Код. Класс MainActivity (отвечающий за стартовый экран приложения).

```
package logic.gamee.codebull;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    /** Первоначальный метод для создания активности. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);          // запуск метода
родительского класса в дополнении с написанным кодом.
        setContentView(R.layout.activity_main);   //  для текущего
объекта activity устанавливается разметка из файла activity_main.xml.
    }

 /** Переход от класса MainActivity(стартовый экран) к классу
SecondActivity(экран с уровнем игры) с помощью нажатия на кнопку "начать
игру" */
    public void startSecondActivity(View v) {
        Intent intent = new Intent(this, SecondActivity.class);
        startActivity(intent);
    }

  /** Переход от класса MainActivity(стартовый экран) к классу
Rulesctivity (экран с правилами игры) с помощью нажатия на кнопку
"Правила игры" */
    public void startRulesActivity(View v) {
        Intent intent2 = new Intent(this, RulesActivity.class);
        startActivity(intent2);
    }
}
```

Код. Класс RulesActivity (отвечающий за экран с правилами).

```
package logic.gamee.codebull;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;

public class RulesActivity extends AppCompatActivity {

    /** Первоначальный метод для создания активности. */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);          // запуск метода
родительского класса в дополнении с написанным кодом.
        setContentView(R.layout.activity_rules);   //  для текущего
объекта activity устанавливается разметка из файла activty_rules.xml.
```

```
    }

/** Переход от класса Rulesctivity(экран с правилами игры) к классу
MainActivity(стартовый экран) с помощью нажатия на кнопку "назад" */
    public void startMainActivity(View v) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
}
```

Код. Класс SecondActivity (отвечающий за экран с уровнем игры).

```
package logic.gamee.codebull;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.Random;

public class SecondActivity extends AppCompatActivity {
    int attemptCounter = 0;   // счетчик для ограничения попыток проверки
введенного числа.
    int digitCounter = 0;    //  счетчик для ограничения цифр в введенном
числе.
    int nullCounter = 0;    //   счетчик для ограничения на ноль в
введённом числе.
    int num = randomNum(); //    переменная, которой присваивается
случайное число для угадывания, генерируемое в методе randomNum.

    /** Создание приватных переменных для кнопок.  */
    private TextView numberView;
    private Button buttonOne;
    private Button buttonTwo;
    private Button buttonThree;
    private Button buttonFour;
    private Button buttonFive;
    private Button buttonSix;
    private Button buttonSeven;
    private Button buttonEight;
    private Button buttonNine;
    private Button buttonZero;
    private Button buttonCancel;
    private Button buttonCheck;
    private TextView resultsView;

    /** Первоначальный метод для создания активности. */
```

```
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);          // запуск метода
родительского класса в дополнении с написанным кодом.
        setContentView(R.layout.second_activity);   //  для текущего
объекта activity устанавливается разметка из файла second_activity.xml.

        /** задавание значений переменных для кнопок с помощью поиска по
id. */
        numberView = findViewById(R.id.numberView);
        buttonOne = findViewById(R.id.buttonOne);
        buttonTwo = findViewById(R.id.buttonTwo);
        buttonThree = findViewById(R.id.buttonThree);
        buttonFour = findViewById(R.id.buttonFour);
        buttonFive = findViewById(R.id.buttonFive);
        buttonSix = findViewById(R.id.buttonSix);
        buttonSeven = findViewById(R.id.buttonSeven);
        buttonEight = findViewById(R.id.buttonEight);
        buttonNine = findViewById(R.id.buttonNine);
        buttonZero = findViewById(R.id.buttonZero);
        buttonCancel = findViewById(R.id.buttonCancel);
        buttonCheck = findViewById(R.id.buttonCheck);
        resultsView = findViewById(R.id.resultsView);
    }

    /**
     Переход от класса SecondActivity(экран с уровнем игры)
     к классу MainActivity(стартовый экран) с помощью нажатия на кнопку
"назад.
     */
    public void startMainActivity(View v) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    /** Действия, при нажатии на кнопку "1". */
    public void onClickOne(View v) {
          buttonOne.setEnabled(false); //блокирование возможности нажатия
на   кнопку
          numberView.setText(numberView.getText() + "1"); //отображение
числа на экране
          digitCounterCheck();
          digitZeroCheck();
    }

    /** Действия, при нажатии на кнопку "2". */
    public void onClickTwo(View v) {
        buttonTwo.setEnabled(false);
        numberView.setText(numberView.getText() + "2");
        digitCounterCheck();
        digitZeroCheck();
    }
```

```
/** Действия, при нажатии на кнопку "3". */
public void onClickThree(View v) {
    buttonThree.setEnabled(false);
    numberView.setText(numberView.getText() + "3");
    digitCounterCheck();
    digitZeroCheck();
}

/** Действия, при нажатии на кнопку "4". */
public void onClickFour(View v) {
    buttonFour.setEnabled(false);
    numberView.setText(numberView.getText() + "4");
    digitCounterCheck();
    digitZeroCheck();
}

/** Действия, при нажатии на кнопку "5". */
public void onClickFive(View v) {
    buttonFive.setEnabled(false);
    numberView.setText(numberView.getText() + "5");
    digitCounterCheck();
    digitZeroCheck();
}

/** Отображение цифры "6" в поле для ввода при нажатии кнопки "6". */
public void onClickSix(View v) {
    buttonSix.setEnabled(false);
    numberView.setText(numberView.getText() + "6");
    digitCounterCheck();
    digitZeroCheck();
}

/** Действия, при нажатии на кнопку "7". */
public void onClickSeven(View v) {
    buttonSeven.setEnabled(false);
    numberView.setText(numberView.getText() + "7");
    digitCounterCheck();
    digitZeroCheck();
}

/** Действия, при нажатии на кнопку "8". */
public void onClickEight(View v) {
    buttonEight.setEnabled(false);
    numberView.setText(numberView.getText() + "8");
    digitCounterCheck();
    digitZeroCheck();
}

/** Действия, при нажатии на кнопку "9". */
public void onClickNine(View v) {
    buttonNine.setEnabled(false);
    numberView.setText(numberView.getText() + "9");
    digitCounterCheck();
    digitZeroCheck();
```

```
    }

    /** Действия, при нажатии на кнопку "0". */
    public void onClickZero(View v) {
        buttonZero.setEnabled(false);
        numberView.setText(numberView.getText() + "0");
        nullCounter++;
        digitCounterCheck();
    }

    /** Действия, при нажатии на кнопку "сброс". */
    public void onClickCancel(View v) {
        clearNum();
        digitCounter = 0;
        nullCounter = 0;
        buttonZero.setEnabled(false);
    }

    /**
     Метод для стирания числа из поля для ввода, а так же включение
возможностей нажимать на кнопки (за исключением нуля, т.к по правилам
игры ноль не может быть первой цифрой в числе. Этот метод
     используется при нажатии на кнопку "сброс" или кнопку "проверить".
    */
    private void clearNum() {
        buttonOne.setEnabled(true);
        buttonTwo.setEnabled(true);
        buttonThree.setEnabled(true);
        buttonFour.setEnabled(true);
        buttonFive.setEnabled(true);
        buttonSix.setEnabled(true);
        buttonSeven.setEnabled(true);
        buttonEight.setEnabled(true);
        buttonNine.setEnabled(true);
        buttonZero.setEnabled(false);
        numberView.setText("");
        buttonCheck.setEnabled(false);
    }

    /**
Отключение возможности нажатия на все кнопки, кроме сброса и проверки.
Используется в случае, когда набранное человеком число уже состоит из
четырех цифр, которое можно либо сереть, либо проверить.
  */
    public void blockButtons() {
        buttonOne.setEnabled(false);
        buttonTwo.setEnabled(false);
        buttonThree.setEnabled(false);
        buttonFour.setEnabled(false);
        buttonFive.setEnabled(false);
        buttonSix.setEnabled(false);
        buttonSeven.setEnabled(false);
        buttonEight.setEnabled(false);
        buttonNine.setEnabled(false);
```

```
        buttonZero.setEnabled(false);
    }

    /** Запуск проверки(основной логики игры) для введенного в поле числа
и вызов метода clearNum, после нажатия на кнопку "проверить".
   */
    public void onClickCheck(View v) {
        gameLogic();
        clearNum();
    }

    /** Основная логика игры. */
private void gameLogic() {
        digitCounter = 0;
        nullCounter = 0;
        String bullstr = " ";
        String cowstr = " ";
        ArrayList<Integer> mysteryListNum = splitStr(num);
        int scanNum = Integer.parseInt(numberView.getText().toString());
        String str = "В числе ";
        int bulls = 0;
        int cows = 0;
        attemptCounter += 1;
            if(attemptCounter < 12){
                ArrayList<Integer> scanListNum = getListNums(scanNum);
                for(int a = 0; a < 4; a++) {
                    for (int b = 0; b < 4; b++) {
                        if
(scanListNum.get(a).equals(mysteryListNum.get(b))) {
                            if (a == b) bulls += 1;
                            else cows += 1;
                        }
                    }
                }
                if (bulls == 4) showInfoAlert("Число " + scanNum + "
угадано, ПОБЕДА!");
                else {
                    if (bulls == 0) bullstr=" быков и ";
                    else if(bulls == 1) bullstr=" бык и ";
                    else if(bulls == 2 || bulls==3 || bulls==4) bullstr="
быка и ";
                    if (cows == 0) cowstr=" коров.";
                    else if(cows == 1) cowstr=" корова.";
                    else if(cows == 2 || cows==3 || cows==4) cowstr="
коровы.";

                    resultsView.setText(resultsView.getText() + "\n" +
attemptCounter + ") " + str + scanNum + " " + bulls + bullstr + cows +
cowstr );
                    if (attemptCounter == 11) showInfoAlert( "Попытки
кончились. поражение" + "\n" + "Загаданное число - " + num );
                }
            }
```

```
            else showInfoAlert( "Попытки кончились. поражение" + "\n" +
"Загаданное число - " + num );
    }
    /** Генерация рандомного числа для игры. */
    public int randomNum() {
        Random r = new Random();
        int d1 = r.nextInt(8) + 1;
        int d2 = r.nextInt(9);
        int d3 = r.nextInt(9);
        int d4 = r.nextInt(9);
        while (d1 == d2 || d1 == d3 || d1 == d4 || d2 == d3 || d2 == d4
|| d3 == d4) {
            if (d1 == d2 || d2 == d3 || d2 == d4) {
                d2 = r.nextInt(9);
            }
            if (d1 == d3 || d2 == d3 || d3 == d4) {
                d3 = r.nextInt(9);
            }
            if (d1 == d4 || d2 == d4 || d3 == d4) {
                d4 = r.nextInt(9);
            }
        }
        return Integer.parseInt(d1 + "" + d2 + "" + d3 + "" + d4);
    }

    /** Получение из введенной строки списка(листа) цифр. */
    private ArrayList<Integer> getListNums(int scanNum) {
        ArrayList<Integer> scanListNum = splitStr(scanNum);
        return scanListNum;
    }

    /**
     Разделение введенного числа на цифры, а так же запись их в
список(лист), для последующей проверки на наличие "быков" и "коров".
   */
    private ArrayList<Integer> splitStr(int ch) {
        ArrayList<Integer> list = new ArrayList();
        list.add(ch/1000);
        list.add(ch%1000/100);
        list.add(ch%100/10);
        list.add(ch%10);
        return list;
    }

    /** Проверка, что введено не более 4х цифр. */
    public void digitCounterCheck() {
        digitCounter++;
        if (digitCounter == 4) {
            blockButtons();
            buttonCheck.setEnabled(true);
        }
    }

    /** Проверка, что первая цифра введенного числа не является нулем. */
```

```
    public void digitZeroCheck() {
        if(nullCounter == 0) {
            if (digitCounter > 0 && digitCounter < 4) {
                buttonZero.setEnabled(true);
            }
        }
    }

    /**
     Метод для появления аллерта(всплывающего окна) при победе или
поражении в игре с последующим единственно возможным действием - выходом
в меню (класс MainActivity).
  */
 private void showInfoAlert(String textAlert) {
        AlertDialog.Builder alertBuilder= new
AlertDialog.Builder(SecondActivity.this);
        alertBuilder.setTitle("Внимание!!!")
                .setMessage(textAlert)
                .setCancelable(false)
                .setPositiveButton("Выходв меню", new
DialogInterface.OnClickListener()     {

                    @Override
                    public void onClick(DialogInterface dialogInterface,
int i) {
                        Intent intent = new Intent(SecondActivity.this,
MainActivity.class);
                       /* Переход от класса SecondActivity(экран с
уровнем игры) к классу
                     MainActivity(стартовый экрану с помощью нажатия на
кнопку
                     “назад"
                       */
                        startActivity(intent);
                    }});
        AlertDialog dialog = alertBuilder.create();
        dialog.show();
    }
}
```