 temp=r.nextInt(7-1);
             }
             while(numBricks>=3 && bricks[numBricks-3].type==temp)
             {
                 temp=r.nextInt(7-1);
             }
             maxScore+=temp;
             bricks[numBricks].type=temp;
             bricks[numBricks].hits=bricks[numBricks].type;
             numBricks ++;
             }
         }

     }

     score=0;
     paused=true;

 }

@Override
```

```
        while (playing) {
            synchronized (pauseLock) {
                while (mPause) {
                    try {
                        pauseLock.wait();
                        } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }

            if(!paused){

                if(firstTime==0)
                {
                    startTime=System.currentTimeMillis();
                    firstTime=1;
                }
                timer=System.currentTimeMillis()-startTime;
                update();
            }

            draw();

        }

    }

    public void update() {

        paddle.update(fps);

        ball.update(fps);

        for(int i = 0; i < numBricks; i++){

            if (bricks[i].getVisibility()){

                if(intersects(bricks[i].getRect(), ball)) {
                    bricks[i].hits--;
                    if(bricks[i].hits==0) {
                        bricks[i].setInvisible();
                    }
                    ball.reverseYVelocity();
                    score = score + 10;
                    soundPool.play(explodeID, 1, 1, 0, 0, 1);
                }
            }
        }

        if(intersects(paddle.getRect(), ball)) {
            ball.setRandomXVelocity();
            ball.reverseYVelocity();
            ball.clearObstacleY(paddle.getRect().top - ball.getR() - 2);
            soundPool.play(beep1ID, 1, 1, 0, 0, 1);
        }
```

```
                ball.reverseYVelocity();
                ball.clearObstacleY(screenY - ball.getR() - 2);
                soundPool.play(loseLifeID, 1, 1, 0, 0, 1);
                scoreDataModel.setScore(Integer.toString(getScore()));
                scoreDataModel.setTime(Integer.toString(getTime()));
                if(fileHandler.isInTopTen(scoreDataModel) == 1) {
                    ArrayList<Integer> arrayList = new ArrayList<>();
                    arrayList.add(getScore());
                    arrayList.add(getTime());
                    createBricksAndRestart();
                    startActivity(new Intent(getApplicationContext(), ScoreEntry.class).putIntegerArrayListExtra("caller",
arrayList));
                }else {
                    createBricksAndRestart();
                }

            }
            if(ball.getY()-ball.getR() < 0){
                ball.reverseYVelocity();
                ball.clearObstacleY(ball.getR()+12);
                soundPool.play(beep2ID, 1, 1, 0, 0, 1);
            }

            if(ball.getX()-ball.getR() < 0){
                ball.reverseXVelocity();
                ball.clearObstacleX(ball.getR() + 2);
                soundPool.play(beep3ID, 1, 1, 0, 0, 1);
            }

            if(ball.getX()+ball.getR() > screenX - 10){
                ball.reverseXVelocity();
                ball.clearObstacleX(screenX - ball.getR() - 22);
                soundPool.play(beep3ID, 1, 1, 0, 0, 1);
            }

            if(score==maxScore*10){
                paused = true;

                ArrayList<Integer> arrayList = new ArrayList<>();
                arrayList.add(getScore());
                arrayList.add(getTime());
                createBricksAndRestart();

                startActivity(new Intent(getApplicationContext(), ScoreEntry.class).putIntegerArrayListExtra("caller",
arrayList));

            }

        }

        public void draw() {

            if (ourHolder.getSurface().isValid()) {
                canvas = ourHolder.lockCanvas();

                canvas.drawColor(Color.rgb( 50, 50, 50));
        canvas.drawRect(paddle.getRect(), paint);


        paint.setColor(Color.argb(255, 255, 255, 255));

        canvas.drawCircle(ball.centerX, ball.centerY, ball.radius, paint);


        paint.setColor(Color.argb(255,  249, 129, 0));
        paint.setUnderlineText(true);
        paint.setTypeface(Typeface.create(Typeface.DEFAULT, Typeface.BOLD));

        for(int i = 0; i < numBricks; i++){
            if(bricks[i].getVisibility()) {
                if(bricks[i].type==5)
                    paint.setColor(Color.argb(255,255, 235, 59));
                else if(bricks[i].type==4)
                    paint.setColor(Color.argb(255, 244, 67, 54));
                else if(bricks[i].type==3)
                    paint.setColor(Color.argb(255, 139, 195, 74));
                else if(bricks[i].type==2)
                    paint.setColor(Color.argb(255, 26, 35, 126));
                else
                    paint.setColor(Color.argb(255,000,188,212));
                canvas.drawRect(bricks[i].getRect(), paint);
            }
        }

        paint.setColor(Color.argb(255,  255, 255, 255));

        paint.setTextSize(50);
        canvas.drawText("Score: " + score +
                "  Timer :"+timer, 150,50, paint);

        ourHolder.unlockCanvasAndPost(canvas);
    }

}


public boolean intersects(RectF rect,Ball ball){

    if((abs(ball.getX() -rect.left)< ball.getR() || abs(ball.getX() -rect.right)< ball.getR() ))
    {
        if (abs(ball.getY()-rect.top)<=ball.getR())
            return true;
        else if(abs(ball.getY()-rect.bottom)<=ball.getR())
            return true;
        else if((ball.getX()-rect.top)<=0 && (ball.getX()-rect.bottom)>=0
            return true;
        else return false;
    }


    return false;
}
```

```
    public void pause() {
        senSensorManager.unregisterListener(this);
        mPause=true;

    }


    public void resume() {
        senSensorManager.registerListener(this, senAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
        playing = true;
        if(firstTimeRun==0)
        {
            firstTimeRun=1;
            gameThread = new Thread(this);
            gameThread.start();
        }
        else{
            synchronized (pauseLock)
            {
                mPause=false;

                pauseLock.notifyAll();
            }
        }

    }

    @Override
    public boolean onTouchEvent(MotionEvent motionEvent) {

        switch (motionEvent.getAction() & MotionEvent.ACTION_MASK) {

            case MotionEvent.ACTION_DOWN:
                paused = false;
                initialTouch=motionEvent.getX();
                break;

            case MotionEvent.ACTION_MOVE:
                float x=motionEvent.getX();
                if(x-initialTouch>0)
                    paddle.setMovementState(paddle.RIGHT);
                else if(x-initialTouch<0)
                    paddle.setMovementState(paddle.LEFT);
                else
                {
                    if(x-paddle.getRect().left<=0)
                        paddle.setMovementState(paddle.LEFT);
                    if(x-paddle.getRect().right>=0)
                        paddle.setMovementState(paddle.RIGHT);
                }
                break;

            case MotionEvent.ACTION_UP:

                paddle.setMovementState(paddle.STOPPED);
            }
            return true;
        }

        public  float Round(float Rval, int Rpl) {
            float p = (float)Math.pow(10,Rpl);
            Rval = Rval * p;
            float tmp = Math.round(Rval);
            return (float)tmp/p;
        }

        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                long curTime = System.currentTimeMillis();
                if ((curTime - lastUpdate) > 100) {
                    long diffTime = (curTime - lastUpdate);
                    lastUpdate = curTime;


                    float x = event.values[SensorManager.DATA_X];
                    float y = event.values[SensorManager.DATA_Y];
                    float z = event.values[SensorManager.DATA_Z];

                    float prevVelocity=ball.xVelocity;

                    if(Round(x,4)>5.0000){
                        ball.xVelocity-=150;
                    }
                    else if(Round(x,4)<-5.0000){
                        ball.xVelocity+=150;
                    }
                    else
                    {
                        ball.xVelocity=prevVelocity;
                    }

                    last_x = x;
                    last_y = y;
                    last_z = z;
                }
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {

        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        breakoutView.resume();
    }

    @Override
```

```
        super.onPause();

        breakoutView.pause();
    }

}
```

3. ScoreEntry.java

```
package com.example.mandeepkaur.breakoutgame.activities;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.mandeepkaur.breakoutgame.models.ScoreDataModel;
import com.example.mandeepkaur.breakoutgame.utilities.FileHandler;
import com.example.mandeepkaur.breakoutgame.R;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.concurrent.TimeUnit;

public class ScoreEntry extends AppCompatActivity {

    TextView entryTime, entryScore;
    EditText entryName;
    Button submitButton, cancelButton;
    ScoreDataModel receivedScoreDataModel;
    FileHandler fileHandler;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_score_entry);

        entryName = (EditText)findViewById(R.id.entryName);
        entryScore = (TextView)findViewById(R.id.entryScore);
        entryTime = (TextView)findViewById(R.id.entryTime);

        submitButton = (Button)findViewById(R.id.buttonSubmit);
        cancelButton = (Button)findViewById(R.id.buttonCancel);


        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        toolbar.setTitle("Score Entry");
        toolbar.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
        toolbar.setTitleTextColor(Color.WHITE);
```

```
receivedScoreDataModel = new ScoreDataModel();
Bundle bundle = getIntent().getExtras();
ArrayList<Integer> arrayList = bundle.getIntegerArrayList("caller");

if(arrayList.size() != 0) {
    receivedScoreDataModel.setScore(arrayList.get(0).toString());
    receivedScoreDataModel.setTime(arrayList.get(1).toString());
    long milliseconds = Long.valueOf(receivedScoreDataModel.getTime());

    String time = String.format("%02d:%02d",
            TimeUnit.MILLISECONDS.toMinutes(milliseconds),
            TimeUnit.MILLISECONDS.toSeconds(milliseconds) -
                    TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(milliseconds))
    );
    entryTime.setText("Time : "+time);
    entryScore.setText("Score : "+receivedScoreDataModel.getScore().toString());
}
cancelButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});

submitButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if(entryName.getText().toString().isEmpty()){
            Toast.makeText(getApplicationContext(), "Please enter your name.",
                    Toast.LENGTH_SHORT).show();
            return;
        }
        fileHandler = new FileHandler(getApplicationContext());

        ArrayList<ScoreDataModel> arrayList = new ArrayList<>();
        arrayList = fileHandler.getDataObject();

        receivedScoreDataModel.setId(fileHandler.getMaxId() + 1);
        receivedScoreDataModel.setName(entryName.getText().toString());

        arrayList.add(receivedScoreDataModel);

        Collections.sort(arrayList, new CustomComparator());

        if(arrayList.size() > 10){
            for(int i=10;i<arrayList.size()-1;i++){
                arrayList.remove(i);
            }
        }

        fileHandler.setDataObject(arrayList);

        finish();
        startActivity(new Intent(getApplicationContext(), HallOfFame.class));
    }
```

```
        }


        public class CustomComparator implements Comparator<ScoreDataModel> {

            @Override
            public int compare(ScoreDataModel lhs, ScoreDataModel rhs) {
                if (Integer.parseInt(lhs.getScore()) > Integer.parseInt(rhs.getScore())){
                    return -1;
                }else if (Integer.parseInt(lhs.getScore()) < Integer.parseInt(rhs.getScore())){
                    return 1;
                }else {
                    return 0;
                }
            }
        }

    }
```

4. HallOfFame.java

```
    package com.example.mandeepkaur.breakoutgame.activities;

    import android.graphics.Color;
    import android.os.Bundle;
    import android.support.v7.app.AppCompatActivity;
    import android.support.v7.widget.Toolbar;
    import android.view.View;
    import android.widget.ListView;

    import com.example.mandeepkaur.breakoutgame.models.ScoreDataModel;
    import com.example.mandeepkaur.breakoutgame.utilities.FileHandler;
    import com.example.mandeepkaur.breakoutgame.R;
    import com.example.mandeepkaur.breakoutgame.adapters.ScoreAdapter;

    import java.util.ArrayList;

    public class HallOfFame extends AppCompatActivity {

        FileHandler fileHandler = new FileHandler(this);
        ArrayList<ScoreDataModel> scoreDataModelArrayList = new ArrayList<>();
        ScoreAdapter scoreAdapter;
        ListView scoreList;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_hall_of_fame);

            Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

            toolbar.setTitle("Hall of Fame");
            toolbar.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            toolbar.setTitleTextColor(Color.WHITE);
        scoreList = (ListView)findViewById(R.id.scoreList);

        scoreDataModelArrayList = fileHandler.getDataObject();

        scoreAdapter = new ScoreAdapter(getApplicationContext(), scoreDataModelArrayList);

        scoreList.setAdapter(scoreAdapter);

    }

    @Override
    public void onResume() {
        super.onResume();

    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
    }
}
```

Adapters:

1. ScoreAdapter.java

```
package com.example.mandeepkaur.breakoutgame.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.mandeepkaur.breakoutgame.R;
import com.example.mandeepkaur.breakoutgame.models.ScoreDataModel;

import java.util.ArrayList;
import java.util.concurrent.TimeUnit;

public class ScoreAdapter extends ArrayAdapter {
    private final Context context;
    private final ArrayList<ScoreDataModel> scoreDataModelArrayList;
    TextView name, timeTaken, score;

    public ScoreAdapter(Context context, ArrayList<ScoreDataModel> scoreDataModelArrayList) {
        super(context, R.layout.hall_of_fame_elements, scoreDataModelArrayList);
        this.context = context;
        this.scoreDataModelArrayList = scoreDataModelArrayList;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.hall_of_fame_elements, parent, false);
        }

        name = (TextView)convertView.findViewById(R.id.name);
        name.setText(scoreDataModelArrayList.get(position).getName());

        timeTaken = (TextView)convertView.findViewById(R.id.timeTaken);
        long milliseconds = Long.valueOf(scoreDataModelArrayList.get(position).getTime());

        String time = String.format("%02d:%02d",
                TimeUnit.MILLISECONDS.toMinutes(milliseconds),
                TimeUnit.MILLISECONDS.toSeconds(milliseconds) -
                        TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(milliseconds))
        );
        timeTaken.setText(time);

        score = (TextView)convertView.findViewById(R.id.score);
        score.setText(scoreDataModelArrayList.get(position).getScore());
        return convertView;
    }
}
```

Models:

1. Ball.java

```
package com.example.mandeepkaur.breakoutgame.models;

import java.util.Random;

public class Ball {
    public float xVelocity;
    public float yVelocity;
    public float centerX;
    public float centerY;
    public float radius;

    public Ball(int screenX, int screenY){

        xVelocity = 100;
        yVelocity = -200;

        radius=screenX/12;
    }

    public void update(long fps){
        centerX=centerX+(xVelocity/fps);
        centerY=centerY+(yVelocity/fps);
    }

    public float getX(){
        return centerX;
    }
    public float getY(){
        return centerY;
    }

    public float getR(){
        return radius;
    }

    public void reverseYVelocity(){
        yVelocity = -yVelocity;
    }

    public void reverseXVelocity(){
        xVelocity = - xVelocity;
    }

    public void setRandomXVelocity(){
        Random generator = new Random();
        int answer = 7-generator.nextInt(8);

        if(answer == 0){
            reverseXVelocity();
        }
    }

    public void clearObstacleY(float y){
        centerY = y;
    }

    public void clearObstacleX(float x){
        centerX=x;
    }

    public void reset(int x, int y){
        centerX = x / 2;
        centerY= y - 350;
    }

}
```

2. Brick.java

```
package com.example.mandeepkaur.breakoutgame.models;

import android.graphics.RectF;

public class Brick {

    private RectF rect;

    private boolean isVisible;

    public int type;

    public int hits;

    public Brick(int row, int column, int width, int height){
        isVisible = true;

        int padding = 1;

        rect = new RectF(column * width + padding,
                row * height + padding,
                column * width + width - padding,
                row * height + height - padding);
    }

    public RectF getRect(){
        return this.rect;
    }

    public void setInvisible(){
        isVisible = false;
    }

    public boolean getVisibility(){
        return isVisible;
    }
}
```

3. Paddle.java

```
package com.example.mandeepkaur.breakoutgame.models;

import android.graphics.RectF;

public class Paddle {

    private RectF rect;
    private float length;
    private float height;
    private int screenWidth;
    private int screenHeight;
    private float x;
    private float y;
    private float paddleSpeed;
    public final int STOPPED = 0;
    public final int LEFT = 1;
    public final int RIGHT = 2;
    private int paddleMoving = STOPPED;

    public Paddle(int screenX, int screenY){
        length = screenX/6;
        height = screenY/6;
        screenWidth=screenX;
        screenHeight=screenY;

        x = screenX / 2-length/2;
        y =  screenY-150;

        rect = new RectF(x, y, x + length, y+height );

        paddleSpeed = 350
    }


    public RectF getRect(){
        return rect;
    }


    public void reset(float screenX,float screenY){
        rect.left = screenX / 2-length/2;
        rect.top =  screenY-250;
        rect.right=rect.left+length;
        rect.bottom=rect.top+length;
    }

    public void setMovementState(int state){
        paddleMoving = state;
    }

    public void update(long fps){
        if(x - paddleSpeed / fps>=10 && paddleMoving == LEFT){
            x = x - paddleSpeed / fps;
        }

        if(x + paddleSpeed / fps+length<= screenWidth-10 &&  paddleMoving == RIGHT){
            x = x + paddleSpeed / fps;
        }

        rect.left = x;
        rect.right = x + length;
    }

}
```

4. ScoreDataModel.java

```
package com.example.mandeepkaur.breakoutgame.models;

import java.util.Comparator;

public class ScoreDataModel implements Comparable {

    private String name;
    private String score;
    private String time;
    private String id;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
    }

    public String getScore() {
        return score;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setScore(String score) {
        this.score = score;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getTime() {
        return time;
    }

    public static Comparator<ScoreDataModel> firstNameComparator = new Comparator<ScoreDataModel>() {

        public int compare(ScoreDataModel c1, ScoreDataModel c2) {

            int value1=c1.getScore().compareTo(c2.getScore());

            if(value1==0){
                return c1.getTime().compareTo(c2.getTime());
            }
            return value1;
        }};

    @Override
    public int compareTo(Object another) {
        return 0;
    }


}
```

Utilities:

1. FileHandler.java

```
package com.example.mandeepkaur.breakoutgame.utilities;

import android.content.Context;
import android.os.Environment;

import com.example.mandeepkaur.breakoutgame.models.ScoreDataModel;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class FileHandler {

    Context fileContext;

    public FileHandler(Context fileContext){
        this.fileContext=fileContext;
    }

    public void setDataObject(ArrayList<ScoreDataModel> inf){
        clearContents();
        try {

            final File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/Download/" );
            File foo=new File(dir,"temp11.txt");

            PrintStream pr = new PrintStream(fileContext.openFileOutput(foo.getName(),
Context.MODE_PRIVATE));

            for(ScoreDataModel temp:inf){
                String str;

                str=temp.getName();
                str="(name:"+str+")";
                pr.print(str);

                str=temp.getScore();
                str="(score:"+str+")";
                pr.print(str);

                str=temp.getTime();
                str="(time:"+str+")";
                pr.print(str);

                str=temp.getId();
                str="(id:"+str+")";
                pr.print(str);

                pr.println("");
            }
        } catch(Exception ex) {
            ex.printStackTrace();
        }
    }

    public ArrayList<ScoreDataModel> getDataObject(){
        String str;

        final File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() +"/Download" );
        File foo=new File(dir,"temp11.txt");

        if(!foo.exists()) {
            try {
                foo.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        try{
            FileInputStream fis=fileContext.openFileInput(foo.getName());
            BufferedReader br=new BufferedReader(new InputStreamReader(fis));
            while((str=br.readLine())!=null)
            {
                inf.add(convertStringtoObject(str));
            }
        }catch(IOException e){
            e.printStackTrace();
        }

        Collections.sort(inf);
        return inf;
    }

    private ScoreDataModel convertStringtoObject(String s){
        HashMap<String,String> temp=convertStringtoMap(s);
        ScoreDataModel inf=new ScoreDataModel();
        for(Map.Entry<String, String> e : temp.entrySet()){


            if(e.getKey().equals("name")){
                inf.setName(e.getValue());
            }

            if(e.getKey().equals("score")){
                inf.setScore(e.getValue());
            }

            if(e.getKey().equals("time")){
                inf.setTime(e.getValue());
            }

            if(e.getKey().equals("id")){
                inf.setId(e.getValue());
            }

        }
        return inf;

    }

    private HashMap<String,String> convertStringtoMap(String s){
        HashMap<String,String> temp=new HashMap<String,String>();
        Matcher m = Pattern.compile("\\((.*?)\\)").matcher(s);
```

```java
            String [] str=m.group(1).split(":",2);
            temp.put(str[0],str[1]);
        }
        return temp;
    }

    private void clearContents(){
        final File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/Download/" );
        File foo=new File(dir,"temp11.txt");
        try{
            FileOutputStream writer = fileContext.openFileOutput(foo.getName(),Context.MODE_PRIVATE);

            writer.write((" ").getBytes());

            writer.close();
        }catch(FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public  String getMaxId() {
        ScoreDataModel temp;
        String str;
        int maxid=0;
        final File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath() +"/Download" );
        File foo=new File(dir,"temp11.txt");
        try{
            FileInputStream fis=fileContext.openFileInput(foo.getName());
            BufferedReader br=new BufferedReader(new InputStreamReader(fis));
            while((str=br.readLine())!=null)
            {
                temp=convertStringtoObject(str);
                if(Integer.parseInt(temp.getId())>maxid){
                    maxid=Integer.parseInt(temp.getId());
                }
            }
        }catch(IOException e){
            e.printStackTrace();
        }

        return String.valueOf(maxid);
    }

    public int isInTopTen(ScoreDataModel obj){
        ArrayList<ScoreDataModel> arr,finalArr;
        arr=getDataObject();
        if(arr.size() < 10){
            return 1;
        }
        int compareScore,compareTime;
        for(ScoreDataModel temp : arr){
            compareScore=obj.getScore().compareTo(temp.getScore());
            compareTime=obj.getTime().compareTo(temp.getTime());
            if(compareScore==1||(compareScore==0 && compareTime==1)){
                return 1;
        }
        return 0;
    }

}
```