**public class** StartTestActivity **extends** AppCompatActivity {  
  
 ImageButton **imgStartTest**;  
 EditText **etFacNumber**;  
 ImageButton **imgLogin**;  
 TextView **tvWelcome**;  
 TextView **checkFacultyNumber**;  
 TextView **tvSpecialityAndYear**;  
 Spinner **optionSpinner**;  
 ArrayAdapter<CharSequence> **adapter**;  
 String **testType**;  
 EditText **specialityAndYear**;  
 **private final** Gson **gson** = **new** GsonBuilder().create();  
  
  
 **protected void** onCreate(Bundle savedInstanceState) {  
  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_start\_test***);  
  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 toolbar.setTitle(**"Start of the test"**);  
 toolbar.setTitleTextColor(Color.***WHITE***);  
 setSupportActionBar(toolbar);  
  
  
 **checkFacultyNumber** = (TextView) findViewById(R.id.***tvFacultyNumber***);  
 **tvSpecialityAndYear** = (TextView) findViewById(R.id.***tvSpecialityAndYear***);  
 **specialityAndYear** = (EditText) findViewById(R.id.***specialityAndYearEditText***);  
 **imgStartTest** = (ImageButton) findViewById(R.id.***StartTest***);  
 **imgStartTest**.setVisibility(View.***INVISIBLE***);  
 *buttonEffect*(**imgStartTest**);  
 **imgLogin** = (ImageButton) findViewById(R.id.***ibStart***);  
 **etFacNumber** = (EditText) findViewById(R.id.***studentEditText***);  
 **tvWelcome** = (TextView) findViewById(R.id.***tvWelcome***);  
 **optionSpinner** = (Spinner) findViewById(R.id.***optionSpinner***);  
 **imgLogin**.setOnClickListener(**onLoginClickListener**);  
 **imgStartTest**.setOnClickListener(**start**);  
 **optionSpinner**.setVisibility(View.***INVISIBLE***);  
 }  
  
 View.OnClickListener **onLoginClickListener** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 **if** (TextUtils.*isEmpty*(**etFacNumber**.getText().toString())) {  
 **etFacNumber**.setError(**"Полето не е попълнено"**);  
 } **else** {  
 login();  
 }  
 }  
 };  
  
 **public** View.OnClickListener **start** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if** (TextUtils.*isEmpty*(**testType**)) {} **else** {  
 startActivity(QuestionActivity.*newIntent*(StartTestActivity.**this**, **testType**));  
 }  
 }  
 };  
  
 **private void** login() {  
 **new** LoginAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
 UserResponse userResponse = **gson**.fromJson(response, UserResponse.**class**);  
 **if** (userResponse==**null**) {  
 showError(**"Server error"**);  
 } **else if**(TextUtils.*isEmpty*(userResponse.**name**) || userResponse.**name**.equalsIgnoreCase(**"null"**)) {  
 showError(**"Wrong credentials"**);  
 } **else** {  
 getTestTypes();  
 }  
 }  
 }).execute(**etFacNumber**.getText().toString());  
 }  
 **private void** getTestTypes() {  
 **new** TypeTestsAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
 TestTypes testTypes = **gson**.fromJson(response, TestTypes.**class**);  
 **adapter** = **new** ArrayAdapter(StartTestActivity.**this**, android.R.layout.***simple\_spinner\_item***);  
 **adapter**.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
 **adapter**.addAll(testTypes.**typeTests**);  
 **optionSpinner**.setAdapter(**adapter**);  
 **optionSpinner**.setVisibility(View.***VISIBLE***);  
 **imgStartTest**.setVisibility(View.***VISIBLE***);  
 setOnTestTypeSelectedListener();  
 }  
 }).execute();  
 }  
  
 **private void** setOnTestTypeSelectedListener() {  
 **optionSpinner**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
  
 ((TextView) parent.getChildAt(0)).setTextColor(Color.***WHITE***);  
 **testType** = (String) **optionSpinner**.getSelectedItem();  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
 }  
  
 **public static void** buttonEffect(View button) {  
 button.setOnTouchListener(**new** View.OnTouchListener() {  
  
 **public boolean** onTouch(View v, MotionEvent event) {  
 **switch** (event.getAction()) {  
 **case** MotionEvent.***ACTION\_DOWN***: {  
 v.getBackground().setColorFilter(0xe0f47521, PorterDuff.Mode.***SRC\_ATOP***);  
 v.invalidate();  
 **break**;  
 }  
 **case** MotionEvent.***ACTION\_UP***: {  
 v.getBackground().clearColorFilter();  
 v.invalidate();  
 **break**;  
 }  
 }  
 **return false**;  
 }  
 });  
 }  
}

Този клас се грижи за логването на студент за решаване на тест.

При кликването на onLoginClickListener с код:

**if** (TextUtils.*isEmpty*(**etFacNumber**.getText().toString())) {  
 **etFacNumber**.setError(**"Полето не е попълнено"**);  
 } **else** {  
 login();  
 }  
 }  
};

се проверява дали в editText-a има въведена информация,ако тя не е правилна или полето не е попълнено се извежда грешка, ако всичко е наред се изпълнява метод login() с код за него :

**private void** login() {  
 **new** LoginAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
 UserResponse userResponse = **gson**.fromJson(response, UserResponse.**class**);  
 **if** (userResponse==**null**) {  
 showError(**"Server error"**);  
 } **else if**(TextUtils.*isEmpty*(userResponse.**name**) || userResponse.**name**.equalsIgnoreCase(**"null"**)) {  
 showError(**"Wrong credentials"**);  
 } **else** {  
 getTestTypes();  
 }  
 }  
 }).execute(**etFacNumber**.getText().toString());  
}

Тук в login() се взима respons-a от LoginAsyncTaks и се обръща в обект, но ако нещо се обърка тогава се ползва метода showError, за да се полкаже грешка:

**private void** showError(String message){  
 AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  
 builder.setTitle(**"Error"**)  
 .setMessage(message)  
 .show();  
}

В този метод login(), както споменах се ползва LoginAsyncTask

Код за Login AsyncTask:

**public class** LoginAsyncTask **extends** AsyncTask<String, Void, String> {  
  
 **private static final** String ***LOGIN\_URL*** = ApiConstants.***URL*** + **"LoginServlet?facNum="**;  
  
 **private** ApiCallback **apiCallback**;  
  
 **public** LoginAsyncTask (ApiCallback apiCallback){  
 **this**.**apiCallback** = apiCallback;  
 }  
   
 @Override  
 **protected** String doInBackground(String... params) {  
 URL url;  
 HttpURLConnection urlConnection;  
 BufferedReader br;  
 **try**{  
 url = **new** URL(***LOGIN\_URL*** + params[0] );  
  
 urlConnection = (HttpURLConnection) url.openConnection();  
  
 br = **new** BufferedReader(**new** InputStreamReader(  
 urlConnection.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = br.readLine();  
  
 **while** (line != **null**){  
 sb.append(line);  
 line = br.readLine();  
 }  
 **return** sb.toString();  
  
 }**catch** (MalformedURLException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String response) {  
  
 **apiCallback**.onResponse(response);  
 }  
 }

LoginAsyncTask се ползва за работа не в Main Thread-a на клиентското приложение, а в свой собствен Thread, за да не натоварва допълнително Main Thread-a. И с помощта на интерфейс ApiCollback елегантно се връща response до main tread-a на приложението.

Тук като допълнително разяснение ще добавя,че в папката с име async има един клас – ApiConstanst и един Интерфейс - ApiCallback.

ApiCallback с код:

**public interface** ApiCallback {  
 **void** onResponse(String response);  
}

Този интерфейс, както горе беше споменато, по елегантен начин изпраша respons-а получен от сървъра до клиента по необходимият URL

ApiConstants е с код:

**public class** ApiConstants {  
 **public static final** String ***URL*** = **"http://192.168.0.104:8080/Server/"**;  
  
}

Като между скобите се задава съответната част от URL-а, обща за приложението, а по-късно в AsyncTask-овете ще се ползва за написването на URL необходим за взимането на данните от сървъра.

Друг AsyncTask в папката async необходим в StartTest e TypeTestsAsyncTask с код:

**public class** TypeTestsAsyncTask **extends** AsyncTask<Void, Void, String> {  
  
 **private static final** String ***TYPE\_TESTS\_URL*** = ApiConstants.***URL*** + **"TypeTests"**;  
 String **content**;  
  
 **private** ApiCallback **apiCallback**;  
  
 **public** TypeTestsAsyncTask(ApiCallback apiCallback){  
 **this**.**apiCallback** = apiCallback;  
  
 }  
 @Override  
 **protected** String doInBackground(Void... params) {  
 URL url;  
 HttpURLConnection urlConnection;  
 BufferedReader br;  
 **try**{  
 url = **new** URL(***TYPE\_TESTS\_URL*** );  
  
 urlConnection = (HttpURLConnection) url.openConnection();  
  
 br = **new** BufferedReader(**new** InputStreamReader(  
 urlConnection.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = br.readLine();  
  
 **while** (line != **null**){  
 sb.append(line);  
 line = br.readLine();  
 }  
 **content** = sb.toString();  
  
 }**catch** (MalformedURLException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String response) {  
  
 **apiCallback**.onResponse(response);  
 }  
 }

Този AsyncTask е подобен на предишният, за логин-а, но тук вместо да се подава праметри в doInBackground() метода като предният път метода получава Void параметри.

Kато допълнение се ползва библиотеката на google “Gson” за създаване на Json обект.След като се вземе respons-a отново с помощта на интерфейс ApiCallback елегантно се връща response до main tread-a на приложението съответно в класа за логин в метод :

**private void** getTestTypes() {  
 **new** TypeTestsAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
 TestTypes testTypes = **gson**.fromJson(response, TestTypes.**class**);  
 **adapter** = **new** ArrayAdapter(StartTestActivity.**this**, android.R.layout.***simple\_spinner\_item***);  
 **adapter**.setDropDownViewResource(android.R.layout.***simple\_spinner\_dropdown\_item***);  
 **adapter**.addAll(testTypes.**typeTests**);  
 **optionSpinner**.setAdapter(**adapter**);  
 **optionSpinner**.setVisibility(View.***VISIBLE***);  
 **imgStartTest**.setVisibility(View.***VISIBLE***);  
 setOnTestTypeSelectedListener();  
 }  
 }).execute();  
 }

След като вече от TestTypesAsyncTask е получн respons-a и е обърнат в обект,този обект съответно се пълни в един adapter, който от своя страна се подава на Spinner с име optionSpinner и се показва в метод:

**private void** setOnTestTypeSelectedListener() {  
 **optionSpinner**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  
  
 ((TextView) parent.getChildAt(0)).setTextColor(Color.***WHITE***);  
 **testType** = (String) **optionSpinner**.getSelectedItem();  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> parent) {  
  
 }  
 });  
  
}

В папката model има клас с име TestTypes, който е модел за gson.

**public class** TestTypes {  
  
 **public** String[] **typeTests**;  
   
}

В папката ui освен StartTest клас също има и 2 класа, 1 от тях е QuestionActivity

с код :

**public class** QuestionActivity **extends** AppCompatActivity **implements** CountdownListener {  
  
 **private static final** String ***EXTRA\_TEST\_TYPE*** = **"testType"**;  
 **public static final** String ***ANDROID\_TEST\_TYPE*** = **"Answer\_table"**;  
 **public static final** String ***CSHARP\_TEST\_TYPE*** = **"CSharpAnswer\_table"**;  
 **public static final** String ***JAVA\_TEST\_TYPE*** = **"JavaAnswer\_table"**;

**double sumPoints**;  
 MainApplication **app**;  
 ObjectAnimator **anim**;  
 **int counter**;  
 **private** TextView **tvTimer**;  
 **private** ProgressBar **pbTimer**;  
 **private** TextView **tvQuestionText**;  
  
 **private** RadioGroup **rgAnswers**;  
 **private** LinearLayout **llCheckBoxes**;  
 **private** EditText **etFreeAnswer**;  
  
 **private** TextView **tvStatus**;  
 **private** Button **btnNextQuestion**;  
  
 **private** CheckBox **chOne**;  
 **private** CheckBox **chTwo**;  
 **private** CheckBox **chThree**;  
 **private** RadioButton **rbOne**;  
 **private** RadioButton **rbTwo**;  
 **private** RadioButton **rbThree**;  
 **private** Gson **gson** = **new** GsonBuilder().create();  
 **private** Question **currentQuestion**;  
  
 **public static** Intent newIntent(Context context, String testType) {  
 Intent intent = **new** Intent(context, QuestionActivity.**class**);  
 intent.putExtra(***EXTRA\_TEST\_TYPE***, testType);  
 **return** intent;  
 }  
  
 **private** String getExtraTestType() {  
 **return** getIntent().getStringExtra(***EXTRA\_TEST\_TYPE***);  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_question\_new***);  
  
 **app** = (MainApplication) getApplication();  
  
 **tvQuestionText** = (TextView) findViewById(R.id.***tvQuestionText***);  
 **btnNextQuestion** = (Button) findViewById(R.id.***btnNext***);  
 **btnNextQuestion**.setOnClickListener(**nextQuestionOnClickListener**);  
 **btnNextQuestion**.setEnabled(**false**);  
 **btnNextQuestion**.setAlpha(0.1f);  
  
 **tvStatus** = (TextView) findViewById(R.id.***tvStatus***);  
 **tvStatus**.setText(**"Все още не е избран отговор!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
  
  
 **llCheckBoxes** = (LinearLayout) findViewById(R.id.***llCheckBoxes***);  
 **rgAnswers** = (RadioGroup) findViewById(R.id.***rgAnswers***);  
 **etFreeAnswer** = (EditText) findViewById(R.id.***etFreeAnswer***);  
  
 **chOne** = (CheckBox) findViewById(R.id.***checkBox1***);  
 **chTwo** = (CheckBox) findViewById(R.id.***checkBox2***);  
 **chThree** = (CheckBox) findViewById(R.id.***checkBox3***);  
  
 **rbOne** = (RadioButton) findViewById(R.id.***radioButton1***);  
 **rbTwo** = (RadioButton) findViewById(R.id.***radioButton2***);  
 **rbThree** = (RadioButton) findViewById(R.id.***radioButton3***);  
  
 **pbTimer** = (ProgressBar) findViewById(R.id.***pbTimer***);  
 **tvTimer** = (TextView) findViewById(R.id.***tvTimer***);  
  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 setSupportActionBar(toolbar);  
 toolbar.setTitleTextColor(Color.***WHITE***);  
 **counter** = 0;  
   
  
 setTitle();  
 initTimer();  
 readQuestionAndAnswerFromServer();  
 animateProgressBar();  
 }  
  
 **private void** setTitle() {  
 **switch** (getExtraTestType()) {  
 **case *ANDROID\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"ANDROID TEST"**);  
 **break**;  
 **case *CSHARP\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"CSHARP TEST"**);  
 **break**;  
 **case *JAVA\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"JAVA TEST"**);  
  
 **break**;  
 }  
 }  
  
 @Override  
 **protected void** onResume() {  
 **super**.onResume();  
 **app**.startTimer();  
 }  
  
 @Override  
 **protected void** onPause() {  
 **super**.onPause();  
 **app**.pauseTimer();  
 }  
  
  
 **private void** addFreeTextChangeListener() {  
 **etFreeAnswer**.addTextChangedListener(**new** TextWatcher() {  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count, **int** after) {  
  
 }  
  
 @Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before, **int** count) {  
 **if** (count == 0) {  
 **tvStatus**.setText(**"Все още не е въведен текст!!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
 **btnNextQuestion**.setEnabled(**false**);  
 **btnNextQuestion**.setAlpha(0.1f);  
 } **else** {  
 **tvStatus**.setText(**"Вече е въведен текс!"**);  
 **tvStatus**.setTextColor(Color.***GREEN***);  
 **btnNextQuestion**.setEnabled(**true**);  
 **btnNextQuestion**.setAlpha(0.5f);  
 }  
 }  
  
 @Override  
 **public void** afterTextChanged(Editable s) {  
  
 }  
 });  
 }  
  
 **private void** validateText() {  
  
 **tvStatus**.setText(**"Все още не е въведен текст!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
  
  
 }  
  
 **private void** readQuestionAndAnswerFromServer() {  
 **new** QAAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
   
 **currentQuestion** = **gson**.fromJson(response, Question.**class**);  
  
 showQuestionLayout(**currentQuestion**);  
 }  
 }).execute(getExtraTestType());  
 }  
  
 **private void** showQuestionLayout(Question question) {  
 **switch** (question.**type**) {  
 **case** AnswerType.***RADIO\_BUTTON***: {  
 **rgAnswers**.setVisibility(***VISIBLE***);  
 **llCheckBoxes**.setVisibility(***GONE***);  
 **etFreeAnswer**.setVisibility(***GONE***);  
 populateRadioButtons(question);  
 setRadioGroupSelectedListener();  
 **break**;  
 }  
 **case** AnswerType.***CHECK\_BOXES***: {  
 **rgAnswers**.setVisibility(***GONE***);  
 **llCheckBoxes**.setVisibility(***VISIBLE***);  
 **etFreeAnswer**.setVisibility(***GONE***);  
 populateCheckBoxes(question);  
 setCheckBoxesChangeListeners();  
 **break**;  
 }  
 **case** AnswerType.***EDIT\_TEXT***: {  
 **rgAnswers**.setVisibility(***GONE***);  
 **llCheckBoxes**.setVisibility(***GONE***);  
 **etFreeAnswer**.setVisibility(***VISIBLE***);  
 addFreeTextChangeListener();  
 **break**;  
 }  
 }  
 **tvQuestionText**.setText(question.**questionText**);  
 }  
  
  
 **private void** populateRadioButtons(Question question) {  
 **rbOne**.setText(question.**answers**.get(0).**answerText**);  
 **rbTwo**.setText(question.**answers**.get(1).**answerText**);  
 **rbThree**.setText(question.**answers**.get(2).**answerText**);  
 }  
  
 **private void** populateCheckBoxes(Question question) {  
 **chOne**.setText(question.**answers**.get(0).**answerText**);  
 **chTwo**.setText(question.**answers**.get(1).**answerText**);  
 **chThree**.setText(question.**answers**.get(2).**answerText**);  
 }  
  
  
 **private void** setCheckBoxesChangeListeners() {  
  
 **chOne**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
  
 **chTwo**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
  
 **chThree**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
 }  
  
  
 **private void** enableNext() {  
 **tvStatus**.setText(**"Отговор бе избран!"**);  
 **tvStatus**.setTextColor(Color.***GREEN***);  
 **btnNextQuestion**.setEnabled(**true**);  
 **btnNextQuestion**.setAlpha(0.5f);  
 }  
  
 **private void** wholeGrade() {  
 Toast toast = Toast.*makeText*(QuestionActivity.**this**, **"Брой точки: "** + String.*valueOf*(**sumPoints**), Toast.***LENGTH\_SHORT***);  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
 }  
  
 **public void** animateProgressBar() {  
 ProgressBar mProgressBar = (ProgressBar) findViewById(R.id.***progressBarFirstQuestion***);  
 **anim** = ObjectAnimator.*ofInt*(mProgressBar, **"progress"**, 100, 0);  
 **anim**.setDuration(MainApplication.***TEST\_DURATION\_IN\_MILLIS***);  
 **anim**.setInterpolator(**new** DecelerateInterpolator());  
 **anim**.start();  
 }  
  
 **private void** setRadioGroupSelectedListener() {  
 **rgAnswers**.setOnCheckedChangeListener(**new** OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(RadioGroup group, **int** checkedId) {  
  
 **tvStatus**.setText(**"Отговор бе избран!"**);  
 **tvStatus**.setTextColor(Color.***GREEN***);  
  
 **btnNextQuestion**.setEnabled(**true**);  
 **btnNextQuestion**.setAlpha(0.5f);  
  
 }  
  
 });  
 }  
  
 View.OnClickListener **nextQuestionOnClickListener** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **switch** (**currentQuestion**.**type**) {  
 **case** 1:  
 **int** currentCheckedId = **rgAnswers**.getCheckedRadioButtonId();  
 RadioButton rbCheckedAnswer = (RadioButton) findViewById(currentCheckedId);  
 String radioAnswer = rbCheckedAnswer.getText().toString();  
 **for** (com.example.sashopc.delcandroidtest.model.Answer answer : **currentQuestion**.**answers**) {  
 **if** (answer.**answerText**.equalsIgnoreCase(radioAnswer) && answer.**isCorrect**) {  
 **sumPoints** += 5;  
 }  
 }  
 rbCheckedAnswer.setChecked(**false**);  
 **break**;  
 **case** 2:  
 **for** (**int** i = 0; i < **currentQuestion**.**answers**.size(); i++) {  
 Answer answer = **currentQuestion**.**answers**.get(i);  
 **if** (answer.**answerText**.equalsIgnoreCase(**chOne**.getText().toString()) && answer.**isCorrect** && **chOne**.isChecked()) {  
 **sumPoints** += 2.5;  
 } **else if** (answer.**answerText**.equalsIgnoreCase(**chTwo**.getText().toString()) && answer.**isCorrect** && **chTwo**.isChecked()) {  
 **sumPoints** += 2.5;  
 } **else if** (answer.**answerText**.equalsIgnoreCase(**chThree**.getText().toString()) && answer.**isCorrect** && **chThree**.isChecked()) {  
 **sumPoints** += 2.5;  
 }  
 }  
 }  
  
 readQuestionAndAnswerFromServer();  
 validateText();  
 **btnNextQuestion**.setEnabled(**false**);  
 **tvStatus**.setText(**"Все още не е избран отговор!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
 **btnNextQuestion**.setAlpha(0.1f);  
  
 **chOne**.setChecked(**false**);  
 **chTwo**.setChecked(**false**);  
 **chThree**.setChecked(**false**);  
  
 **rbOne**.setChecked(**false**);  
 **rbTwo**.setChecked(**false**);  
 **rbThree**.setChecked(**false**);  
  
 **etFreeAnswer**.setText(**null**);  
  
 wholeGrade();  
 **counter**++;  
  
  
 **if** (**counter** == 6) {  
 Intent moveToEnd = **new** Intent(QuestionActivity.**this**, EndTestActivity.**class**);  
 moveToEnd.putExtra(**"Grade"**, **sumPoints**);  
 startActivity(moveToEnd);  
 }  
  
 }  
 };  
  
 **private void** initTimer() {  
 **app**.setCountdownListener(**this**);  
 **pbTimer**.setMax((**int**) MainApplication.***TEST\_DURATION\_IN\_MILLIS***);  
 }  
  
 @Override  
 **public void** timerTick(**long** millisUntilFinished) {  
 **tvTimer**.setText(String.*format*(**"%d : %d"**,  
 TimeUnit.***MILLISECONDS***.toMinutes(millisUntilFinished),  
 TimeUnit.***MILLISECONDS***.toSeconds(millisUntilFinished) -  
 TimeUnit.***MINUTES***.toSeconds(TimeUnit.***MILLISECONDS***.toMinutes(millisUntilFinished))));  
  
 **pbTimer**.setProgress((**int**) millisUntilFinished);  
 }  
  
 @Override  
 **public void** onBackPressed() {  
 Toast.*makeText*(QuestionActivity.**this**, **"BackPressButton Disabled!"**, Toast.***LENGTH\_SHORT***).show();  
 }  
}

Този код се грижи за получаването на въпрос чрез метод с код:

**private void** readQuestionAndAnswerFromServer() {  
 **new** QAAsyncTask(**new** ApiCallback() {  
 @Override  
 **public void** onResponse(String response) {  
   
 currentQuestion = **gson**.fromJson(response, Question.**class**);  
   
 showQuestionLayout(currentQuestion);  
 }  
 }).execute(getExtraTestType());  
}

Като с помощта на QAAsyncTask, който е описан по горе в документацията, gson библеотеката и на базата на getExtraTestType() се построява обект от получения response от сървъра и елегантно се връща на main Thread-a на клиентското приложение чрез ApiCallbask интерфеиса.

Тук getExtraTestType() e метод :

**private** String getExtraTestType() {  
 **return** getIntent().getStringExtra(EXTRA\_TEST\_TYPE);  
}

за получаването на extra свързана с типа на теста ( пояснявам, че има 3 типа тест които са Android\_test,CSharp\_test,Java\_test и за трите типа има въпроси и отговори ), за това въпросите и отговорите са базирани на типа на теста.

Метода:

**private void** setTitle() {  
 **switch** (getExtraTestType()) {  
 **case *ANDROID\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"ANDROID TEST"**);  
 **break**;  
 **case *CSHARP\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"CSHARP TEST"**);  
 **break**;  
 **case *JAVA\_TEST\_TYPE***:  
 getSupportActionBar().setTitle(**"JAVA TEST"**);  
  
 **break**;  
 }  
}

Се грижи за заглавието на ActionBar-a в зависимост отново на getExtraTestType() метода .

Метода с код:

**private void** addFreeTextChangeListener() {  
 **etFreeAnswer**.addTextChangedListener(**new** TextWatcher() {  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count, **int** after) {  
  
 }  
  
 @Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before, **int** count) {  
 **if** (count == 0) {  
 **tvStatus**.setText(**"Все още не е въведен текст!!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
 btnNextQuestion.setEnabled(**false**);  
 btnNextQuestion.setAlpha(0.1f);  
 } **else** {  
 **tvStatus**.setText(**"Вече е въведен текс!"**);  
 **tvStatus**.setTextColor(Color.***GREEN***);  
 btnNextQuestion.setEnabled(**true**);  
 btnNextQuestion.setAlpha(0.5f);  
 }  
 }  
  
 @Override  
 **public void** afterTextChanged(Editable s) {  
  
 }  
 });  
}

В този метод е имплементирана валидацията на единия от типовете отговори ( именно това е отговор със свободен от студента текст ).

На него е закъчен addTextChangeListener(), като при textChanged се извършва самата валидация.

Метода с код :

**private void** showQuestionLayout(Question question) {  
 **switch** (question.**type**) {  
 **case** AnswerType.***RADIO\_BUTTON***: {  
 **rgAnswers**.setVisibility(***VISIBLE***);  
 **llCheckBoxes**.setVisibility(***GONE***);  
 **etFreeAnswer**.setVisibility(***GONE***);  
 populateRadioButtons(question);  
 setRadioGroupSelectedListener();  
 **break**;  
 }  
 **case** AnswerType.***CHECK\_BOXES***: {  
 **rgAnswers**.setVisibility(***GONE***);  
 **llCheckBoxes**.setVisibility(***VISIBLE***);  
 **etFreeAnswer**.setVisibility(***GONE***);  
 populateCheckBoxes(question);  
 setCheckBoxesChangeListeners();  
 **break**;  
 }  
 **case** AnswerType.***EDIT\_TEXT***: {  
 **rgAnswers**.setVisibility(***GONE***);  
 **llCheckBoxes**.setVisibility(***GONE***);  
 **etFreeAnswer**.setVisibility(***VISIBLE***);  
 addFreeTextChangeListener();  
 **break**;  
 }  
 }  
 **tvQuestionText**.setText(question.**questionText**);  
}

В този метод се попълват 3те layout-a за 3те отговора. Като 3те layout-а репрезентират единичен правилен отговор, множество от правилни отговори и отговор със свободен текст.

На базата на типа на Question –a преобърнат в обект от от gson библеотеката се пуска switch () case {} съответно за 3те отговора. Всеки case е отговор и на базата на това се попълва конкретният layout.

Методите :

**private void** populateRadioButtons(Question question) {  
 rbOne.setText(question.**answers**.get(0).**answerText**);  
 **rbTwo**.setText(question.**answers**.get(1).**answerText**);  
 **rbThree**.setText(question.**answers**.get(2).**answerText**);  
}  
  
**private void** populateCheckBoxes(Question question) {  
 **chOne**.setText(question.**answers**.get(0).**answerText**);  
 **chTwo**.setText(question.**answers**.get(1).**answerText**);  
 **chThree**.setText(question.**answers**.get(2).**answerText**);  
}

Са специално за попълването на текста (setText) на radio buttons респективно за първият метод и за check boxes за вторият, а за text area-та съответно няма такъв метод, защото както подсказва името отговора ще бъде въведен от студента.

Метода с код:

**private void** setRadioGroupSelectedListener() {  
 **rgAnswers**.setOnCheckedChangeListener(**new** OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(RadioGroup group, **int** checkedId) {  
  
 **tvStatus**.setText(**"Отговор бе избран!"**);  
 **tvStatus**.setTextColor(Color.***GREEN***);  
  
 **btnNextQuestion**.setEnabled(**true**);  
 **btnNextQuestion**.setAlpha(0.5f);  
  
 }  
  
 });  
}

Отново има за цел да валидира бутона за следващ въпрос, но този път от гледна точка на радио група-та, самият метод се вика във вече описан от мен метод с име showQuestionLayout ()

Като за него (леко да подскажа ) имаше значение типа на въпроса и от него в един switch () case {} за отговорите се определяха и layout-ите, който се показват в приложението.

Там специално за layout-a за рдио групата се поставя метода с име setRadioGroupSelectedListener();

Съответно логиката е еднаква за чек боксовете, с метод :

**private void** setCheckBoxesChangeListeners() {  
  
 **chOne**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
  
 **chTwo**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
  
 **chThree**.setOnCheckedChangeListener(**new** CompoundButton.OnCheckedChangeListener() {  
 @Override  
 **public void** onCheckedChanged(CompoundButton buttonView, **boolean** isChecked) {  
 enableNext();  
 }  
 });  
}

Като в метода с име enableNext()

се случва валидацията,която позволява да се отиде на друг ( генериран на случаен принцип ) въпрос.

Методът с код :

View.OnClickListener **nextQuestionOnClickListener** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **switch** (**currentQuestion**.**type**) {  
 **case** 1:  
 **int** currentCheckedId = **rgAnswers**.getCheckedRadioButtonId();  
 RadioButton rbCheckedAnswer = (RadioButton) findViewById(currentCheckedId);  
 String radioAnswer = rbCheckedAnswer.getText().toString();  
 **for** (com.example.sashopc.delcandroidtest.model.Answer answer : **currentQuestion**.**answers**) {  
 **if** (answer.**answerText**.equalsIgnoreCase(radioAnswer) && answer.**isCorrect**) {  
 **sumPoints** += 5;  
 }  
 }  
 rbCheckedAnswer.setChecked(**false**);  
 **break**;  
 **case** 2:  
 **for** (**int** i = 0; i < **currentQuestion**.**answers**.size(); i++) {  
 Answer answer = **currentQuestion**.**answers**.get(i);  
 **if** (answer.**answerText**.equalsIgnoreCase(**chOne**.getText().toString()) && answer.**isCorrect** && **chOne**.isChecked()) {  
 **sumPoints** += 2.5;  
 } **else if** (answer.**answerText**.equalsIgnoreCase(**chTwo**.getText().toString()) && answer.**isCorrect** && **chTwo**.isChecked()) {  
 **sumPoints** += 2.5;  
 } **else if** (answer.**answerText**.equalsIgnoreCase(**chThree**.getText().toString()) && answer.**isCorrect** && **chThree**.isChecked()) {  
 **sumPoints** += 2.5;  
 }  
 }  
 }  
  
 readQuestionAndAnswerFromServer();  
 validateText();  
 btnNextQuestion.setEnabled(**false**);  
 **tvStatus**.setText(**"Все още не е избран отговор!"**);  
 **tvStatus**.setTextColor(Color.***RED***);  
 btnNextQuestion.setAlpha(0.1f);  
  
 **chOne**.setChecked(**false**);  
 **chTwo**.setChecked(**false**);  
 **chThree**.setChecked(**false**);  
  
 **rbOne**.setChecked(**false**);  
 **rbTwo**.setChecked(**false**);  
 **rbThree**.setChecked(**false**);  
  
 **etFreeAnswer**.setText(**null**);  
  
 wholeGrade();  
 **counter**++;  
  
 Log.*d*(**"asd"**, **"sumPoints: "** + **sumPoints**);  
  
 **if** (**counter** == 6) {  
 Intent moveToEnd = **new** Intent(QuestionActivity.**this**, EndTestActivity.**class**);  
 moveToEnd.putExtra(**"Grade"**, **sumPoints**);  
 startActivity(moveToEnd);  
 }  
  
 }  
};

Не е нищо повече от един listener , който слуша за натискането на бутон отговарящ за следващият въпрос, и на базата на това дали е бил single answer ( radio button ) или multiple answers (check boxes ) раздава точките за избраният отговор.

Като също така се грижи един след друг въпросите да са 6 и когато се отговорят тези въпроси да се преминава на EndTestActivity.

Методът с име getPointsAsWhole () с код:

**private void** getPointsAsWhole() {  
 Toast toast = Toast.*makeText*(QuestionActivity.**this**, **"Брой точки: "** + String.*valueOf*(**sumPoints**), Toast.***LENGTH\_SHORT***);  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
}

Взима точките от 6те въпроса и ги пoставя като extra за Intent-a необходим за стартирането на EndTestActivity:

Intent moveToEnd = **new** Intent(QuestionActivity.**this**, EndTestActivity.**class**);  
moveToEnd.putExtra(**"Grade"**, sumPoints);  
startActivity(moveToEnd);

Още един AsyncTask в папката async е QAAsyncTask с код:

**public class** QAAsyncTask **extends** AsyncTask<String, Void, String> {  
  
 **private static final** String ***TYPE\_TESTS\_URL*** = ApiConstants.***URL*** + **"QuestionAndAnswerServlet?testType="**;  
  
 **private** ApiCallback **apiCallback**;  
  
 **public** QAAsyncTask(ApiCallback apiCallback){  
 **this**.**apiCallback** = apiCallback;  
  
 }  
 @Override  
 **protected** String doInBackground(String... params) {  
 URL url;  
 HttpURLConnection urlConnection;  
 BufferedReader br;  
 **try**{  
 url = **new** URL(***TYPE\_TESTS\_URL*** + params[0]);  
  
 urlConnection = (HttpURLConnection) url.openConnection();  
  
 br = **new** BufferedReader(**new** InputStreamReader(  
 urlConnection.getInputStream()));  
  
 StringBuilder sb = **new** StringBuilder();  
 String line = br.readLine();  
  
 **while** (line != **null**){  
 sb.append(line);  
 line = br.readLine();  
 }  
 **return** sb.toString();  
  
 }**catch** (MalformedURLException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **return null**;  
 }  
  
 @Override  
 **protected void** onPostExecute(String response) {  
   
 **apiCallback**.onResponse(response);  
 }  
 }

Този AsyncTask e много подобен на LoginAsyncTask, но с тази разлика, че вместо да връща студент за логване в този случай той връща въпросите на които трябва да се отговори от студента. В AsyncTask-a отново се ползва ApiCallback, за връщане на въпрос от Thread-a в mainThread-a на приложението и този път в метода doInBackground() приема параметри от тип String.

В папката model, освен TestTypes има и класове за Question който служи за точен модел от базата данни( maping с таблицата от базата данни ) с код:   
 **public class** Question {  
 **public int id**;  
 **public** String **questionText**;  
 **public int type**;  
 **public** List<Answer> **answers**;  
  
 @Override  
 **public** String toString() {  
 **return "Question{"** +  
 **"id="** + **id** +  
 **", questionText='"** + **questionText** + **'\''** +  
 **", type='"** + **type** + **'\''** +  
 **", answers="** + **answers** +  
 **'}'**;  
 }   
 }

С метод toString() за да се получи съответния въпрос, а не някакви непознати символи. Този клас се прави от програмиста ръчно с цел след това gson библеотеката да преобразува полученият string (response ) обратно в обект.

Answer с код:

**public class** Answer {  
 **public int** id;  
 **public int questionId**;  
 **public** String **answerText**;  
 **public boolean isCorrect**;  
  
 @Override  
 **public** String toString() {  
 **return "Answer{"** +  
 **"id="** + id +  
 **", questionId="** + **questionId** +  
 **", answerText='"** + **answerText** + **'\''** +  
 **", isCorrect="** + **isCorrect** +  
 **'}'**;  
 }  
 }

При този клас е същата история както с Question класа, но тои е за Отговор

И последното Activity е с име EndTestActivity.

То е с код:   
**public class** EndTestActivity **extends** AppCompatActivity {  
  
 **double wholeGrade**;  
 ImageButton **sumAnswers**;  
 ImageButton **closeApp**;  
 **int grade**;  
 TextView **moreInformation**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_end\_test***);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 toolbar.setTitle(**"End of the test"**);  
 toolbar.setTitleTextColor(Color.***WHITE***);  
 setSupportActionBar(toolbar);  
 **closeApp** = (ImageButton)findViewById(R.id.***ibClose***);  
 **sumAnswers**= (ImageButton) findViewById(R.id.***ibCheckScale***);  
 **sumAnswers**.setOnClickListener(**moreInfoAboutTest**);  
 **closeApp**.setOnClickListener(**closeApplication**);  
 getWholeGrade();  
 }  
 **private void** getWholeGrade(){  
  
 **wholeGrade** = getIntent().getDoubleExtra(**"Grade"**,**grade**);  
 **if**(**wholeGrade**>25.5 && **wholeGrade**<=30){  
 **grade**=6;  
 }**else if**(**wholeGrade**>20.5 && **wholeGrade**<=25.5){  
 **grade** = 5;  
 }**else if**(**wholeGrade**>15.5 && **wholeGrade**<=20.5){  
 **grade** = 4;  
 }**else if**(**wholeGrade**>=10.5 && **wholeGrade**<=15.5){  
 **grade** = 3;  
 }**else**{  
 **grade** = 2;  
 }  
 Toast toast = Toast.*makeText*(EndTestActivity.**this**, **"Оценка от теста: "** + String.*valueOf*(**grade**), Toast.***LENGTH\_SHORT***);  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
 }  
 View.OnClickListener **moreInfoAboutTest** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **moreInformation** = (TextView)findViewById(R.id.***tvMoreInfo***);  
 **moreInformation**.setText(**"Скала за оценяване: \nпод 10.5 точки 2\n"** +  
 **"от 10.5 до 15.5 точки: 3\n"** +  
 **"от 16.5 до 20.5 точки: 4\n"** +  
 **"от 21.5 до 25.5 точки: 5\n"** +  
 **"от 26.5 до 30 точки: 6\n"**);  
  
 }  
 };  
  
 View.OnClickListener **closeApplication** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Toast toast = Toast.*makeText*(EndTestActivity.**this**, **"Android test ще се затвори!"**, Toast.***LENGTH\_SHORT***);  
  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
  
 finishAffinity();  
 }  
 };  
  
}

Като този клас има за цел да показва скалата за оценяване чрез метод:

**private void** getWholeGrade(){  
  
 **wholeGrade** = getIntent().getDoubleExtra(**"Grade"**,**grade**);  
  
 **if**(**wholeGrade**>25.5 && **wholeGrade**<=30){  
 **grade**=6;  
 }**else if**(**wholeGrade**>20.5 && **wholeGrade**<=25.5){  
 **grade** = 5;  
 }**else if**(**wholeGrade**>15.5 && **wholeGrade**<=20.5){  
 **grade** = 4;  
 }**else if**(**wholeGrade**>=10.5 && **wholeGrade**<=15.5){  
 **grade** = 3;  
 }**else**{  
 **grade** = 2;  
 }  
 Toast toast = Toast.*makeText*(EndTestActivity.**this**, **"Оценка от теста: "** + String.*valueOf*(**grade**), Toast.***LENGTH\_SHORT***);  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
}

на базата на получените точки от QuestionActivity.

Да показва в едно TextView точната скал за оценяването чрез метода:

View.OnClickListener **moreInfoAboutTest** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **moreInformation** = (TextView)findViewById(R.id.***tvMoreInfo***);  
 **moreInformation**.setText(**"Скала за оценяване: \nпод 10.5 точки 2\n"** +  
 **"от 10.5 до 15.5 точки: 3\n"** +  
 **"от 16.5 до 20.5 точки: 4\n"** +  
 **"от 21.5 до 25.5 точки: 5\n"** +  
 **"от 26.5 до 30 точки: 6\n"**);  
  
 }  
};

И излизането от тест при натискането на бутон за излизане от клиентското приложение:

View.OnClickListener **closeApplication** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Toast toast = Toast.*makeText*(EndTestActivity.**this**, **"Android test ще се затвори!"**, Toast.***LENGTH\_SHORT***);  
  
 View toastView = toast.getView();  
  
 toastView.setBackgroundResource(R.drawable.***toast\_message\_style***);  
  
 toast.show();  
  
 finishAffinity();  
 }  
 };  
  
}