CS-223
CODE
DOCUMENT

for

Project 7
Classroom Visualisation App-1

Prepared by:
Group-12
Mitansh Jain - 160101042
Sujoy Ghosh - 160101073
Akul Agrawal - 160101085

April 22, 2018

Contents

1 Login Activity

~/Desktop/facedetect/LoginActivity1.java - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : LoginActivity
Date of Creation    : 10-04-2018
Author              : Sujoy Ghosh

Modification History : 11-04-2018 :- added validate()

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public void onCreate(Bundle)
                      public void login()
                      protected void onActivityResult(int, int, Intent)
                      public void onBackPressed()
                      public void onLoginSuccess()
                      public void onLoginFailed()
                      public boolean validate()

</header>
*/

package org.opencv.samples.facedetect;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import android.content.Intent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import butterknife.BindView;
import butterknife.ButterKnife;

public class LoginActivity1 extends AppCompatActivity {

    private static final String TAG = "LoginActivity";      // for debugging purpose
    private static final int REQUEST_SIGNUP = 0;            // stores the values corresponding to whether user asks for signup or not
    SQLiteDatabase sqLiteDatabaseObj;                       // used for database handling
    SQLiteHelper sqLiteHelper;                              // variable for accessing the users' database.
    Cursor cursor;                                          // stores the values returned by database.
    private String TempPassword;                            // used for storing the password as provided by database
```

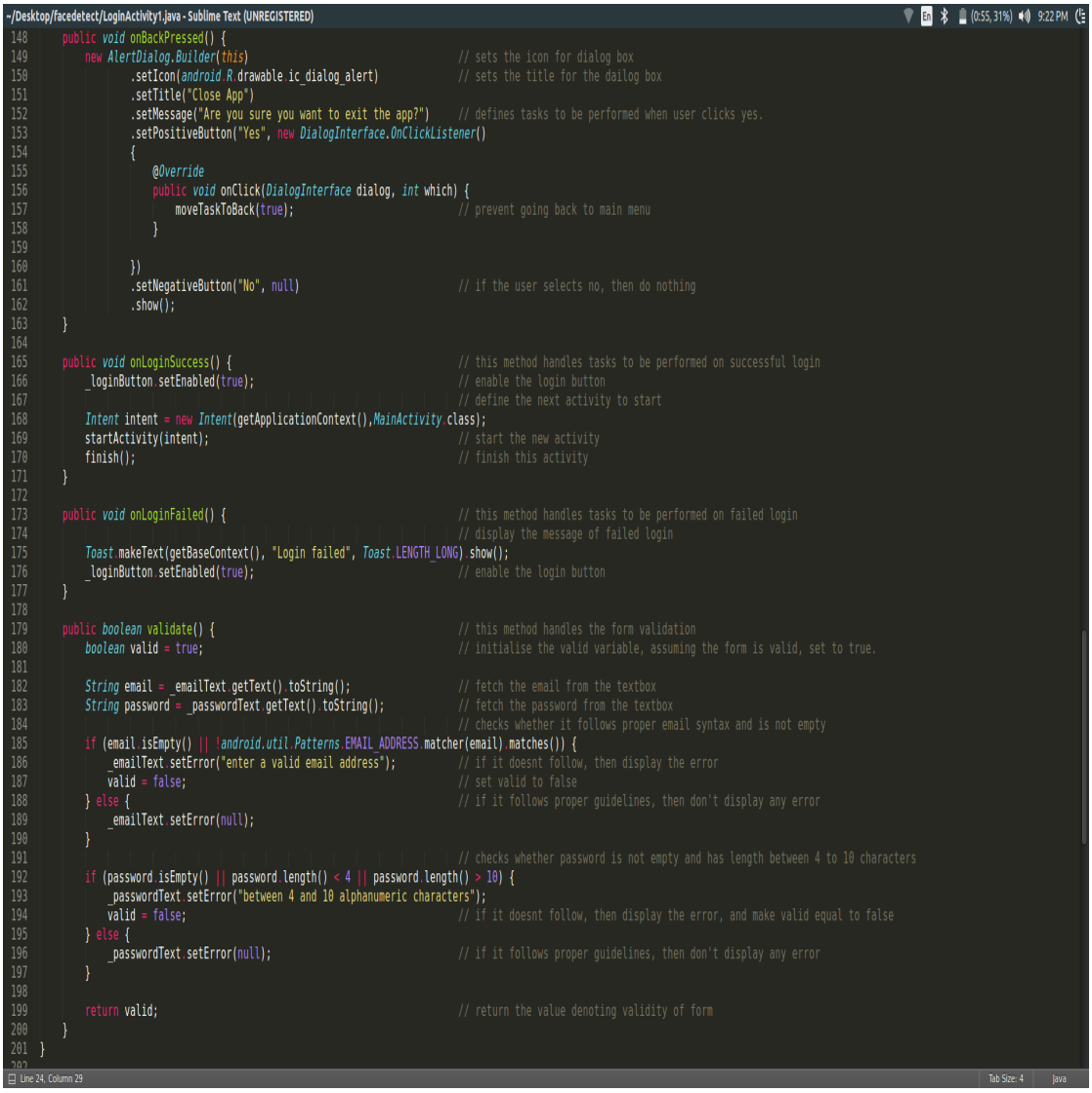
Line 24, Column 29 — Tab Size: 4 — Java

~/Desktop/facedetect/LoginActivity1.java - Sublime Text (UNREGISTERED)

```
52      SQLiteHelper sqLiteHelper;                          // variable for accessing the users' database.
53      Cursor cursor;                                      // stores the values returned by database.
54      private String TempPassword;                        // used for storing the password as provided by database
55
56      @BindView(R.id.input_email) EditText _emailText;        // binding the textbox with name _emailText
57      @BindView(R.id.input_password) EditText _passwordText;  // binding the textbox with name _passwordText
58      @BindView(R.id.btn_login) Button _loginButton;          // binding the button with name _loginButton
59      @BindView(R.id.link_signup) TextView _signupLink;       // binding the label with name _signupLink
60
61      @Override                                                   // this functions overrides the function defined in the parent class "AppCompatActivity".
62      public void onCreate(Bundle savedInstanceState) {           // this method is called when the activity is first created.
63          super.onCreate(savedInstanceState);                     // creates the activity.
64          setContentView(R.layout.activity_login);                // sets the content UI to be displayed on screen.
65          int PERMIT_ALL = 1;                                     // give permissions to all required permissions.
66                                                                  // contains all permissions
67          String[] PERMISSIONS = {Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE};
68          ActivityCompat.requestPermissions(this, PERMISSIONS, PERMIT_ALL); // asks for granting permissions if not done already.
69          ButterKnife.bind(this);                                 // handle all ui elements.
70          sqLiteHelper = new SQLiteHelper(this);                  // creates instance of users' database for accessing.
71          _loginButton.setOnClickListener(new View.OnClickListener() {    // this creates the constructor for onClick().
72              @Override                                           // this functions overrides the function defined in the parent class "AppCompatActivity".
73              public void onClick(View v) {                       // this function defines the tasks to be done when button is clicked.
74                  login();                                        // run the login method on clicking login button
75              }
76          });
77
78          _signupLink.setOnClickListener(new View.OnClickListener() {     // this creates the constructor for onClick().
79              @Override                                           // this functions overrides the function defined in the parent class "AppCompatActivity".
80              public void onClick(View v) {                       // this function defines the tasks to be done when button is clicked.
81                                                                  // Start the Signup activity
82                  Intent intent = new Intent(getApplicationContext(), SignupActivity.class);
83                  startActivityForResult(intent, REQUEST_SIGNUP);         // start the signup activity.
84                  finish();                                       // finish the present activity, clearing memory usage and decreasing ram usage
85                                                                  // defines the animation of moving from one activity to another.
86                  overridePendingTransition(R.anim.push_left_in, R.anim.push_left_out);
87              }
88          });
89      }
90
91      public void login() {                                       // this method handles login of the user
92          // Log.d(TAG, "Login");                                    //
93
94          if (!validate()) {                                      // if all fields are not valid entries, then display login failed message
95              onLoginFailed();                                    // display the login failed message.
96              return;
97          }
98
99          _loginButton.setEnabled(false);                         // if the form is valid, then disable the login button.
100                                                                  // start the progressbar
101         final ProgressDialog progressDialog = new ProgressDialog(LoginActivity1.this,
102                 R.style.AppTheme_Dark_Dialog);
103         progressDialog.setIndeterminate(true);                  // keep it running till the process completes
104         progressDialog.setMessage("Authenticating...");         // set the message to be displayed
105         progressDialog.show();                                  // display the progress bar
```

Line 24, Column 29 — Tab Size: 4 — Java


~/Desktop/facedetect/LoginActivity1.java - Sublime Text (UNREGISTERED)
9:22 PM
// start the progressbar
final ProgressDialog progressDialog = new ProgressDialog(LoginActivity1.this,
R.style.AppTheme_Dark_Dialog);
progressDialog.setIndeterminate(true); // keep it running till the process completes
progressDialog.setMessage("Authenticating..."); // set the message to be displayed
progressDialog.show(); // display the progress bar
String email = _emailText.getText().toString(); // obtain the email as enterd by user
final String password = _passwordText.getText().toString(); // fetch the password as entered by user
Cursor result = sqLiteHelper.getPassword(email); // get the password corresponding to given email id.
if(result.getCount() == 0) { // if there are no records with given email id
new android.os.Handler().postDelayed(// this defines what to be performed under this circumstance
new Runnable() { // define what to run
public void run() {
onLoginFailed(); // display that login has failed.
progressDialog.dismiss(); // close the progress bar
}
}, 3000);
return;
}
result.moveToFirst(); // else move to first row
TempPassword = result.getString(3); // extract the password from it.
new android.os.Handler().postDelayed(// this defines what to be performed under this circumstance
new Runnable() { // defines what to run on login success/failed
public void run() {
if(TempPassword.equals(password)) { // if extracted password equals entered passowrd, call onLoginSuccess()
onLoginSuccess();
} else { // if not, then call onLoginFailed()
onLoginFailed();
}
progressDialog.dismiss(); // close the progress dialog
}
}, 3000);
}
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (requestCode == REQUEST_SIGNUP) { // if signup is successful then finish the login activity and progress to main_activity
if (resultCode == RESULT_OK) {
this.finish();
}
}
}
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
public void onBackPressed() {
new AlertDialog.Builder(this) // sets the icon for dialog box
.setIcon(android.R.drawable.ic_dialog_alert) // sets the title for the dailog box
.setTitle("Close App")
.setMessage("Are you sure you want to exit the app?") // defines tasks to be performed when user clicks yes.
.setPositiveButton("Yes", new DialogInterface.OnClickListener()
{
Line 24, Column 29
Tab Size: 4
Java



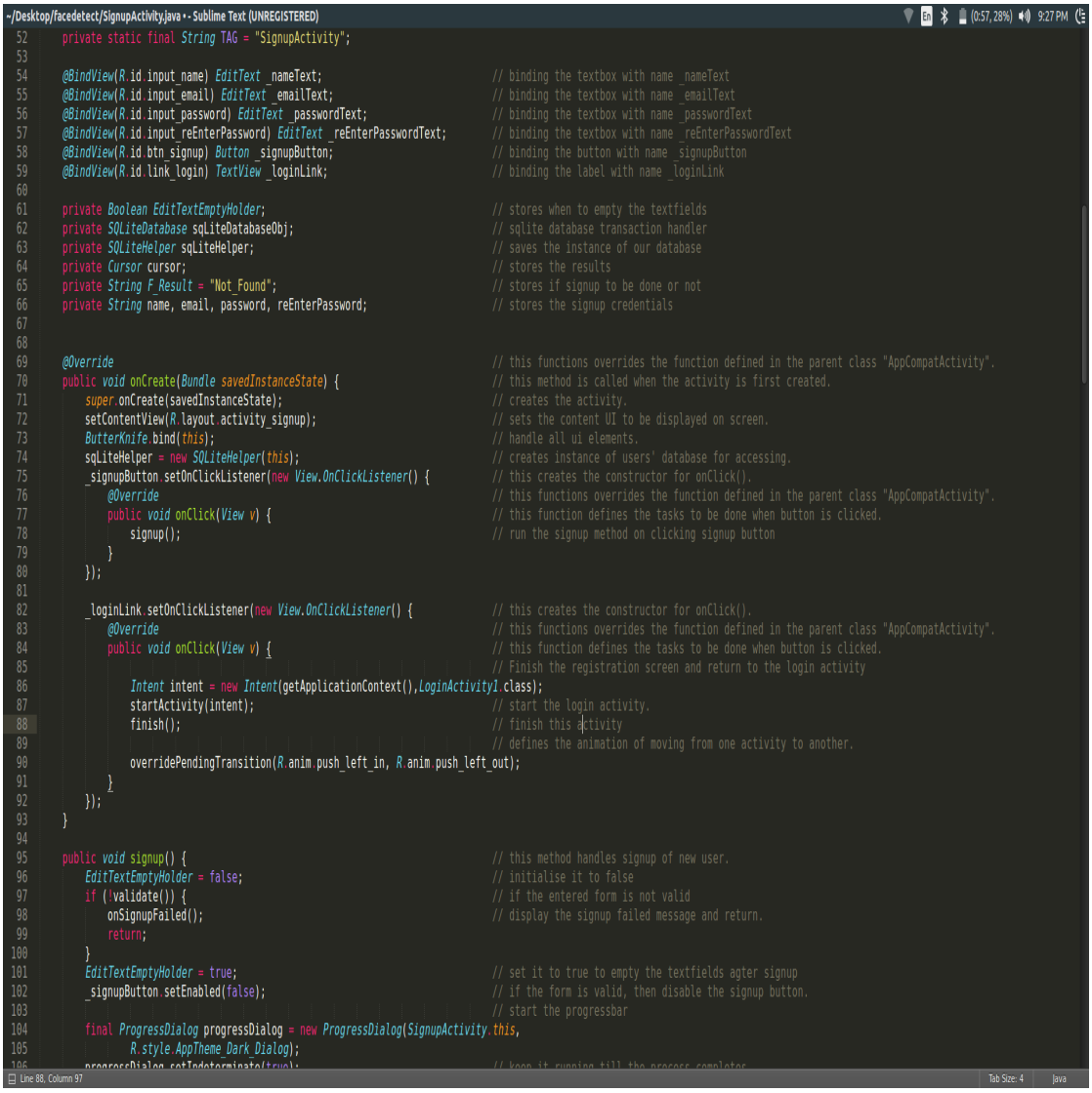

```
    public void onBackPressed() {
        new AlertDialog.Builder(this)                               // sets the icon for dialog box
                .setIcon(android.R.drawable.ic_dialog_alert)        // sets the title for the dailog box
                .setTitle("Close App")
                .setMessage("Are you sure you want to exit the app?")     // defines tasks to be performed when user clicks yes.
                .setPositiveButton("Yes", new DialogInterface.OnClickListener()
                {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        moveTaskToBack(true);                       // prevent going back to main menu
                    }

                })
                .setNegativeButton("No", null)                      // if the user selects no, then do nothing
                .show();
    }

    public void onLoginSuccess() {                                  // this method handles tasks to be performed on successful login
        _loginButton.setEnabled(true);                              // enable the login button
                                                                    // define the next activity to start
        Intent intent = new Intent(getApplicationContext(),MainActivity.class);
        startActivity(intent);                                      // start the new activity
        finish();                                                   // finish this activity
    }

    public void onLoginFailed() {                                   // this method handles tasks to be performed on failed login
                                                                    // display the message of failed login
        Toast.makeText(getBaseContext(), "Login failed", Toast.LENGTH_LONG).show();
        _loginButton.setEnabled(true);                              // enable the login button
    }

    public boolean validate() {                                     // this method handles the form validation
        boolean valid = true;                                       // initialise the valid variable, assuming the form is valid, set to true.

        String email = _emailText.getText().toString();             // fetch the email from the textbox
        String password = _passwordText.getText().toString();       // fetch the password from the textbox
                                                                    // checks whether it follows proper email syntax and is not empty
        if (email.isEmpty() || !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            _emailText.setError("enter a valid email address");     // if it doesnt follow, then display the error
            valid = false;                                          // set valid to false
        } else {                                                    // if it follows proper guidelines, then don't display any error
            _emailText.setError(null);
        }
                                                                    // checks whether password is not empty and has length between 4 to 10 characters
        if (password.isEmpty() || password.length() < 4 || password.length() > 10) {
            _passwordText.setError("between 4 and 10 alphanumeric characters");
            valid = false;                                          // if it doesnt follow, then display the error, and make valid equal to false
        } else {
            _passwordText.setError(null);                           // if it follows proper guidelines, then don't display any error
        }

        return valid;                                               // return the value denoting validity of form
    }
}
```

2 Signup Activity

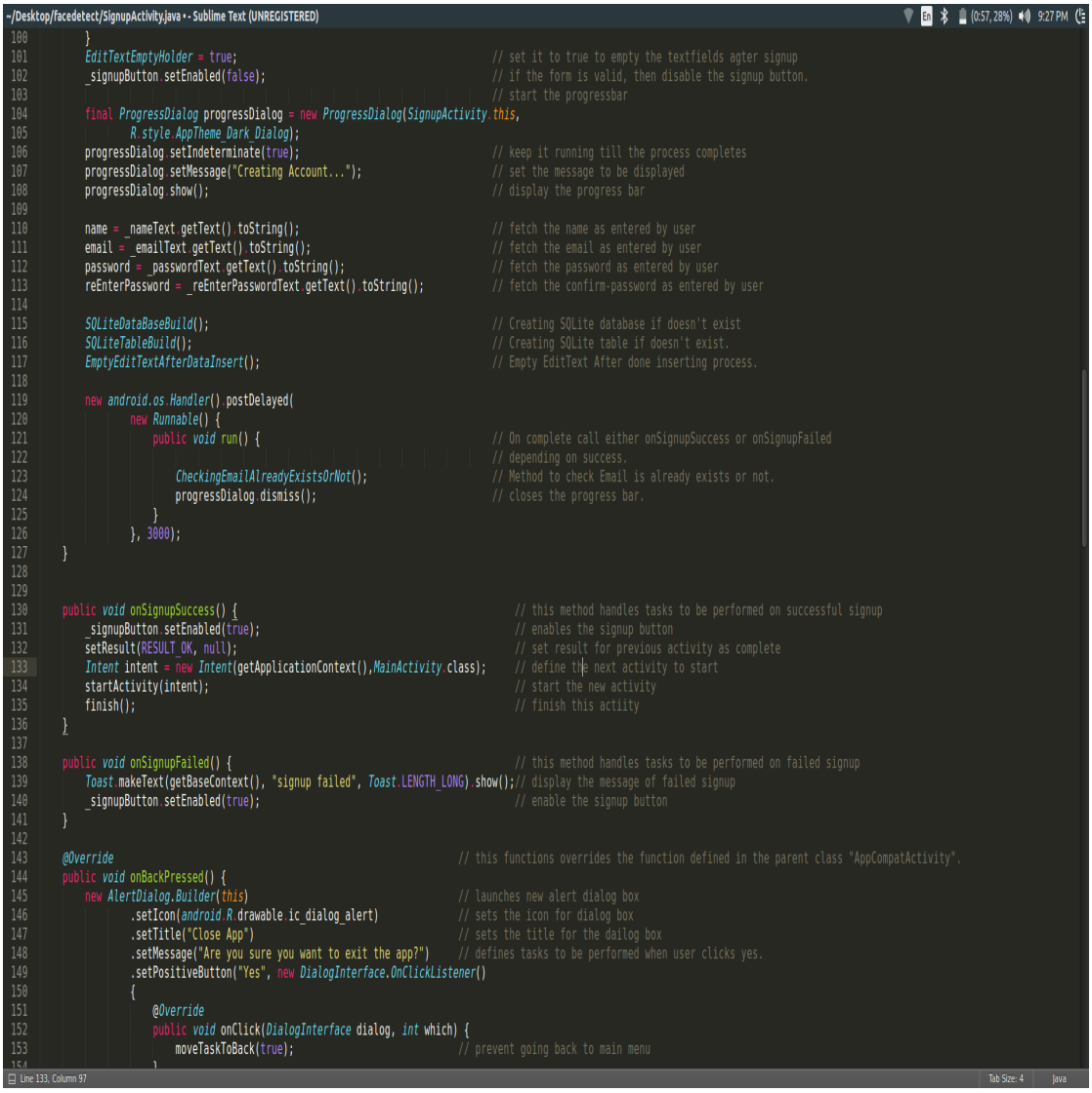
~/Desktop/facedetect/SignupActivity.java • - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : SignupActivity
Date of Creation    : 10-04-2018
Author              : Sujoy Ghosh

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public void onCreate(Bundle)
                      public void onClick(View)
                      public void signup()
                      public void onSignupSuccess()
                      public void onSignupFailed()
                      public void onBackPressed()
                      public void SQLiteDataBaseBuild()
                      public void SQLiteTableBuild()
                      public void EmptyEditTextAfterDataInsert()
                      public void CheckingEmailAlreadyExistsOrNot()
                      public void CheckFinalResult()
                      public boolean validate()

</header>
*/

package org.opencv.samples.facedetect;

//
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import butterknife.BindView;
import butterknife.ButterKnife;

public class SignupActivity extends AppCompatActivity {
    private static final String TAG = "SignupActivity";

    @BindView(R.id.input_name) EditText _nameText;                      // binding the textbox with name _nameText
    @BindView(R.id.input_email) EditText _emailText;                    // binding the textbox with name _emailText
```

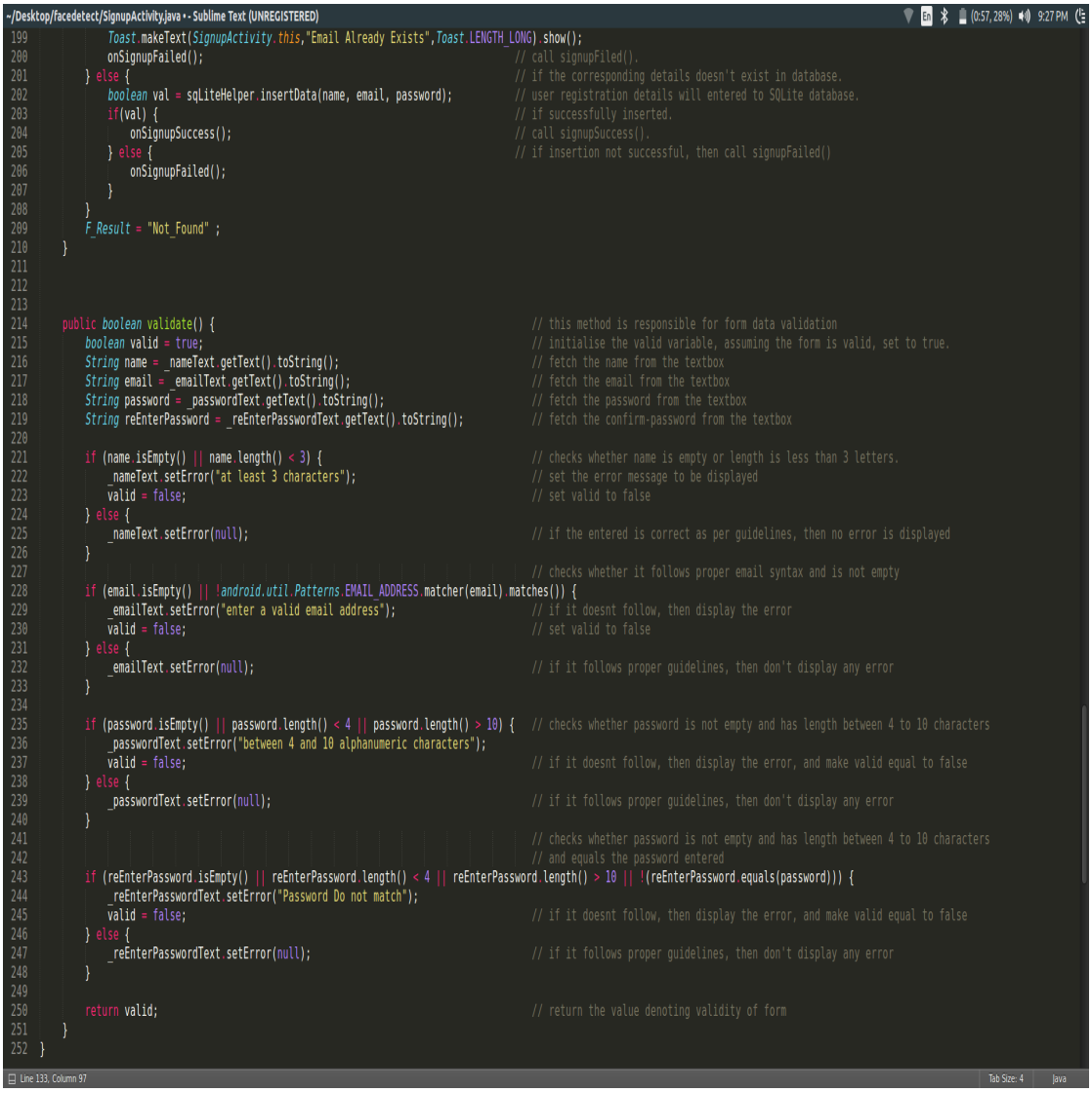
Line 172, Column 10 — Tab Size: 4 — Java




```
52      private static final String TAG = "SignupActivity";
53
54      @BindView(R.id.input_name) EditText _nameText;                          // binding the textbox with name _nameText
55      @BindView(R.id.input_email) EditText _emailText;                        // binding the textbox with name _emailText
56      @BindView(R.id.input_password) EditText _passwordText;                  // binding the textbox with name _passwordText
57      @BindView(R.id.input_reEnterPassword) EditText _reEnterPasswordText;    // binding the textbox with name _reEnterPasswordText
58      @BindView(R.id.btn_signup) Button _signupButton;                        // binding the button with name _signupButton
59      @BindView(R.id.link_login) TextView _loginLink;                         // binding the label with name _loginLink
60
61      private Boolean EditTextEmptyHolder;                                    // stores when to empty the textfields
62      private SQLiteDatabase sqLiteDatabaseObj;                               // sqlite database transaction handler
63      private SQLiteHelper sqLiteHelper;                                      // saves the instance of our database
64      private Cursor cursor;                                                  // stores the results
65      private String F_Result = "Not_Found";                                  // stores if signup to be done or not
66      private String name, email, password, reEnterPassword;                  // stores the signup credentials
67
68
69      @Override                                                               // this functions overrides the function defined in the parent class "AppCompatActivity".
70      public void onCreate(Bundle savedInstanceState) {                       // this method is called when the activity is first created.
71          super.onCreate(savedInstanceState);                                 // creates the activity.
72          setContentView(R.layout.activity_signup);                           // sets the content UI to be displayed on screen.
73          ButterKnife.bind(this);                                             // handle all ui elements.
74          sqLiteHelper = new SQLiteHelper(this);                              // creates instance of users' database for accessing.
75          _signupButton.setOnClickListener(new View.OnClickListener() {       // this creates the constructor for onClick().
76              @Override                                                       // this functions overrides the function defined in the parent class "AppCompatActivity".
77              public void onClick(View v) {                                   // this function defines the tasks to be done when button is clicked.
78                  signup();                                                   // run the signup method on clicking signup button
79              }
80          });
81
82          _loginLink.setOnClickListener(new View.OnClickListener() {          // this creates the constructor for onClick().
83              @Override                                                       // this functions overrides the function defined in the parent class "AppCompatActivity".
84              public void onClick(View v) {                                   // this function defines the tasks to be done when button is clicked.
85                                                                              // Finish the registration screen and return to the login activity
86                  Intent intent = new Intent(getApplicationContext(),LoginActivity1.class);
87                  startActivity(intent);                                      // start the login activity.
88                  finish();                                                   // finish this activity
89                                                                              // defines the animation of moving from one activity to another.
90                  overridePendingTransition(R.anim.push_left_in, R.anim.push_left_out);
91              }
92          });
93      }
94
95      public void signup() {                                                  // this method handles signup of new user.
96          EditTextEmptyHolder = false;                                        // initialise it to false
97          if (!validate()) {                                                  // if the entered form is not valid
98              onSignupFailed();                                               // display the signup failed message and return.
99              return;
100         }
101         EditTextEmptyHolder = true;                                         // set it to true to empty the textfields agter signup
102         _signupButton.setEnabled(false);                                    // if the form is valid, then disable the signup button.
103                                                                             // start the progressbar
104         final ProgressDialog progressDialog = new ProgressDialog(SignupActivity.this,
105                 R.style.AppTheme_Dark_Dialog);
106         progressDialog.setIndeterminate(true);                              // keep it running till the process completes
```


~/Desktop/facedetect/SignupActivity.java • - Sublime Text (UNREGISTERED)
(0:57, 28%)
9:27 PM
}
EditTextEmptyHolder = true;
// set it to true to empty the textfields agter signup
_signupButton.setEnabled(false);
// if the form is valid, then disable the signup button.
// start the progressbar
final ProgressDialog progressDialog = new ProgressDialog(SignupActivity.this,
R.style.AppTheme_Dark_Dialog);
progressDialog.setIndeterminate(true);
// keep it running till the process completes
progressDialog.setMessage("Creating Account...");
// set the message to be displayed
progressDialog.show();
// display the progress bar
name = _nameText.getText().toString();
// fetch the name as entered by user
email = _emailText.getText().toString();
// fetch the email as entered by user
password = _passwordText.getText().toString();
// fetch the password as entered by user
reEnterPassword = _reEnterPasswordText.getText().toString();
// fetch the confirm-password as entered by user
SQLiteDataBaseBuild();
// Creating SQLite database if doesn't exist
SQLiteTableBuild();
// Creating SQLite table if doesn't exist.
EmptyEditTextAfterDataInsert();
// Empty EditText After done inserting process.
new android.os.Handler().postDelayed(
new Runnable() {
public void run() {
// On complete call either onSignupSuccess or onSignupFailed
// depending on success.
CheckingEmailAlreadyExistsOrNot();
// Method to check Email is already exists or not.
progressDialog.dismiss();
// closes the progress bar.
}
}, 3000);
}
public void onSignupSuccess() {
// this method handles tasks to be performed on successful signup
_signupButton.setEnabled(true);
// enables the signup button
setResult(RESULT_OK, null);
// set result for previous activity as complete
Intent intent = new Intent(getApplicationContext(),MainActivity.class);
// define the next activity to start
startActivity(intent);
// start the new activity
finish();
// finish this actiity
}
public void onSignupFailed() {
// this method handles tasks to be performed on failed signup
Toast.makeText(getBaseContext(), "signup failed", Toast.LENGTH_LONG).show();// display the message of failed signup
_signupButton.setEnabled(true);
// enable the signup button
}
@Override
// this functions overrides the function defined in the parent class "AppCompatActivity".
public void onBackPressed() {
new AlertDialog.Builder(this)
// launches new alert dialog box
.setIcon(android.R.drawable.ic_dialog_alert)
// sets the icon for dialog box
.setTitle("Close App")
// sets the title for the dailog box
.setMessage("Are you sure you want to exit the app?")
// defines tasks to be performed when user clicks yes.
.setPositiveButton("Yes", new DialogInterface.OnClickListener()
{
@Override
public void onClick(DialogInterface dialog, int which) {
moveTaskToBack(true);
// prevent going back to main menu
Line 133, Column 97
Tab Size: 4
Java


~/Desktop/facedetect/SignupActivity.java • - Sublime Text (UNREGISTERED)
@Override
public void onClick(DialogInterface dialog, int which) {
moveTaskToBack(true); // prevent going back to main menu
}
})
.setNegativeButton("No", null) // if the user selects no, then do nothing
.show();
}
// SQLite database build method.
public void SQLiteDataBaseBuild(){
// creates the database if it doesn't exist already
sqLiteDatabaseObj = openOrCreateDatabase(SQLiteHelper.DATABASE_NAME, Context.MODE_PRIVATE, null);
}
// SQLite table build method.
public void SQLiteTableBuild() {
// creates the table if it doesn't exist
sqLiteDatabaseObj.execSQL("CREATE TABLE IF NOT EXISTS " + SQLiteHelper.TABLE_NAME + "(" + SQLiteHelper.Table_Column_ID +
" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, " + SQLiteHelper.Table_Column_1_Name + " VARCHAR, " + SQLiteHelper.Table_Column_2_Email +
" VARCHAR, " + SQLiteHelper.Table_Column_3_Password + " VARCHAR);");
}
// Empty edittext after done inserting process method.
public void EmptyEditTextAfterDataInsert(){
_nameText.getText().clear(); // clear the name textbox
_emailText.getText().clear(); // clear the email textbox
_passwordText.getText().clear(); // clear the password textbox
_reEnterPasswordText.getText().clear(); // clear the confirm-password textbox
}
public void CheckingEmailAlreadyExistsOrNot(){ // this method is responsible for checking if Email already exists or not.
boolean ifEmailExists = sqLiteHelper.ifEmailExists(email); // Adding search email query to cursor.
if(ifEmailExists){ // If Email is already exists then Result variable value set as Email Found.
F_Result = "Email Found";
}
CheckFinalResult(); // Calling method to check final result and insert data into SQLite database.
}
public void CheckFinalResult(){ // this method is responsible for check final result.
if(F_Result.equalsIgnoreCase("Email Found")) { // Checking whether email is already exists or not.
// If email is exists then disply the corresponding message.
Toast.makeText(SignupActivity.this,"Email Already Exists",Toast.LENGTH_LONG).show();
onSignupFailed(); // call signupFiled().
} else { // if the corresponding details doesn't exist in database.
boolean val = sqLiteHelper.insertData(name, email, password); // user registration details will entered to SQLite database.
if(val) { // if successfully inserted.
onSignupSuccess(); // call signupSuccess().
Line 133, Column 97
Tab Size: 4
Java



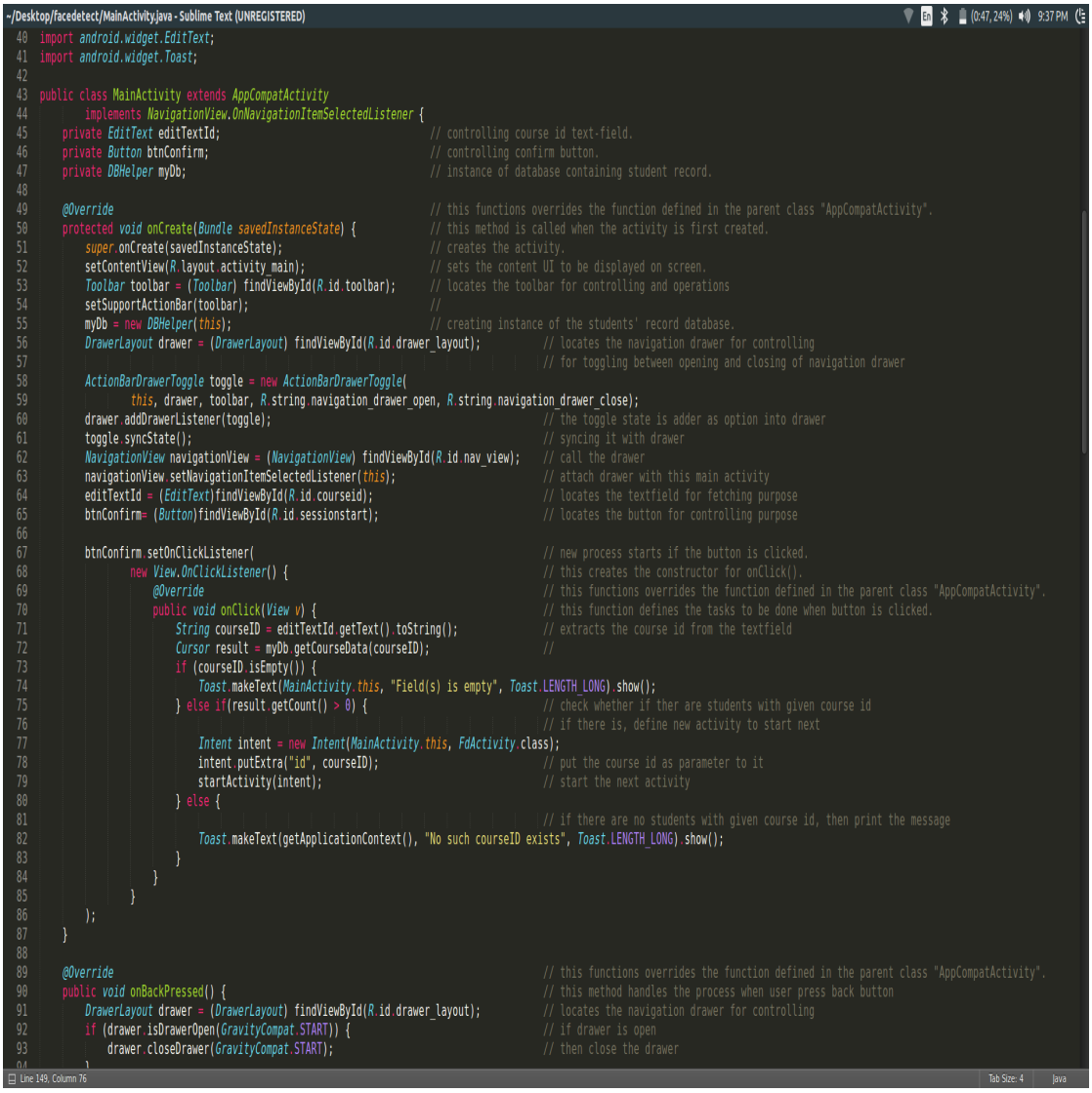

```
            Toast.makeText(SignupActivity.this,"Email Already Exists",Toast.LENGTH_LONG).show();
            onSignupFailed();                                       // call signupFiled().
        } else {                                                    // if the corresponding details doesn't exist in database.
            boolean val = sqLiteHelper.insertData(name, email, password);   // user registration details will entered to SQLite database.
            if(val) {                                               // if successfully inserted.
                onSignupSuccess();                                  // call signupSuccess().
            } else {                                                // if insertion not successful, then call signupFailed()
                onSignupFailed();
            }
        }
        F_Result = "Not_Found" ;
    }



    public boolean validate() {                                     // this method is responsible for form data validation
        boolean valid = true;                                       // initialise the valid variable, assuming the form is valid, set to true.
        String name = _nameText.getText().toString();               // fetch the name from the textbox
        String email = _emailText.getText().toString();             // fetch the email from the textbox
        String password = _passwordText.getText().toString();       // fetch the password from the textbox
        String reEnterPassword = _reEnterPasswordText.getText().toString();     // fetch the confirm-password from the textbox

        if (name.isEmpty() || name.length() < 3) {                  // checks whether name is empty or length is less than 3 letters.
            _nameText.setError("at least 3 characters");            // set the error message to be displayed
            valid = false;                                          // set valid to false
        } else {
            _nameText.setError(null);                               // if the entered is correct as per guidelines, then no error is displayed
        }
                                                                    // checks whether it follows proper email syntax and is not empty
        if (email.isEmpty() || !android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            _emailText.setError("enter a valid email address");     // if it doesnt follow, then display the error
            valid = false;                                          // set valid to false
        } else {
            _emailText.setError(null);                              // if it follows proper guidelines, then don't display any error
        }

        if (password.isEmpty() || password.length() < 4 || password.length() > 10) {   // checks whether password is not empty and has length between 4 to 10 characters
            _passwordText.setError("between 4 and 10 alphanumeric characters");
            valid = false;                                          // if it doesnt follow, then display the error, and make valid equal to false
        } else {
            _passwordText.setError(null);                           // if it follows proper guidelines, then don't display any error
        }
                                                                    // checks whether password is not empty and has length between 4 to 10 characters
                                                                    // and equals the password entered
        if (reEnterPassword.isEmpty() || reEnterPassword.length() < 4 || reEnterPassword.length() > 10 || !(reEnterPassword.equals(password))) {
            _reEnterPasswordText.setError("Password Do not match");
            valid = false;                                          // if it doesnt follow, then display the error, and make valid equal to false
        } else {
            _reEnterPasswordText.setError(null);                    // if it follows proper guidelines, then don't display any error
        }

        return valid;                                               // return the value denoting validity of form
    }
}
```

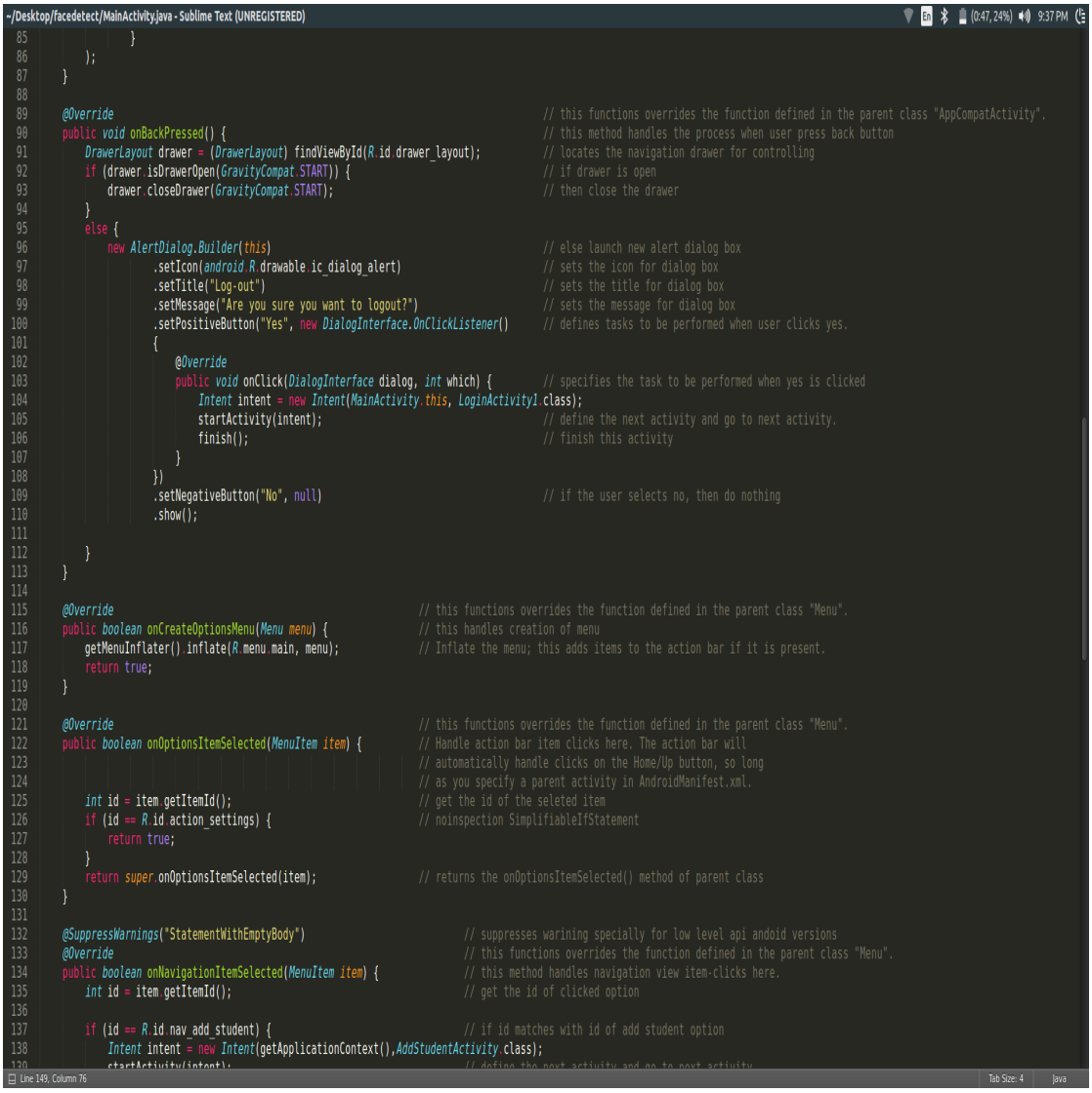
3 Main Activity

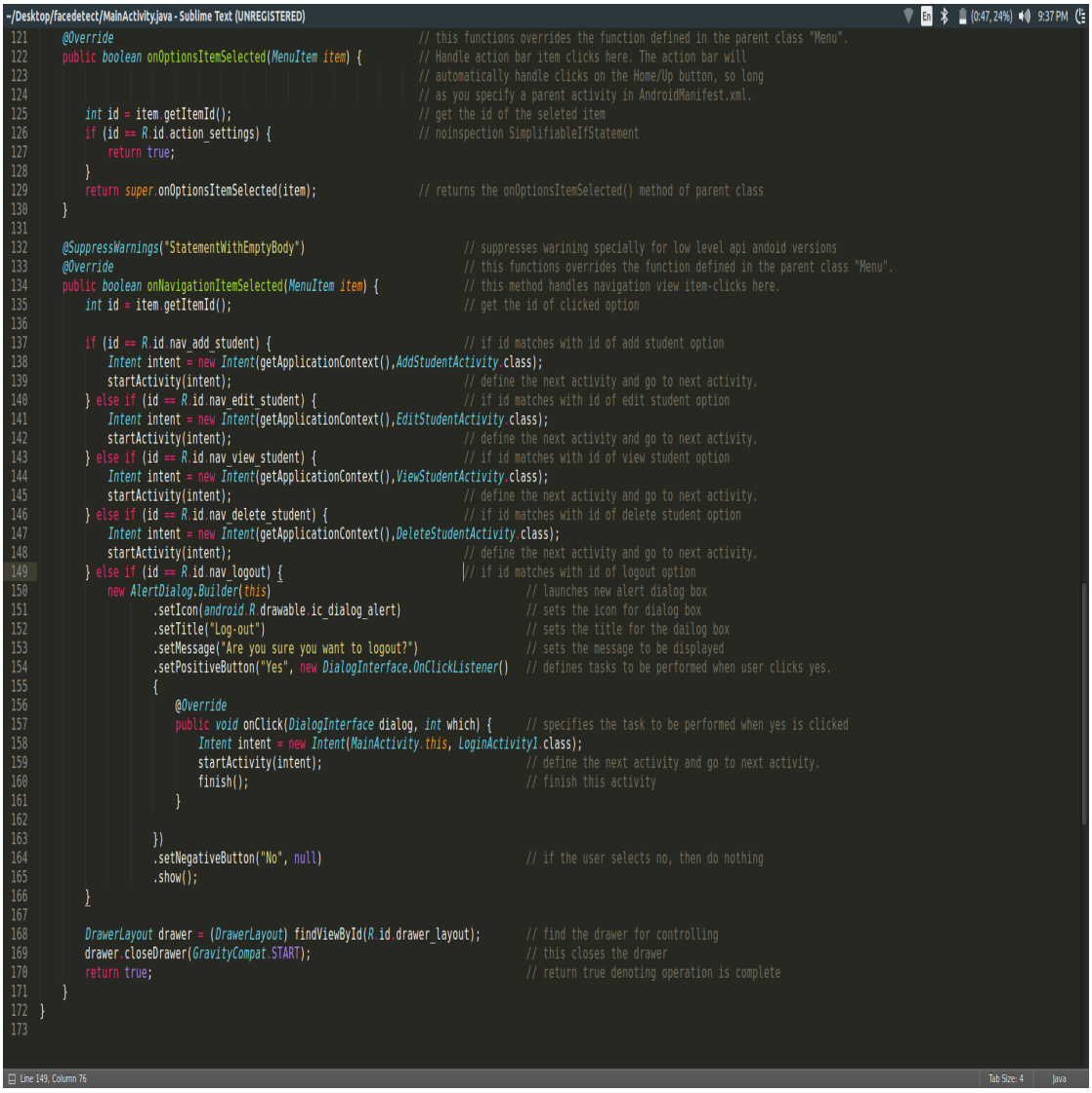
~/Desktop/facedetect/MainActivity.java - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : MainActivity
Date of Creation    : 10-04-2018
Author              : Akul Agrawal

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : protected void onCreate(Bundle)
                      public void onClick(View)
                      public void onBackPressed()
                      public boolean onCreateOptionsMenu(Menu)
                      public boolean onOptionsItemSelected(MenuItem)
                      public boolean onNavigationItemSelected(MenuItem)

</header>
*/
package org.opencv.samples.facedetect;

//
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
    private EditText editTextId;                                    // controlling course id text-field.
    private Button btnConfirm;                                      // controlling confirm button.
    private DBHelper myDb;                                          // instance of database containing student record.

    @Override                                                       // this functions overrides the function defined in the parent class "AppCompatActivity".
    protected void onCreate(Bundle savedInstanceState) {            // this method is called when the activity is first created.
        super.onCreate(savedInstanceState);                         // creates the activity.
        setContentView(R.layout.activity_main);                     // sets the content UI to be displayed on screen.
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);     // locates the toolbar for controlling and operations
        setSupportActionBar(toolbar);                               //
        myDb = new DBHelper(this);                                  // creating instance of the students' record database
```

Line 149, Column 76 — Tab Size: 4 — Java


~/Desktop/facedetect/MainActivity.java - Sublime Text (UNREGISTERED)
(0:47, 24%)
9:37 PM
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity
implements NavigationView.OnNavigationItemSelectedListener {
private EditText editTextId; // controlling course id text-field.
private Button btnConfirm; // controlling confirm button.
private DBHelper myDb; // instance of database containing student record.
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
super.onCreate(savedInstanceState); // creates the activity.
setContentView(R.layout.activity_main); // sets the content UI to be displayed on screen.
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); // locates the toolbar for controlling and operations
setSupportActionBar(toolbar); //
myDb = new DBHelper(this); // creating instance of the students' record database.
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // locates the navigation drawer for controlling
// for toggling between opening and closing of navigation drawer
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle); // the toggle state is adder as option into drawer
toggle.syncState(); // syncing it with drawer
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view); // call the drawer
navigationView.setNavigationItemSelectedListener(this); // attach drawer with this main activity
editTextId = (EditText)findViewById(R.id.courseid); // locates the textfield for fetching purpose
btnConfirm= (Button)findViewById(R.id.sessionstart); // locates the button for controlling purpose
btnConfirm.setOnClickListener(// new process starts if the button is clicked.
new View.OnClickListener() { // this creates the constructor for onClick().
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
public void onClick(View v) { // this function defines the tasks to be done when button is clicked.
String courseID = editTextId.getText().toString(); // extracts the course id from the textfield
Cursor result = myDb.getCourseData(courseID); //
if (courseID.isEmpty()) {
Toast.makeText(MainActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
} else if(result.getCount() > 0) { // check whether if ther are students with given course id
// if there is, define new activity to start next
Intent intent = new Intent(MainActivity.this, FdActivity.class);
intent.putExtra("id", courseID); // put the course id as parameter to it
startActivity(intent); // start the next activity
} else {
// if there are no students with given course id, then print the message
Toast.makeText(getApplicationContext(), "No such courseID exists", Toast.LENGTH_LONG).show();
}
}
}
);
}
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
public void onBackPressed() { // this method handles the process when user press back button
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // locates the navigation drawer for controlling
if (drawer.isDrawerOpen(GravityCompat.START)) { // if drawer is open
drawer.closeDrawer(GravityCompat.START); // then close the drawer
Line 149, Column 76
Tab Size: 4
Java


~/Desktop/facedetect/MainActivity.java - Sublime Text (UNREGISTERED)
(0:47, 24%)
9:37 PM
}
);
}
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
public void onBackPressed() { // this method handles the process when user press back button
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // locates the navigation drawer for controlling
if (drawer.isDrawerOpen(GravityCompat.START)) { // if drawer is open
drawer.closeDrawer(GravityCompat.START); // then close the drawer
}
else {
new AlertDialog.Builder(this) // else launch new alert dialog box
.setIcon(android.R.drawable.ic_dialog_alert) // sets the icon for dialog box
.setTitle("Log-out") // sets the title for dialog box
.setMessage("Are you sure you want to logout?") // sets the message for dialog box
.setPositiveButton("Yes", new DialogInterface.OnClickListener() // defines tasks to be performed when user clicks yes.
{
@Override
public void onClick(DialogInterface dialog, int which) { // specifies the task to be performed when yes is clicked
Intent intent = new Intent(MainActivity.this, LoginActivity1.class);
startActivity(intent); // define the next activity and go to next activity.
finish(); // finish this activity
}
})
.setNegativeButton("No", null) // if the user selects no, then do nothing
.show();
}
}
@Override // this functions overrides the function defined in the parent class "Menu".
public boolean onCreateOptionsMenu(Menu menu) { // this handles creation of menu
getMenuInflater().inflate(R.menu.main, menu); // Inflate the menu; this adds items to the action bar if it is present.
return true;
}
@Override // this functions overrides the function defined in the parent class "Menu".
public boolean onOptionsItemSelected(MenuItem item) { // Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId(); // get the id of the seleted item
if (id == R.id.action_settings) { // noinspection SimplifiableIfStatement
return true;
}
return super.onOptionsItemSelected(item); // returns the onOptionsItemSelected() method of parent class
}
@SuppressWarnings("StatementWithEmptyBody") // suppresses warining specially for low level api andoid versions
@Override // this functions overrides the function defined in the parent class "Menu".
public boolean onNavigationItemSelected(MenuItem item) { // this method handles navigation view item-clicks here.
int id = item.getItemId(); // get the id of clicked option
if (id == R.id.nav_add_student) { // if id matches with id of add student option
Intent intent = new Intent(getApplicationContext(),AddStudentActivity.class);
Line 149, Column 76
Tab Size: 4
Java


~/Desktop/facedetect/MainActivity.java - Sublime Text (UNREGISTERED)
(0:47, 24%)
9:37 PM
@Override // this functions overrides the function defined in the parent class "Menu".
public boolean onOptionsItemSelected(MenuItem item) { // Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId(); // get the id of the seleted item
if (id == R.id.action_settings) { // noinspection SimplifiableIfStatement
return true;
}
return super.onOptionsItemSelected(item); // returns the onOptionsItemSelected() method of parent class
}
@SuppressWarnings("StatementWithEmptyBody") // suppresses warining specially for low level api andoid versions
@Override // this functions overrides the function defined in the parent class "Menu".
public boolean onNavigationItemSelected(MenuItem item) { // this method handles navigation view item-clicks here.
int id = item.getItemId(); // get the id of clicked option
if (id == R.id.nav_add_student) { // if id matches with id of add student option
Intent intent = new Intent(getApplicationContext(),AddStudentActivity.class);
startActivity(intent); // define the next activity and go to next activity.
} else if (id == R.id.nav_edit_student) { // if id matches with id of edit student option
Intent intent = new Intent(getApplicationContext(),EditStudentActivity.class);
startActivity(intent); // define the next activity and go to next activity.
} else if (id == R.id.nav_view_student) { // if id matches with id of view student option
Intent intent = new Intent(getApplicationContext(),ViewStudentActivity.class);
startActivity(intent); // define the next activity and go to next activity.
} else if (id == R.id.nav_delete_student) { // if id matches with id of delete student option
Intent intent = new Intent(getApplicationContext(),DeleteStudentActivity.class);
startActivity(intent); // define the next activity and go to next activity.
} else if (id == R.id.nav_logout) { // if id matches with id of logout option
new AlertDialog.Builder(this) // launches new alert dialog box
.setIcon(android.R.drawable.ic_dialog_alert) // sets the icon for dialog box
.setTitle("Log-out") // sets the title for the dailog box
.setMessage("Are you sure you want to logout?") // sets the message to be displayed
.setPositiveButton("Yes", new DialogInterface.OnClickListener() // defines tasks to be performed when user clicks yes.
{
@Override
public void onClick(DialogInterface dialog, int which) { // specifies the task to be performed when yes is clicked
Intent intent = new Intent(MainActivity.this, LoginActivity1.class);
startActivity(intent); // define the next activity and go to next activity.
finish(); // finish this activity
}
})
.setNegativeButton("No", null) // if the user selects no, then do nothing
.show();
}
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout); // find the drawer for controlling
drawer.closeDrawer(GravityCompat.START); // this closes the drawer
return true; // return true denoting operation is complete
}
}
Line 149, Column 76
Tab Size: 4
Java

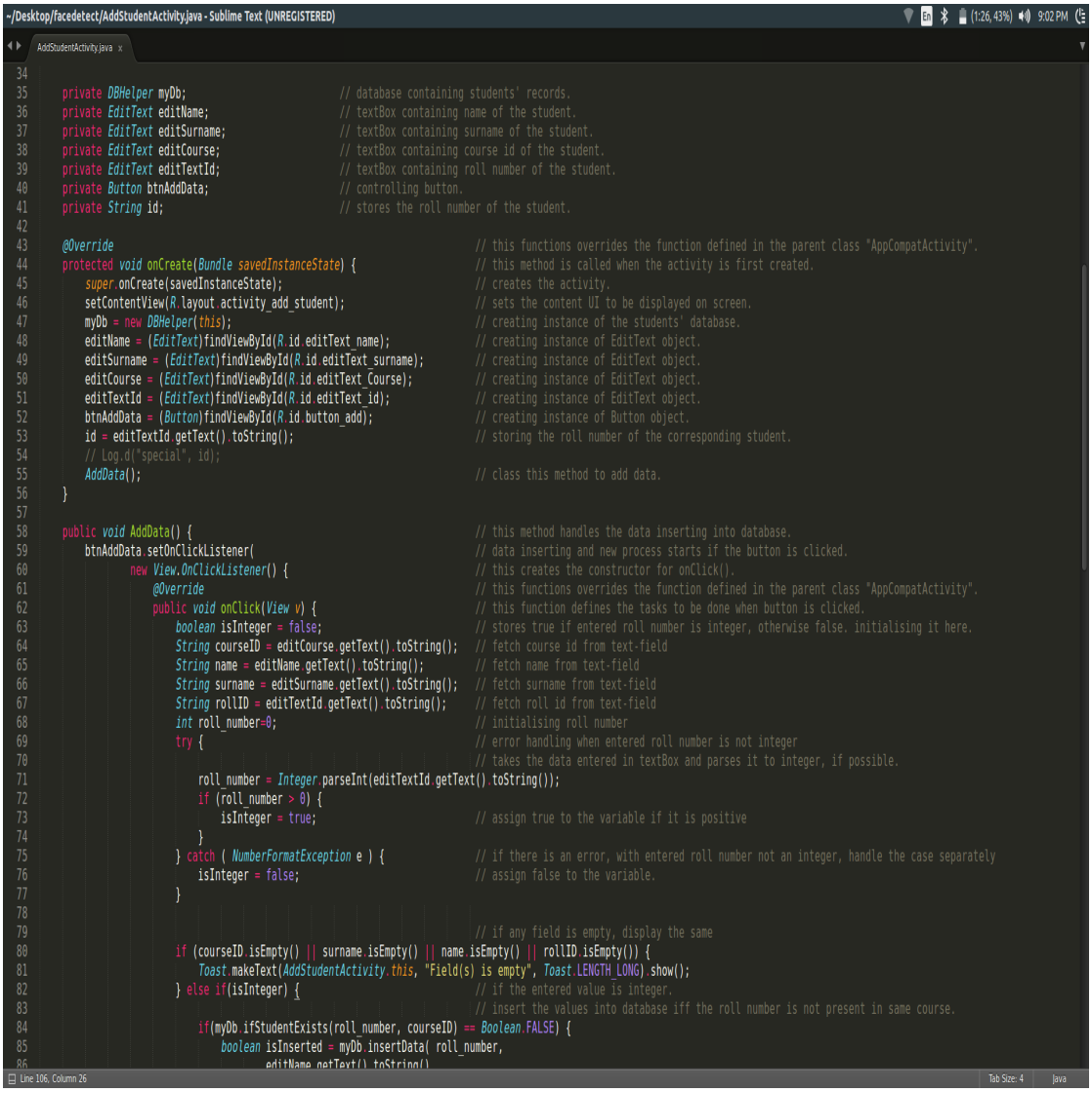
4 Add Student

File Edit Selection Find View Goto Tools Project Preferences Help

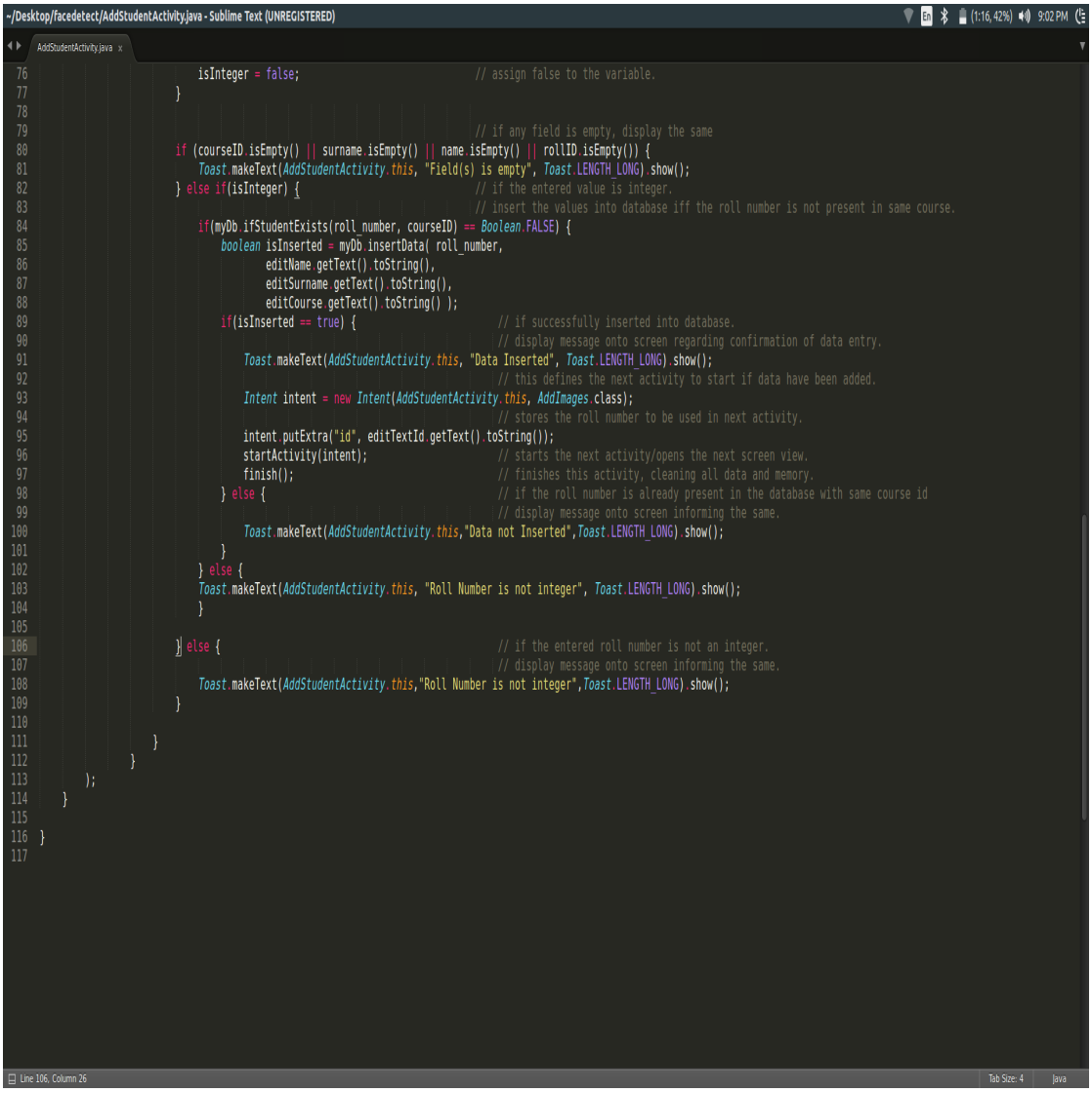
AddStudentActivity.java

```java
/*
<header>
Module Name          : AddStudentActivity
Date of Creation     : 10-04-2018
Author               : Akul Agrawal

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis             : This module is responsible for adding new student record.

Global Variables     : None

Functions            : void AddData()
                       void onCreate(Bundle)
</header>
*/

package org.opencv.samples.facedetect;

// importing packages for android
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;
import android.util.Log;

public class AddStudentActivity extends AppCompatActivity {

    private DBHelper myDb;                      // database containing students' records.
    private EditText editName;                  // textBox containing name of the student.
    private EditText editSurname;               // textBox containing surname of the student.
    private EditText editCourse;                // textBox containing course id of the student.
    private EditText editTextId;                // textBox containing roll number of the student.
    private Button btnAddData;                  // controlling button.
    private String id;                          // stores the roll number of the student.

    @Override                                                   // this functions overrides the function defined in the parent class "AppCompatActivity".
    protected void onCreate(Bundle savedInstanceState) {        // this method is called when the activity is first created.
        super.onCreate(savedInstanceState);                     // creates the activity.
        setContentView(R.layout.activity_add_student);          // sets the content UI to be displayed on screen.
        myDb = new DBHelper(this);                              // creating instance of the students' database.
        editName = (EditText)findViewById(R.id.editText_name);  // creating instance of EditText object.
        editSurname = (EditText)findViewById(R.id.editText_surname);  // creating instance of EditText object.
        editCourse = (EditText)findViewById(R.id.editText_Course);    // creating instance of EditText object.
        editTextId = (EditText)findViewById(R.id.editText_id);  // creating instance of EditText object.
        btnAddData = (Button)findViewById(R.id.button_add);     // creating instance of Button object.
        id = editTextId.getText().toString();                   // storing the roll number of the corresponding student
```

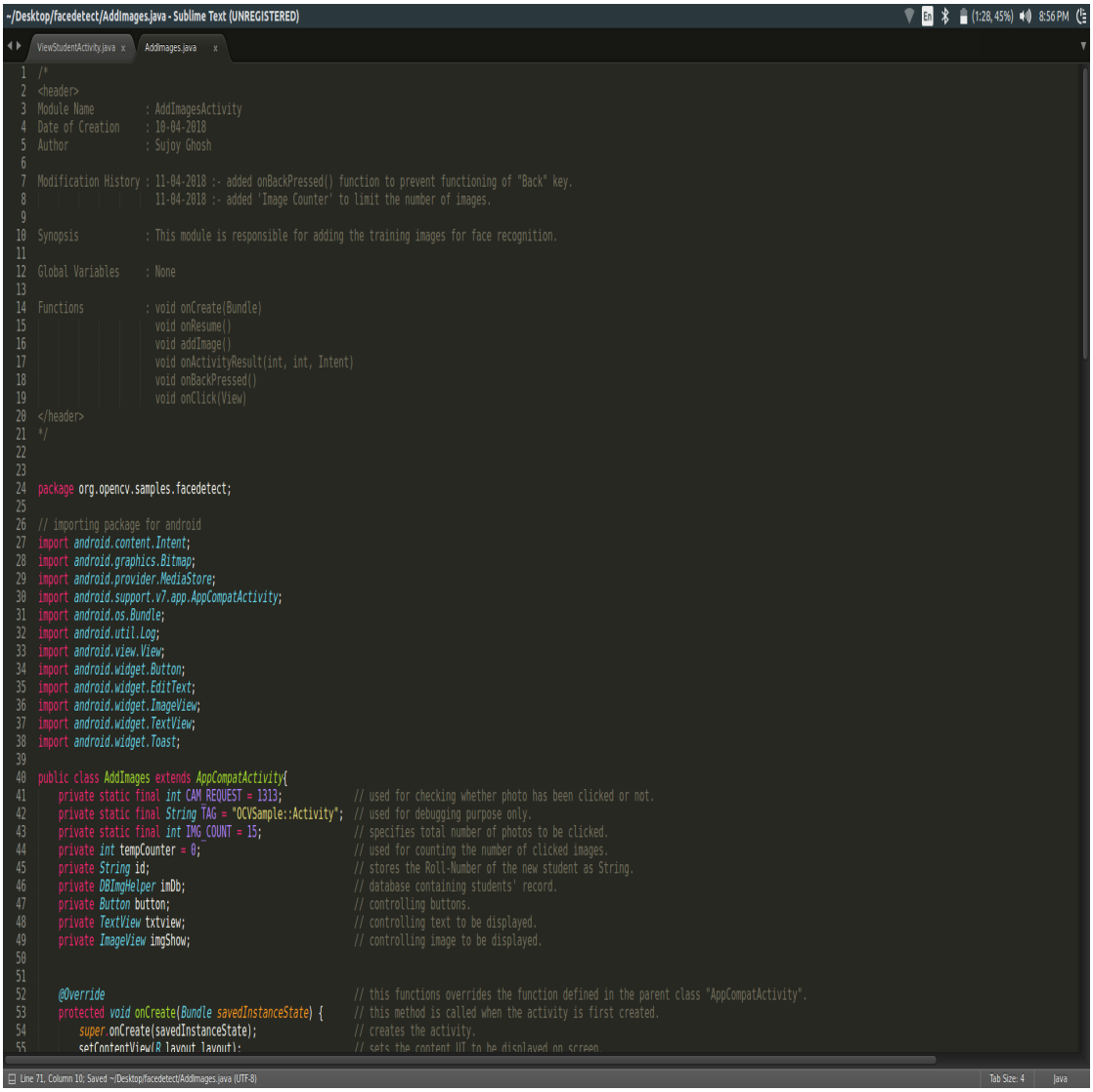
Line 106, Column 26 Tab Size: 4 Java

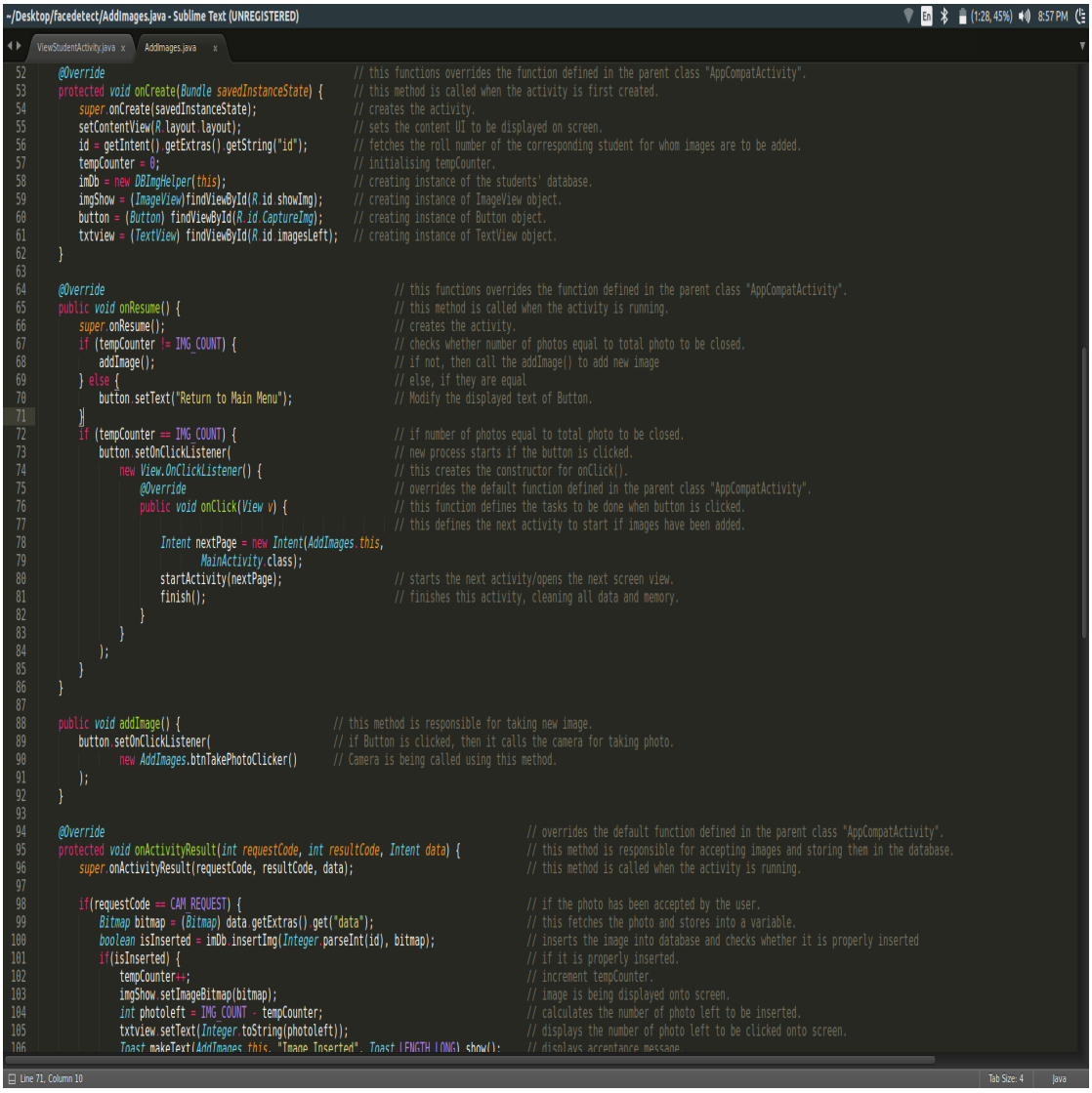


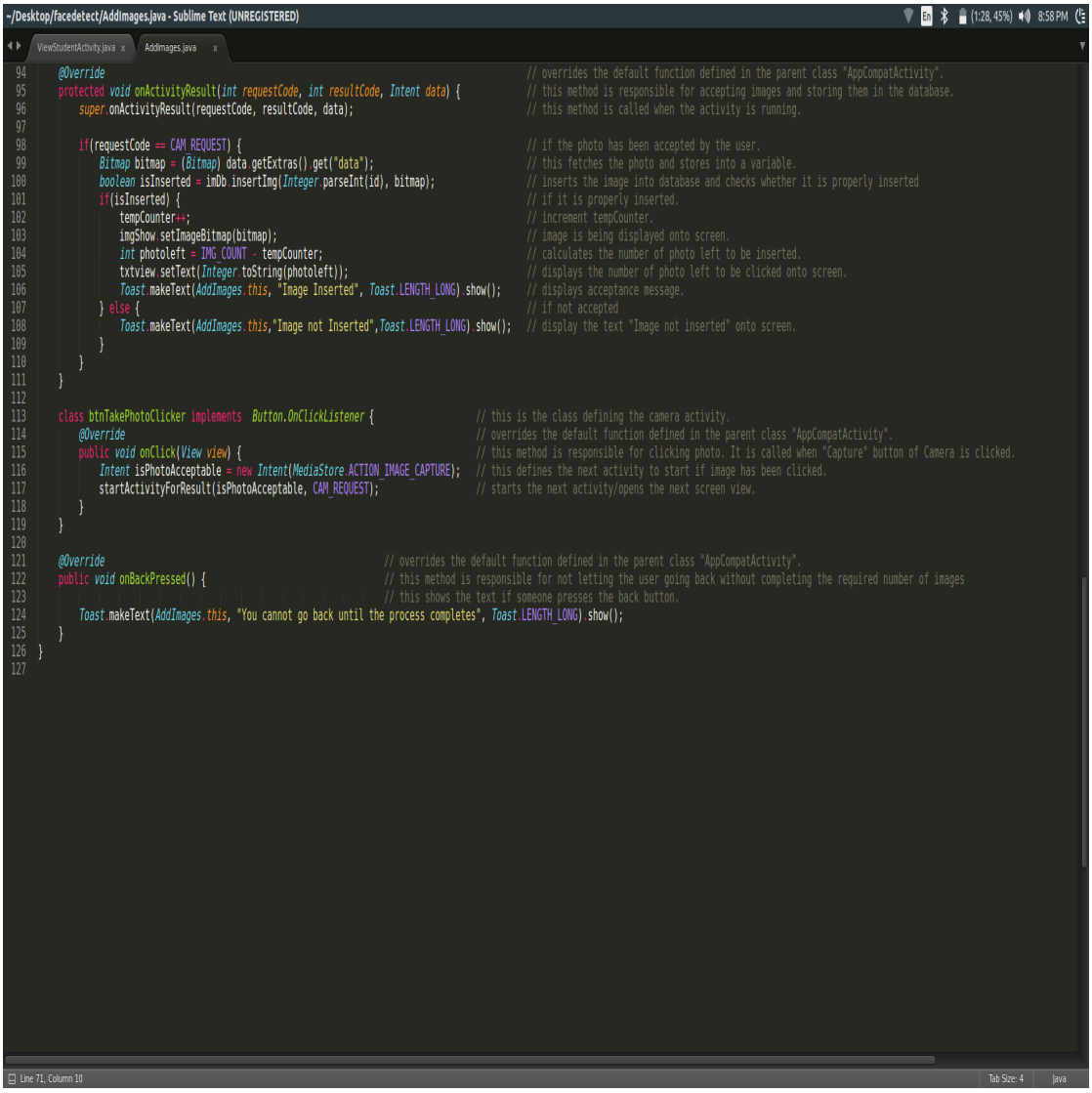

```
34
35      private DBHelper myDb;                          // database containing students' records.
36      private EditText editName;                      // textBox containing name of the student.
37      private EditText editSurname;                   // textBox containing surname of the student.
38      private EditText editCourse;                    // textBox containing course id of the student.
39      private EditText editTextId;                    // textBox containing roll number of the student.
40      private Button btnAddData;                      // controlling button.
41      private String id;                              // stores the roll number of the student.
42
43      @Override                                                   // this functions overrides the function defined in the parent class "AppCompatActivity".
44      protected void onCreate(Bundle savedInstanceState) {        // this method is called when the activity is first created.
45          super.onCreate(savedInstanceState);                     // creates the activity.
46          setContentView(R.layout.activity_add_student);          // sets the content UI to be displayed on screen.
47          myDb = new DBHelper(this);                              // creating instance of the students' database.
48          editName = (EditText)findViewById(R.id.editText_name);  // creating instance of EditText object.
49          editSurname = (EditText)findViewById(R.id.editText_surname);    // creating instance of EditText object.
50          editCourse = (EditText)findViewById(R.id.editText_Course);      // creating instance of EditText object.
51          editTextId = (EditText)findViewById(R.id.editText_id);          // creating instance of EditText object.
52          btnAddData = (Button)findViewById(R.id.button_add);             // creating instance of Button object.
53          id = editTextId.getText().toString();                   // storing the roll number of the corresponding student.
54          // Log.d("special", id);
55          AddData();                                              // class this method to add data.
56      }
57
58      public void AddData() {                                     // this method handles the data inserting into database.
59          btnAddData.setOnClickListener(                          // data inserting and new process starts if the button is clicked.
60                  new View.OnClickListener() {                    // this creates the constructor for onClick().
61                      @Override                                   // this functions overrides the function defined in the parent class "AppCompatActivity".
62                      public void onClick(View v) {               // this function defines the tasks to be done when button is clicked.
63                          boolean isInteger = false;              // stores true if entered roll number is integer, otherwise false. initialising it here.
64                          String courseID = editCourse.getText().toString();  // fetch course id from text-field
65                          String name = editName.getText().toString();        // fetch name from text-field
66                          String surname = editSurname.getText().toString();  // fetch surname from text-field
67                          String rollID = editTextId.getText().toString();    // fetch roll id from text-field
68                          int roll_number=0;                      // initialising roll number
69                          try {                                   // error handling when entered roll number is not integer
70                                                                  // takes the data entered in textBox and parses it to integer, if possible.
71                              roll_number = Integer.parseInt(editTextId.getText().toString());
72                              if (roll_number > 0) {
73                                  isInteger = true;               // assign true to the variable if it is positive
74                              }
75                          } catch ( NumberFormatException e ) {   // if there is an error, with entered roll number not an integer, handle the case separately
76                              isInteger = false;                  // assign false to the variable.
77                          }
78
79                                                                  // if any field is empty, display the same
80                          if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
81                              Toast.makeText(AddStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
82                          } else if(isInteger) {                  // if the entered value is integer.
83                                                                  // insert the values into database iff the roll number is not present in same course.
84                              if(myDb.ifStudentExists(roll_number, courseID) == Boolean.FALSE) {
85                                  boolean isInserted = myDb.insertData( roll_number,
86                                          editName.getText().toString()
```


~/Desktop/facedetect/AddStudentActivity.java - Sublime Text (UNREGISTERED)
AddStudentActivity.java
isInteger = false; // assign false to the variable.
}
// if any field is empty, display the same
if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
Toast.makeText(AddStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
} else if(isInteger) { // if the entered value is integer.
// insert the values into database iff the roll number is not present in same course.
if(myDb.ifStudentExists(roll_number, courseID) == Boolean.FALSE) {
boolean isInserted = myDb.insertData(roll_number,
editName.getText().toString(),
editSurname.getText().toString(),
editCourse.getText().toString());
if(isInserted == true) { // if successfully inserted into database.
// display message onto screen regarding confirmation of data entry.
Toast.makeText(AddStudentActivity.this, "Data Inserted", Toast.LENGTH_LONG).show();
// this defines the next activity to start if data have been added.
Intent intent = new Intent(AddStudentActivity.this, AddImages.class);
// stores the roll number to be used in next activity.
intent.putExtra("id", editTextId.getText().toString());
startActivity(intent); // starts the next activity/opens the next screen view.
finish(); // finishes this activity, cleaning all data and memory.
} else { // if the roll number is already present in the database with same course id
// display message onto screen informing the same.
Toast.makeText(AddStudentActivity.this,"Data not Inserted",Toast.LENGTH_LONG).show();
}
} else {
Toast.makeText(AddStudentActivity.this, "Roll Number is not integer", Toast.LENGTH_LONG).show();
}
} else { // if the entered roll number is not an integer.
// display message onto screen informing the same.
Toast.makeText(AddStudentActivity.this,"Roll Number is not integer",Toast.LENGTH_LONG).show();
}
}
}
);
}
}
Line 106, Column 26
Tab Size: 4
Java

5 Add Images


~/Desktop/facedetect/AddImages.java - Sublime Text (UNREGISTERED)
ViewStudentActivity.java
AddImages.java
/*
<header>
Module Name : AddImagesActivity
Date of Creation : 10-04-2018
Author : Sujoy Ghosh
Modification History : 11-04-2018 :- added onBackPressed() function to prevent functioning of "Back" key.
11-04-2018 :- added 'Image Counter' to limit the number of images.
Synopsis : This module is responsible for adding the training images for face recognition.
Global Variables : None
Functions : void onCreate(Bundle)
void onResume()
void addImage()
void onActivityResult(int, int, Intent)
void onBackPressed()
void onClick(View)
</header>
*/
package org.opencv.samples.facedetect;
// importing package for android
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
public class AddImages extends AppCompatActivity{
private static final int CAM_REQUEST = 1313; // used for checking whether photo has been clicked or not.
private static final String TAG = "OCVSample::Activity"; // used for debugging purpose only.
private static final int IMG_COUNT = 15; // specifies total number of photos to be clicked.
private int tempCounter = 0; // used for counting the number of clicked images.
private String id; // stores the Roll-Number of the new student as String.
private DBImgHelper imDb; // database containing students' record.
private Button button; // controlling buttons.
private TextView txtview; // controlling text to be displayed.
private ImageView imgShow; // controlling image to be displayed.
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
super.onCreate(savedInstanceState); // creates the activity.
Line 71, Column 10; Saved ~/Desktop/facedetect/AddImages.java (UTF-8)
Tab Size: 4
Java


~/Desktop/facedetect/AddImages.java - Sublime Text (UNREGISTERED)
ViewStudentActivity.java
AddImages.java
Line 71, Column 10
Tab Size: 4
Java

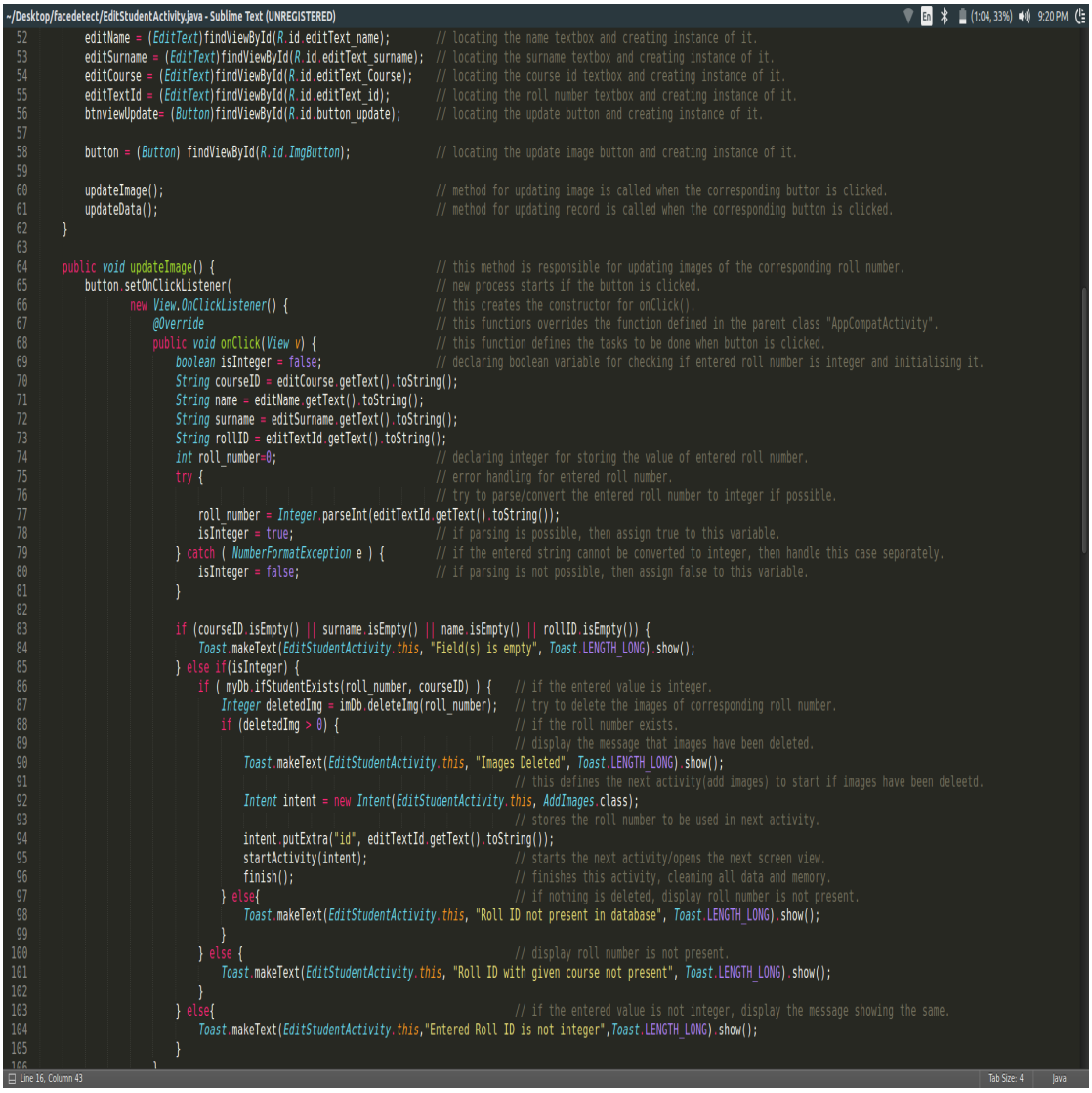

~/Desktop/facedetect/AddImages.java - Sublime Text (UNREGISTERED)
ViewStudentActivity.java
AddImages.java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if(requestCode == CAM_REQUEST) {
Bitmap bitmap = (Bitmap) data.getExtras().get("data");
boolean isInserted = imDb.insertImg(Integer.parseInt(id), bitmap);
if(isInserted) {
tempCounter++;
imgShow.setImageBitmap(bitmap);
int photoleft = IMG_COUNT - tempCounter;
txtview.setText(Integer.toString(photoleft));
Toast.makeText(AddImages.this, "Image Inserted", Toast.LENGTH_LONG).show();
} else {
Toast.makeText(AddImages.this,"Image not Inserted",Toast.LENGTH_LONG).show();
}
}
}
class btnTakePhotoClicker implements Button.OnClickListener {
@Override
public void onClick(View view) {
Intent isPhotoAcceptable = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(isPhotoAcceptable, CAM_REQUEST);
}
}
@Override
public void onBackPressed() {
Toast.makeText(AddImages.this, "You cannot go back until the process completes", Toast.LENGTH_LONG).show();
}
}
Line 71, Column 10
Tab Size: 4
Java

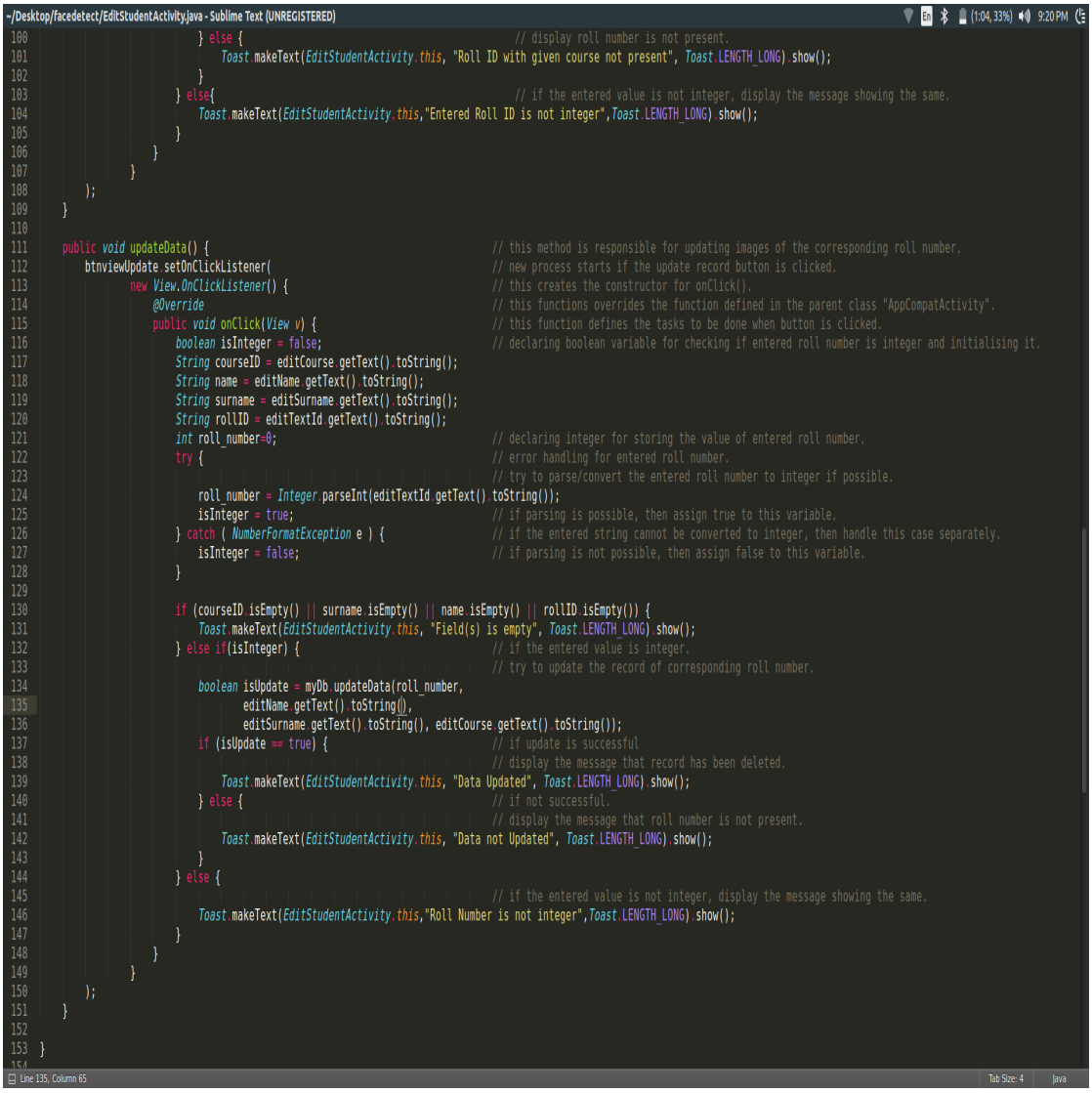
6 Edit Student

~/Desktop/facedetect/EditStudentActivity.java - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : EditStudentActivity
Date of Creation    : 10-04-2018
Author              : Akul Agrawal

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public void onCreate(Bundle)
                      public void updateImage()
                      public void updateData()
                      public void onClick(View)

</header>
*/

package org.opencv.samples.facedetect;

// importing package for android
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class EditStudentActivity extends AppCompatActivity {

    private DBHelper myDb;                      // instance of database containing student record.
    private EditText editName;                  // controlling name to be entered by user.
    private EditText editSurname;               // controlling surname to be entered by user.
    private EditText editCourse;                // controlling course-id to be entered by user.
    private EditText editTextId;                // controlling roll number to be entered by user.
    private Button btnviewUpdate;               // controlling the update button.
    private DBImgHelper imDb;                   // instance of database containing images.
    private Button button;                      // controlling reset-image option button.

    @Override                                               // this functions overrides the function defined in the parent class "AppCompatActivity".
    public void onCreate(Bundle savedInstanceState) {       // this method is called when the activity is first created.
        super.onCreate(savedInstanceState);                 // creates the activity.
        setContentView(R.layout.activity_edit_student);     // sets the content UI to be displayed on screen.
        myDb = new DBHelper(this);                          // creating instance of the students' record database.
        imDb = new DBImgHelper(this);                       // creating instance of the students' image database.

        editName = (EditText)findViewById(R.id.editText_name);      // locating the name textbox and creating instance of it.
        editSurname = (EditText)findViewById(R.id.editText_surname); // locating the surname textbox and creating instance of it.
        editCourse = (EditText)findViewById(R.id.editText_Course);   // locating the course id textbox and creating instance of it.
        editTextId = (EditText)findViewById(R.id.editText_id);       // locating the roll number textbox and creating instance of it
```

7 characters selected — Tab Size: 4 — Java

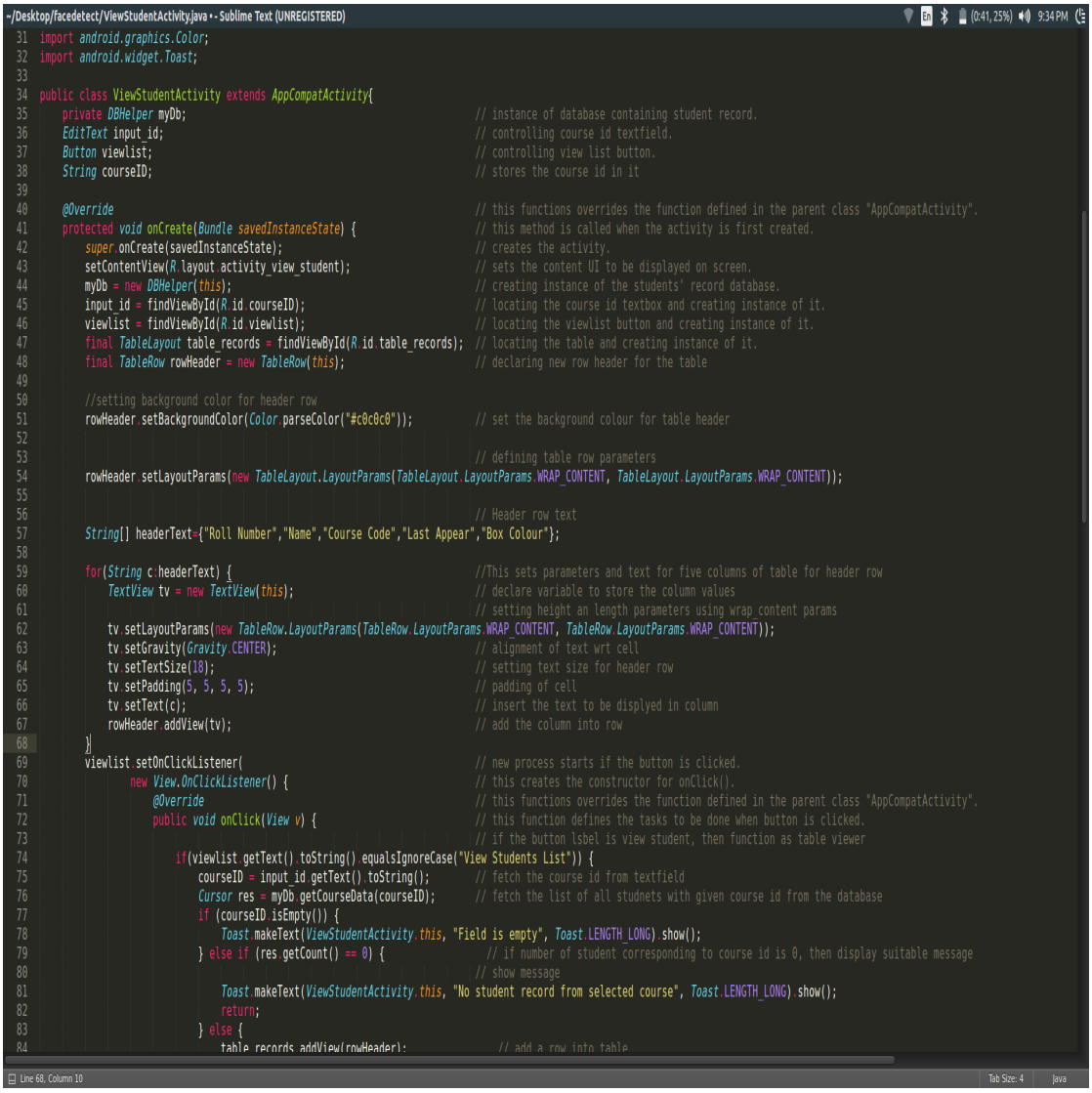

~/Desktop/facedetect/EditStudentActivity.java - Sublime Text (UNREGISTERED)
editName = (EditText)findViewById(R.id.editText_name);
editSurname = (EditText)findViewById(R.id.editText_surname);
editCourse = (EditText)findViewById(R.id.editText_Course);
editTextId = (EditText)findViewById(R.id.editText_id);
btnviewUpdate= (Button)findViewById(R.id.button_update);
button = (Button) findViewById(R.id.ImgButton);
updateImage();
updateData();
public void updateImage() {
button.setOnClickListener(
new View.OnClickListener() {
@Override
public void onClick(View v) {
boolean isInteger = false;
String courseID = editCourse.getText().toString();
String name = editName.getText().toString();
String surname = editSurname.getText().toString();
String rollID = editTextId.getText().toString();
int roll_number=0;
try {
roll_number = Integer.parseInt(editTextId.getText().toString());
isInteger = true;
} catch (NumberFormatException e) {
isInteger = false;
}
if (courseID.isEmpty() || surname.isEmpty() || name.isEmpty() || rollID.isEmpty()) {
Toast.makeText(EditStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
} else if(isInteger) {
if (myDb.ifStudentExists(roll_number, courseID)) {
Integer deletedImg = imDb.deleteImg(roll_number);
if (deletedImg > 0) {
Toast.makeText(EditStudentActivity.this, "Images Deleted", Toast.LENGTH_LONG).show();
Intent intent = new Intent(EditStudentActivity.this, AddImages.class);
intent.putExtra("id", editTextId.getText().toString());
startActivity(intent);
finish();
} else{
Toast.makeText(EditStudentActivity.this, "Roll ID not present in database", Toast.LENGTH_LONG).show();
}
} else {
Toast.makeText(EditStudentActivity.this, "Roll ID with given course not present", Toast.LENGTH_LONG).show();
}
} else{
Toast.makeText(EditStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
}
Line 16, Column 43
Tab Size: 4
Java

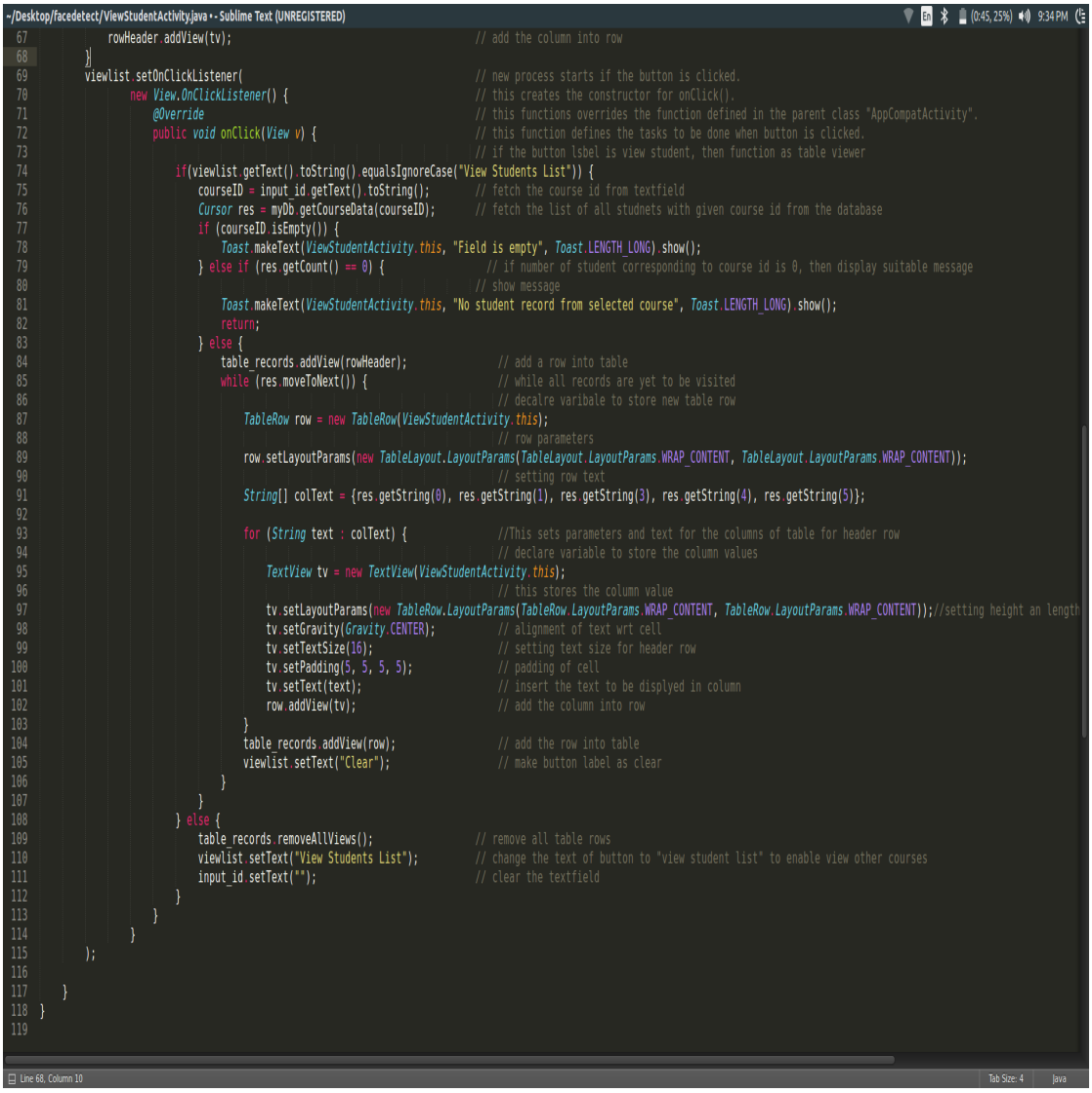

~/Desktop/facedetect/EditStudentActivity.java - Sublime Text (UNREGISTERED)
9:20 PM
Line 135, Column 65
Tab Size: 4
Java

7 View Student List

~/Desktop/facedetect/ViewStudentActivity.java - Sublime Text (UNREGISTERED)

```java
/*
<header>
Module Name         : ViewStudentActivity
Date of Creation    : 10-04-2018
Author              : Sujoy Ghosh

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public void onCreate(Bundle)
                      public void onClick(View)

</header>
*/

package org.opencv.samples.facedetect;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.view.Gravity;
import android.graphics.Color;
import android.widget.Toast;

public class ViewStudentActivity extends AppCompatActivity{
    private DBHelper myDb;                                          // instance of database containing student record.
    EditText input_id;                                              // controlling course id textfield.
    Button viewlist;                                                // controlling view list button.
    String courseID;                                                // stores the course id in it

    @Override                                                       // this functions overrides the function defined in the parent class "AppCompatActivity".
    protected void onCreate(Bundle savedInstanceState) {            // this method is called when the activity is first created.
        super.onCreate(savedInstanceState);                         // creates the activity.
        setContentView(R.layout.activity_view_student);             // sets the content UI to be displayed on screen.
        myDb = new DBHelper(this);                                  // creating instance of the students' record database.
        input_id = findViewById(R.id.courseID);                     // locating the course id textbox and creating instance of it.
        viewlist = findViewById(R.id.viewlist);                     // locating the viewlist button and creating instance of it.
        final TableLayout table_records = findViewById(R.id.table_records);  // locating the table and creating instance of it.
        final TableRow rowHeader = new TableRow(this);              // declaring new row header for the table

        //setting background color for header row
        rowHeader.setBackgroundColor(Color.parseColor("#c0c0c0"));  // set the background colour for table header

                                                                    // defining table row parameters
        rowHeader.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));
```


~/Desktop/facedetect/ViewStudentActivity.java • - Sublime Text (UNREGISTERED)
(0:41, 25%)
9:34 PM
import android.graphics.Color;
import android.widget.Toast;
public class ViewStudentActivity extends AppCompatActivity{
private DBHelper myDb; // instance of database containing student record.
EditText input_id; // controlling course id textfield.
Button viewlist; // controlling view list button.
String courseID; // stores the course id in it
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
protected void onCreate(Bundle savedInstanceState) { // this method is called when the activity is first created.
super.onCreate(savedInstanceState); // creates the activity.
setContentView(R.layout.activity_view_student); // sets the content UI to be displayed on screen.
myDb = new DBHelper(this); // creating instance of the students' record database.
input_id = findViewById(R.id.courseID); // locating the course id textbox and creating instance of it.
viewlist = findViewById(R.id.viewlist); // locating the viewlist button and creating instance of it.
final TableLayout table_records = findViewById(R.id.table_records); // locating the table and creating instance of it.
final TableRow rowHeader = new TableRow(this); // declaring new row header for the table
//setting background color for header row
rowHeader.setBackgroundColor(Color.parseColor("#c0c0c0")); // set the background colour for table header
// defining table row parameters
rowHeader.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));
// Header row text
String[] headerText={"Roll Number","Name","Course Code","Last Appear","Box Colour"};
for(String c:headerText) { //This sets parameters and text for five columns of table for header row
TextView tv = new TextView(this); // declare variable to store the column values
// setting height an length parameters using wrap content params
tv.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
tv.setGravity(Gravity.CENTER); // alignment of text wrt cell
tv.setTextSize(18); // setting text size for header row
tv.setPadding(5, 5, 5, 5); // padding of cell
tv.setText(c); // insert the text to be displyed in column
rowHeader.addView(tv); // add the column into row
}
viewlist.setOnClickListener(// new process starts if the button is clicked.
new View.OnClickListener() { // this creates the constructor for onClick().
@Override // this functions overrides the function defined in the parent class "AppCompatActivity".
public void onClick(View v) { // this function defines the tasks to be done when button is clicked.
// if the button lsbel is view student, then function as table viewer
if(viewlist.getText().toString().equalsIgnoreCase("View Students List")) {
courseID = input_id.getText().toString(); // fetch the course id from textfield
Cursor res = myDb.getCourseData(courseID); // fetch the list of all studnets with given course id from the database
if (courseID.isEmpty()) {
Toast.makeText(ViewStudentActivity.this, "Field is empty", Toast.LENGTH_LONG).show();
} else if (res.getCount() == 0) { // if number of student corresponding to course id is 0, then display suitable message
// show message
Toast.makeText(ViewStudentActivity.this, "No student record from selected course", Toast.LENGTH_LONG).show();
return;
} else {
table_records.addView(rowHeader); // add a row into table
Line 68, Column 10
Tab Size: 4
Java

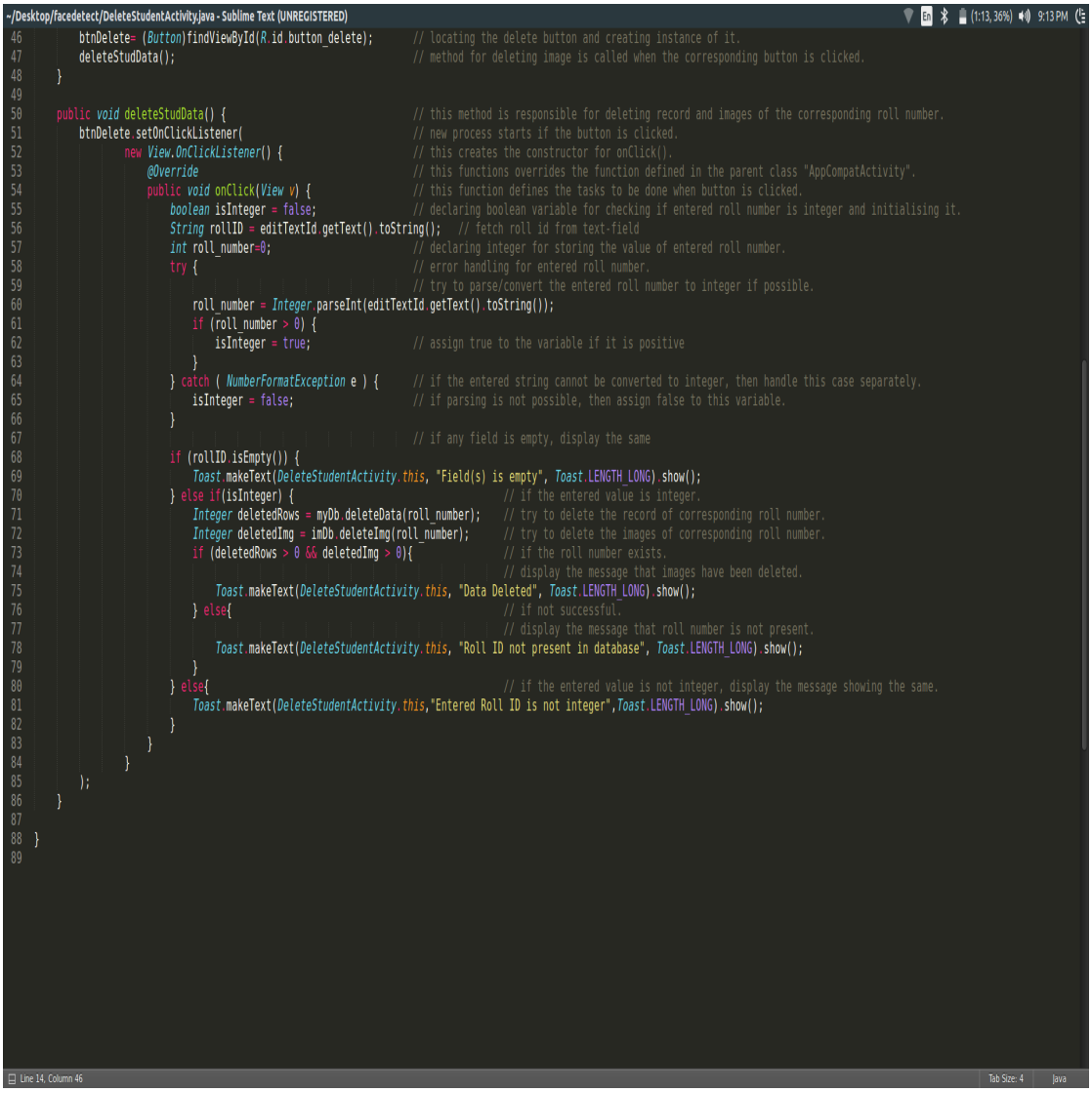

~/Desktop/facedetect/ViewStudentActivity.java • - Sublime Text (UNREGISTERED)
rowHeader.addView(tv);
}
viewlist.setOnClickListener(
new View.OnClickListener() {
@Override
public void onClick(View v) {
if(viewlist.getText().toString().equalsIgnoreCase("View Students List")) {
courseID = input_id.getText().toString();
Cursor res = myDb.getCourseData(courseID);
if (courseID.isEmpty()) {
Toast.makeText(ViewStudentActivity.this, "Field is empty", Toast.LENGTH_LONG).show();
} else if (res.getCount() == 0) {
// show message
Toast.makeText(ViewStudentActivity.this, "No student record from selected course", Toast.LENGTH_LONG).show();
return;
} else {
table_records.addView(rowHeader);
while (res.moveToNext()) {
TableRow row = new TableRow(ViewStudentActivity.this);
row.setLayoutParams(new TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT, TableLayout.LayoutParams.WRAP_CONTENT));
String[] colText = {res.getString(0), res.getString(1), res.getString(3), res.getString(4), res.getString(5)};
for (String text : colText) {
TextView tv = new TextView(ViewStudentActivity.this);
tv.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT, TableRow.LayoutParams.WRAP_CONTENT));
tv.setGravity(Gravity.CENTER);
tv.setTextSize(16);
tv.setPadding(5, 5, 5, 5);
tv.setText(text);
row.addView(tv);
}
table_records.addView(row);
viewlist.setText("Clear");
}
}
} else {
table_records.removeAllViews();
viewlist.setText("View Students List");
input_id.setText("");
}
}
}
);
}
}
Line 68, Column 10
Tab Size: 4
Java

8 Delete Student

~/Desktop/facedetect/DeleteStudentActivity.java - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : DeleteStudentActivity
Date of Creation    : 10-04-2018
Author              : Akul Agrawal

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : protected void onCreate(Bundle)
                      public void deleteStudData()
                      public void onClick(View)
</header>
*/

package org.opencv.samples.facedetect;

// importing package for android
import android.database.Cursor;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class DeleteStudentActivity extends AppCompatActivity {

    private DBHelper myDb;                                  // instance of database containing student record.
    private DBImgHelper imDb;                               // instance of database containing student images.
    private EditText editTextId;                            // controlling course-id to be entered by user.
    private Button btnDelete;                               // controlling delete button.

    @Override                                               // this functions overrides the function defined in the parent class "AppCompatActivity".
    protected void onCreate(Bundle savedInstanceState) {    // this method is called when the activity is first created.
        super.onCreate(savedInstanceState);                 // creates the activity.
        setContentView(R.layout.activity_delete_student);   // sets the content UI to be displayed on screen.
        myDb = new DBHelper(this);                          // creating instance of the students' record database.
        imDb = new DBImgHelper(this);                       // creating instance of the students' image database.

        editTextId = (EditText)findViewById(R.id.editText_id);  // locating the course id textbox and creating instance of it.
        btnDelete= (Button)findViewById(R.id.button_delete);    // locating the delete button and creating instance of it.
        deleteStudData();                                   // method for deleting image is called when the corresponding button is clicked.
    }

    public void deleteStudData() {                          // this method is responsible for deleting record and images of the corresponding roll number.
        btnDelete.setOnClickListener(                       // new process starts if the button is clicked.
                new View.OnClickListener() {                // this creates the constructor for onClick().
                    @Override                               // this functions overrides the function defined in the parent class "AppCompatActivity".
                    public void onClick(View v) {           // this function defines the tasks to be done when button is clicked.
                        boolean isInteger = false;          // declaring boolean variable for checking if entered roll number is integer and initializing it
```

Line 14, Column 46 — Tab Size: 4 — Java

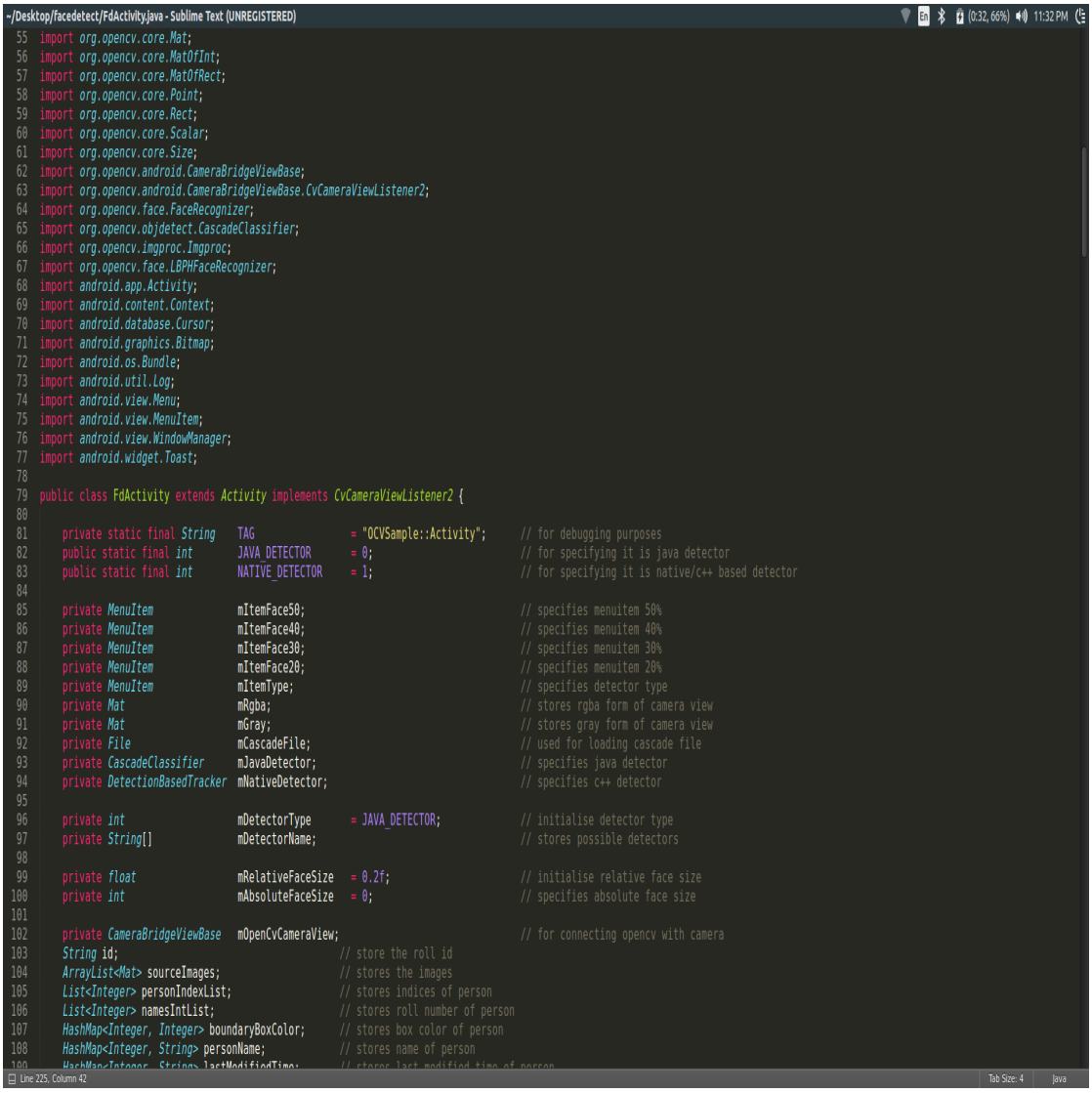

~/Desktop/facedetect/DeleteStudentActivity.java - Sublime Text (UNREGISTERED)
9:13 PM
btnDelete= (Button)findViewById(R.id.button_delete);
deleteStudData();
public void deleteStudData() {
btnDelete.setOnClickListener(
new View.OnClickListener() {
@Override
public void onClick(View v) {
boolean isInteger = false;
String rollID = editTextId.getText().toString();
int roll_number=0;
try {
roll_number = Integer.parseInt(editTextId.getText().toString());
if (roll_number > 0) {
isInteger = true;
}
} catch (NumberFormatException e) {
isInteger = false;
}
if (rollID.isEmpty()) {
Toast.makeText(DeleteStudentActivity.this, "Field(s) is empty", Toast.LENGTH_LONG).show();
} else if(isInteger) {
Integer deletedRows = myDb.deleteData(roll_number);
Integer deletedImg = imDb.deleteImg(roll_number);
if (deletedRows > 0 && deletedImg > 0){
Toast.makeText(DeleteStudentActivity.this, "Data Deleted", Toast.LENGTH_LONG).show();
} else{
Toast.makeText(DeleteStudentActivity.this, "Roll ID not present in database", Toast.LENGTH_LONG).show();
}
} else{
Toast.makeText(DeleteStudentActivity.this,"Entered Roll ID is not integer",Toast.LENGTH_LONG).show();
}
}
}
);
}
}
Line 14, Column 46
Tab Size: 4
Java

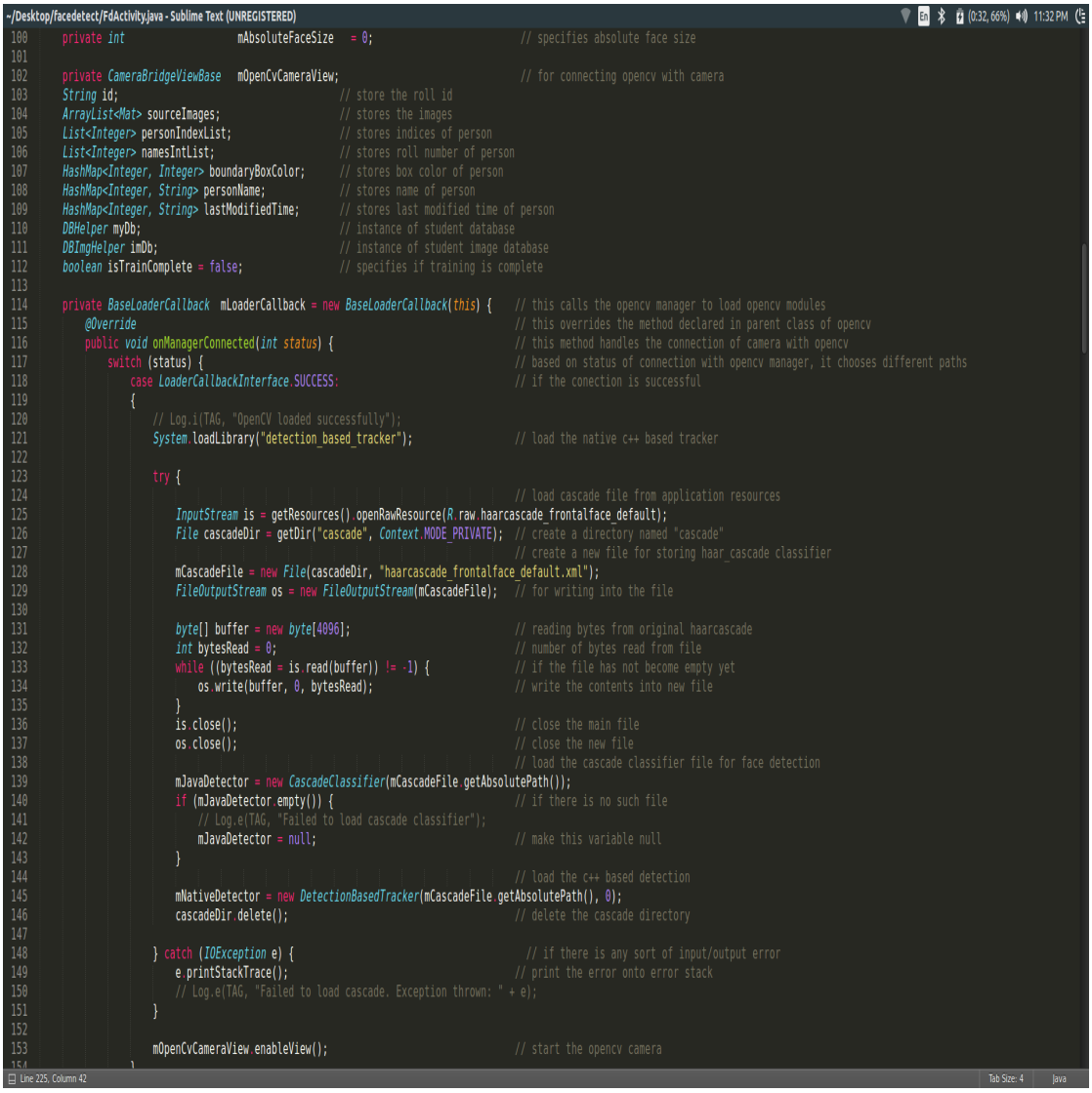
9 Camera Session

~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)

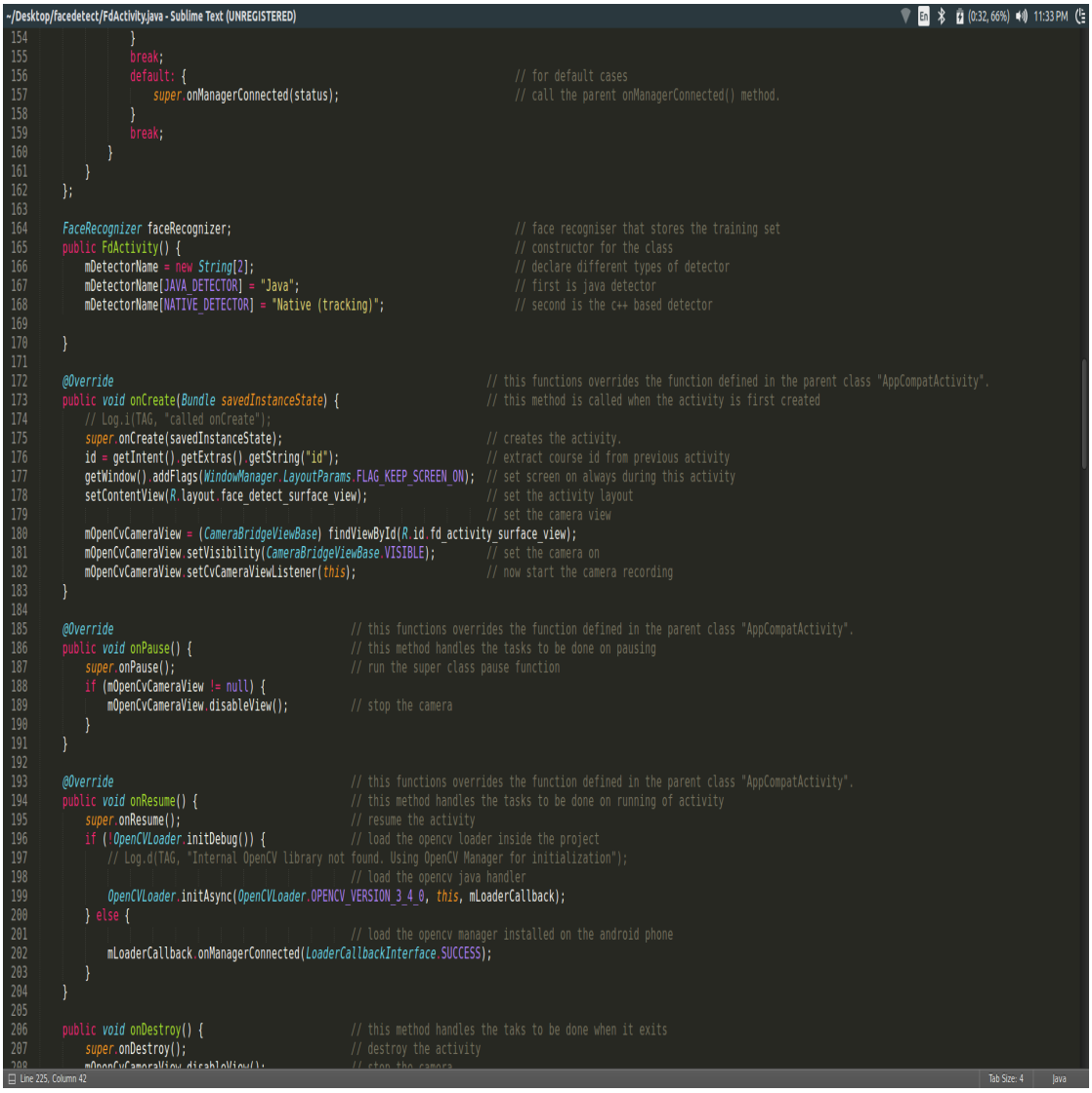
```
/*
<header>
Module Name         : FaceRecognitionActivity
Date of Creation    : 15-04-2018
Author              : Sujoy Ghosh

Modification History : added LBPHFaceRecogniser() instead of FisherFaceRecogniser()

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public FdActivity()
                      public void onManagerConnected(int)
                      public void onCreate(Bundle)
                      public void onPause()
                      public void onResume()
                      public void onDestroy()
                      public void onCameraViewStarted(int, int)
                      public void onCameraViewStopped()
                      private Scalar getBoxColor(int)
                      private long getTimeDiff(String, String)
                      public void training()
                      public void onBackPressed()
                      public Mat onCameraFrame(CvCameraViewFrame)
                      public boolean onCreateOptionsMenu(Menu)
                      private void setDetectorType(int)
                      private void setMinFaceSize(float)
                      public boolean onOptionsItemSelected(MenuItem)
</header>
*/

package org.opencv.samples.facedetect;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Random;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.Mat;
```

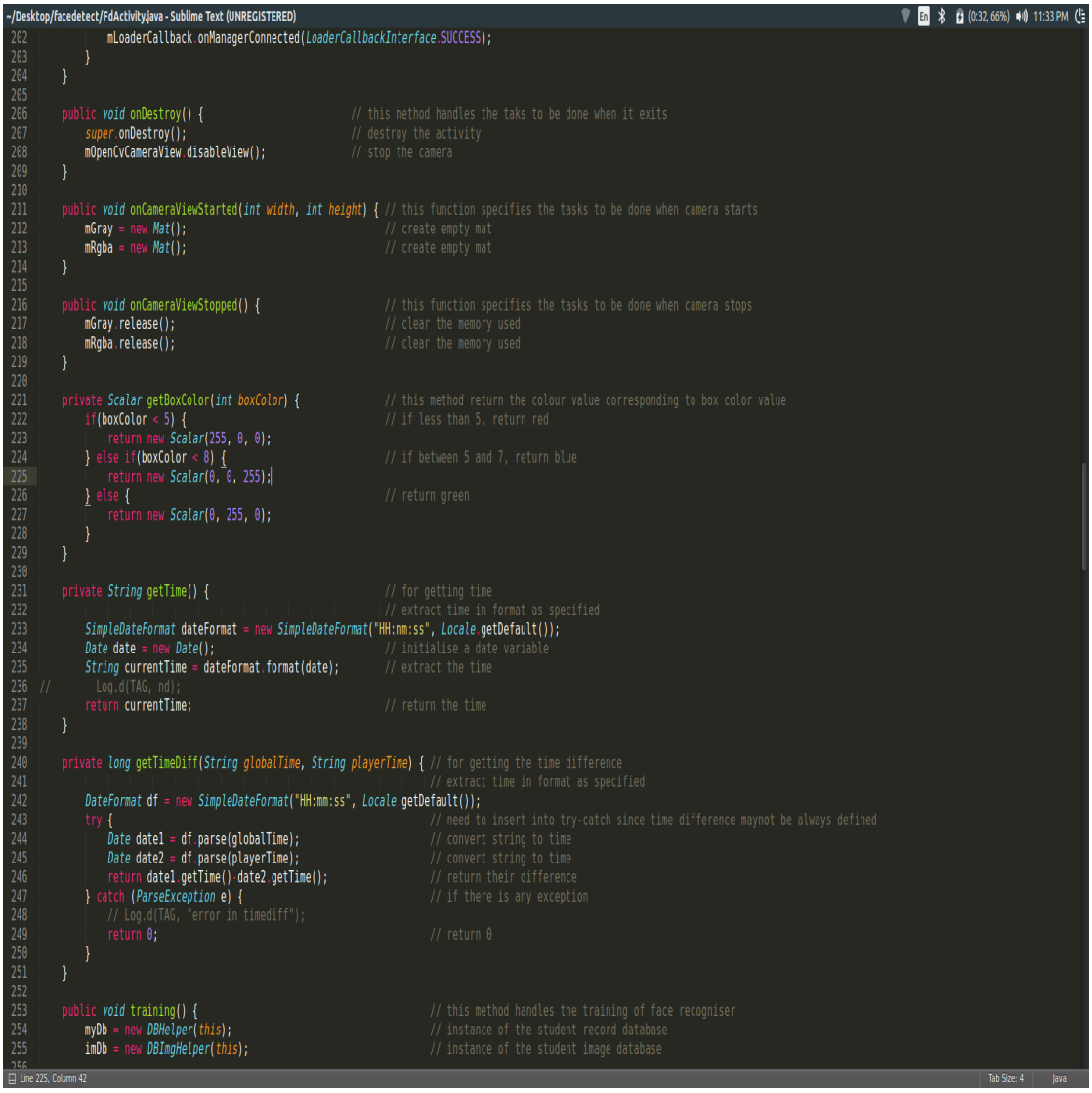
Line 225, Column 42 — Tab Size: 4 — Java

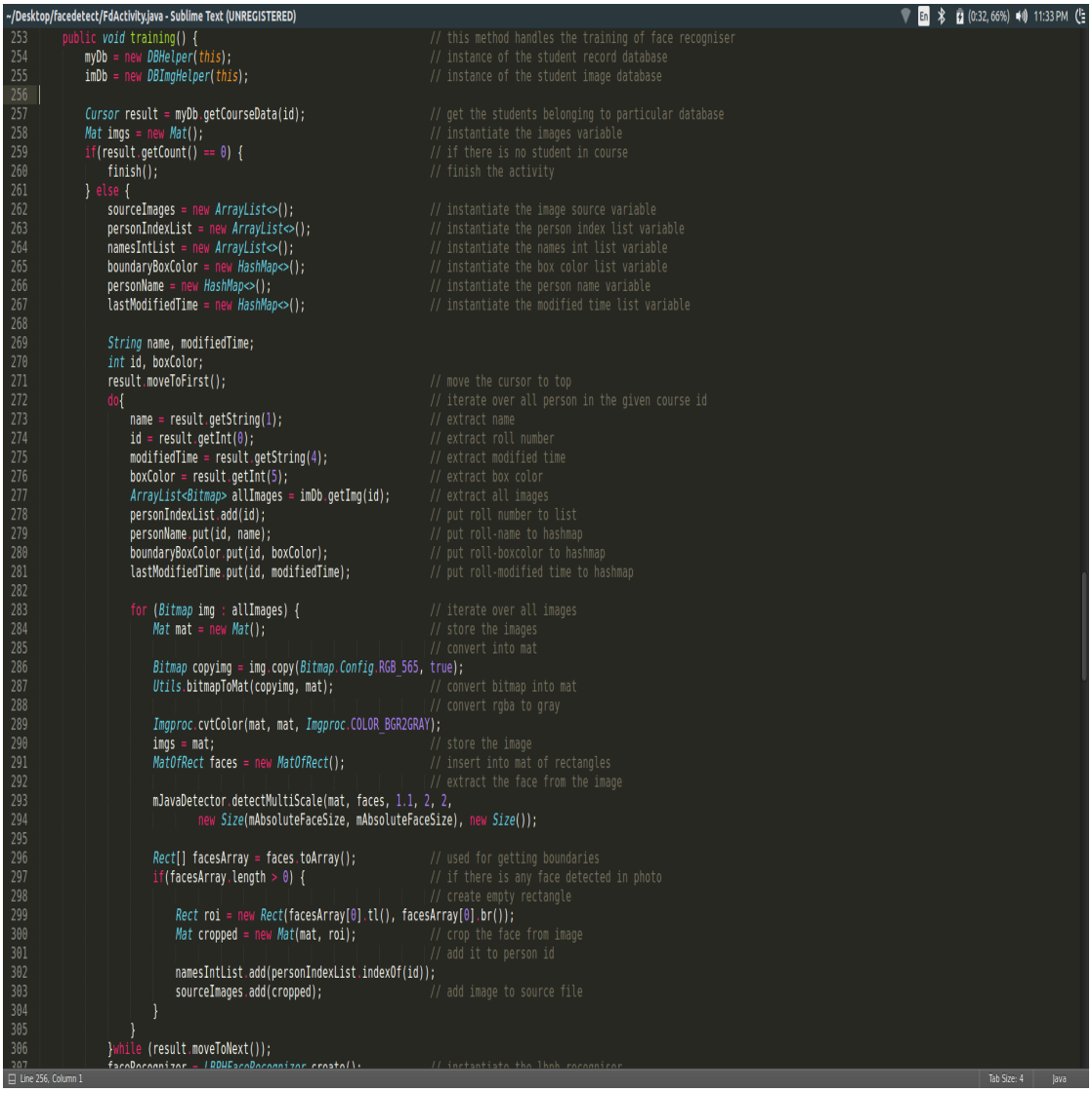

~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
11:32 PM
import org.opencv.core.Mat;
import org.opencv.core.MatOfInt;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;
import org.opencv.face.FaceRecognizer;
import org.opencv.objdetect.CascadeClassifier;
import org.opencv.imgproc.Imgproc;
import org.opencv.face.LBPHFaceRecognizer;
import android.app.Activity;
import android.content.Context;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.WindowManager;
import android.widget.Toast;
public class FdActivity extends Activity implements CvCameraViewListener2 {
private static final String TAG = "OCVSample::Activity"; // for debugging purposes
public static final int JAVA_DETECTOR = 0; // for specifying it is java detector
public static final int NATIVE_DETECTOR = 1; // for specifying it is native/c++ based detector
private MenuItem mItemFace50; // specifies menuitem 50%
private MenuItem mItemFace40; // specifies menuitem 40%
private MenuItem mItemFace30; // specifies menuitem 30%
private MenuItem mItemFace20; // specifies menuitem 20%
private MenuItem mItemType; // specifies detector type
private Mat mRgba; // stores rgba form of camera view
private Mat mGray; // stores gray form of camera view
private File mCascadeFile; // used for loading cascade file
private CascadeClassifier mJavaDetector; // specifies java detector
private DetectionBasedTracker mNativeDetector; // specifies c++ detector
private int mDetectorType = JAVA_DETECTOR; // initialise detector type
private String[] mDetectorName; // stores possible detectors
private float mRelativeFaceSize = 0.2f; // initialise relative face size
private int mAbsoluteFaceSize = 0; // specifies absolute face size
private CameraBridgeViewBase mOpenCvCameraView; // for connecting opencv with camera
String id; // store the roll id
ArrayList<Mat> sourceImages; // stores the images
List<Integer> personIndexList; // stores indices of person
List<Integer> namesIntList; // stores roll number of person
HashMap<Integer, Integer> boundaryBoxColor; // stores box color of person
HashMap<Integer, String> personName; // stores name of person
Line 225, Column 42
Tab Size: 4
Java

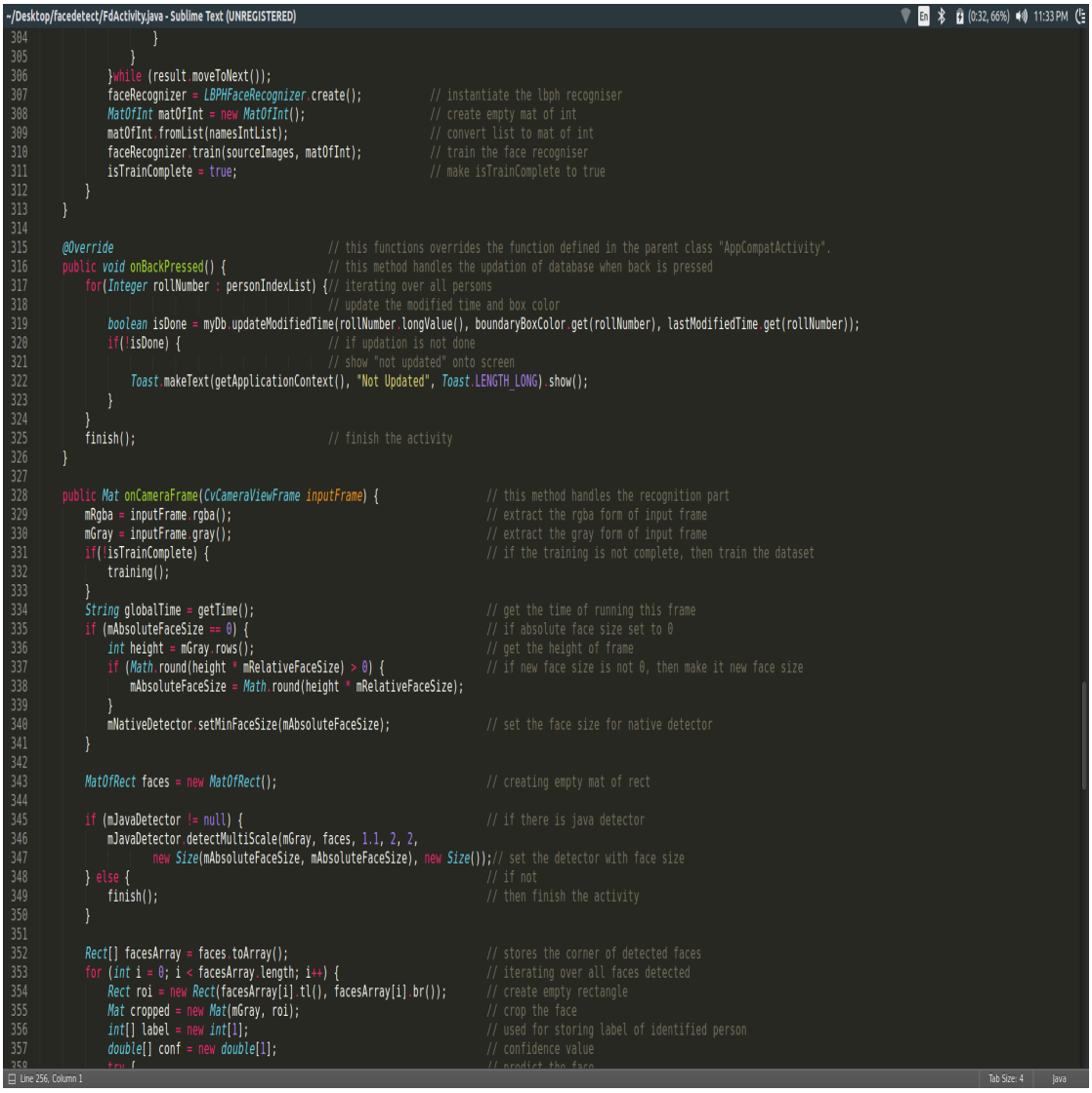



```
    private int                  mAbsoluteFaceSize   = 0;                        // specifies absolute face size

    private CameraBridgeViewBase   mOpenCvCameraView;                            // for connecting opencv with camera
    String id;                                  // store the roll id
    ArrayList<Mat> sourceImages;                // stores the images
    List<Integer> personIndexList;              // stores indices of person
    List<Integer> namesIntList;                 // stores roll number of person
    HashMap<Integer, Integer> boundaryBoxColor;     // stores box color of person
    HashMap<Integer, String> personName;            // stores name of person
    HashMap<Integer, String> lastModifiedTime;      // stores last modified time of person
    DBHelper myDb;                              // instance of student database
    DBImgHelper imDb;                           // instance of student image database
    boolean isTrainComplete = false;            // specifies if training is complete

    private BaseLoaderCallback  mLoaderCallback = new BaseLoaderCallback(this) {    // this calls the opencv manager to load opencv modules
        @Override                                                                   // this overrides the method declared in parent class of opencv
        public void onManagerConnected(int status) {                                // this method handles the connection of camera with opencv
            switch (status) {                                                       // based on status of connection with opencv manager, it chooses different paths
                case LoaderCallbackInterface.SUCCESS:                               // if the conection is successful
                {
                    // Log.i(TAG, "OpenCV loaded successfully");
                    System.loadLibrary("detection_based_tracker");                  // load the native c++ based tracker

                    try {
                                                                                    // load cascade file from application resources
                        InputStream is = getResources().openRawResource(R.raw.haarcascade_frontalface_default);
                        File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);  // create a directory named "cascade"
                                                                                    // create a new file for storing haar cascade classifier
                        mCascadeFile = new File(cascadeDir, "haarcascade_frontalface_default.xml");
                        FileOutputStream os = new FileOutputStream(mCascadeFile);   // for writing into the file

                        byte[] buffer = new byte[4096];                             // reading bytes from original haarcascade
                        int bytesRead = 0;                                          // number of bytes read from file
                        while ((bytesRead = is.read(buffer)) != -1) {               // if the file has not become empty yet
                            os.write(buffer, 0, bytesRead);                         // write the contents into new file
                        }
                        is.close();                                                 // close the main file
                        os.close();                                                 // close the new file
                                                                                    // load the cascade classifier file for face detection
                        mJavaDetector = new CascadeClassifier(mCascadeFile.getAbsolutePath());
                        if (mJavaDetector.empty()) {                                // if there is no such file
                            // Log.e(TAG, "Failed to load cascade classifier");
                            mJavaDetector = null;                                   // make this variable null
                        }
                                                                                    // load the c++ based detection
                        mNativeDetector = new DetectionBasedTracker(mCascadeFile.getAbsolutePath(), 0);
                        cascadeDir.delete();                                        // delete the cascade directory

                    } catch (IOException e) {                                       // if there is any sort of input/output error
                        e.printStackTrace();                                        // print the error onto error stack
                        // Log.e(TAG, "Failed to load cascade. Exception thrown: " + e);
                    }

                    mOpenCvCameraView.enableView();                                 // start the opencv camera
                }
```

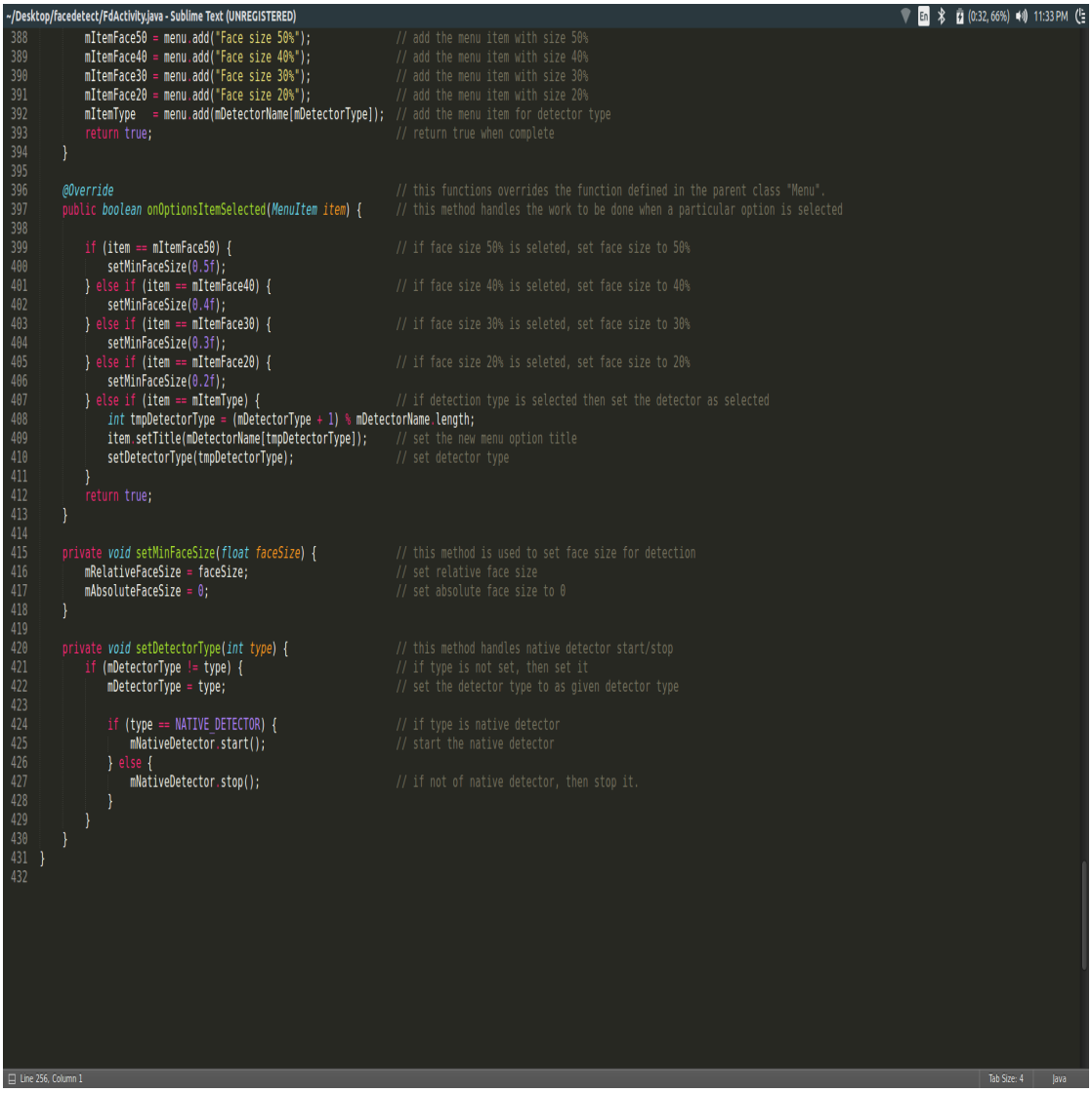

~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
(0:32, 66%) 11:33 PM
}
break;
default: {
super.onManagerConnected(status);
}
break;
}
}
};
FaceRecognizer faceRecognizer;
public FdActivity() {
mDetectorName = new String[2];
mDetectorName[JAVA_DETECTOR] = "Java";
mDetectorName[NATIVE_DETECTOR] = "Native (tracking)";
}
@Override
public void onCreate(Bundle savedInstanceState) {
// Log.i(TAG, "called onCreate");
super.onCreate(savedInstanceState);
id = getIntent().getExtras().getString("id");
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
setContentView(R.layout.face_detect_surface_view);
mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.fd_activity_surface_view);
mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
mOpenCvCameraView.setCvCameraViewListener(this);
}
@Override
public void onPause() {
super.onPause();
if (mOpenCvCameraView != null) {
mOpenCvCameraView.disableView();
}
}
@Override
public void onResume() {
super.onResume();
if (!OpenCVLoader.initDebug()) {
// Log.d(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");
OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0, this, mLoaderCallback);
} else {
mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
}
}
public void onDestroy() {
super.onDestroy();
Line 225, Column 42
Tab Size: 4
Java


~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
}
}
public void onDestroy() { // this method handles the taks to be done when it exits
super.onDestroy(); // destroy the activity
mOpenCvCameraView.disableView(); // stop the camera
}
public void onCameraViewStarted(int width, int height) { // this function specifies the tasks to be done when camera starts
mGray = new Mat(); // create empty mat
mRgba = new Mat(); // create empty mat
}
public void onCameraViewStopped() { // this function specifies the tasks to be done when camera stops
mGray.release(); // clear the memory used
mRgba.release(); // clear the memory used
}
private Scalar getBoxColor(int boxColor) { // this method return the colour value corresponding to box color value
if(boxColor < 5) { // if less than 5, return red
return new Scalar(255, 0, 0);
} else if(boxColor < 8) { // if between 5 and 7, return blue
return new Scalar(0, 0, 255);
} else { // return green
return new Scalar(0, 255, 0);
}
}
private String getTime() { // for getting time
// extract time in format as specified
SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
Date date = new Date(); // initialise a date variable
String currentTime = dateFormat.format(date); // extract the time
// Log.d(TAG, nd);
return currentTime; // return the time
}
private long getTimeDiff(String globalTime, String playerTime) { // for getting the time difference
// extract time in format as specified
DateFormat df = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
try { // need to insert into try-catch since time difference maynot be always defined
Date date1 = df.parse(globalTime); // convert string to time
Date date2 = df.parse(playerTime); // convert string to time
return date1.getTime() - date2.getTime(); // return their difference
} catch (ParseException e) { // if there is any exception
// Log.d(TAG, "error in timediff");
return 0; // return 0
}
}
public void training() { // this method handles the training of face recogniser
myDb = new DBHelper(this); // instance of the student record database
imDb = new DBImgHelper(this); // instance of the student image database
Line 225, Column 42
Tab Size: 4
Java

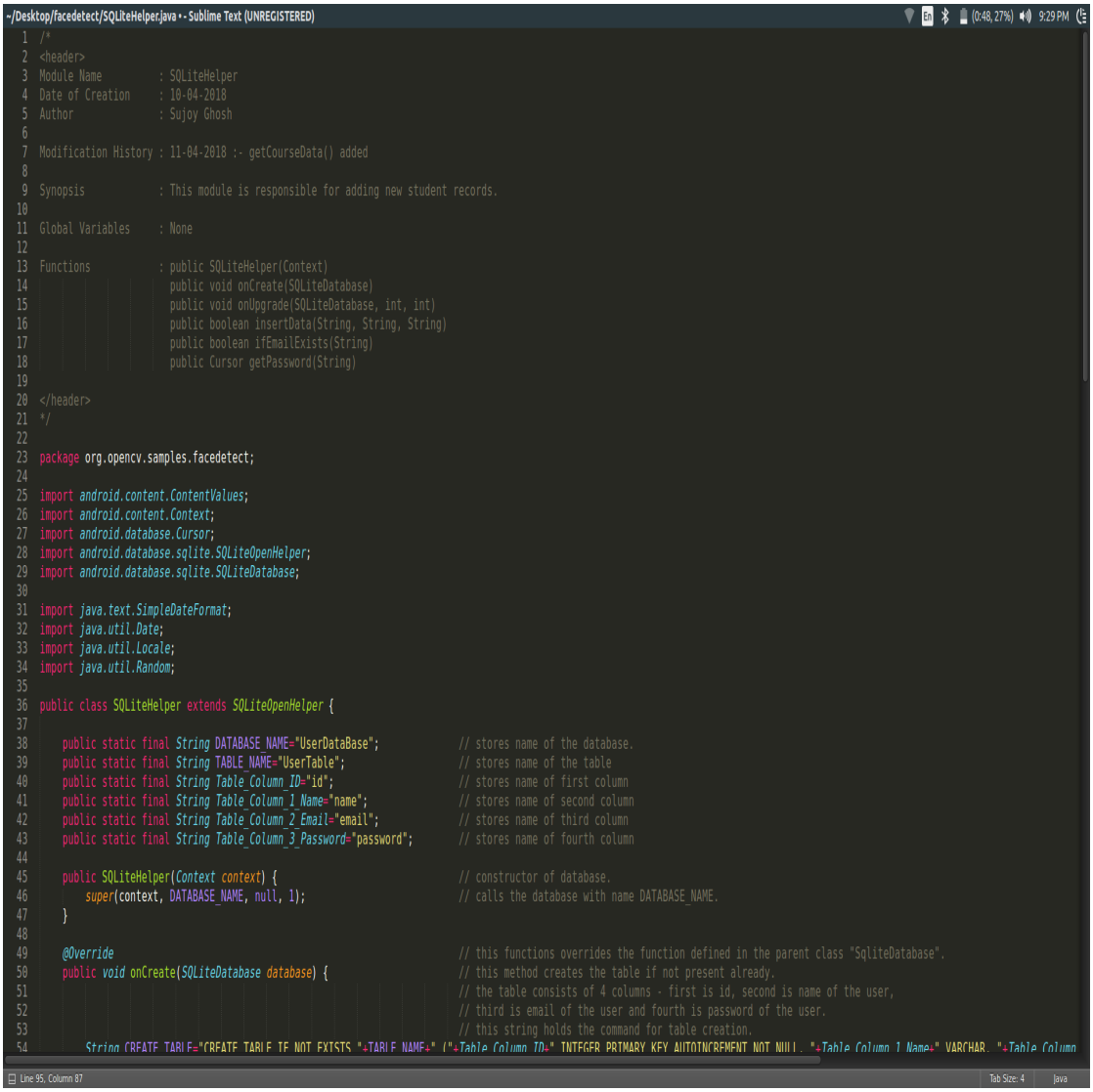

~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
11:33 PM
public void training() {
myDb = new DBHelper(this);
imDb = new DBImgHelper(this);
Cursor result = myDb.getCourseData(id);
Mat imgs = new Mat();
if(result.getCount() == 0) {
finish();
} else {
sourceImages = new ArrayList<>();
personIndexList = new ArrayList<>();
namesIntList = new ArrayList<>();
boundaryBoxColor = new HashMap<>();
personName = new HashMap<>();
lastModifiedTime = new HashMap<>();
String name, modifiedTime;
int id, boxColor;
result.moveToFirst();
do{
name = result.getString(1);
id = result.getInt(0);
modifiedTime = result.getString(4);
boxColor = result.getInt(5);
ArrayList<Bitmap> allImages = imDb.getImg(id);
personIndexList.add(id);
personName.put(id, name);
boundaryBoxColor.put(id, boxColor);
lastModifiedTime.put(id, modifiedTime);
for (Bitmap img : allImages) {
Mat mat = new Mat();
Bitmap copyimg = img.copy(Bitmap.Config.RGB_565, true);
Utils.bitmapToMat(copyimg, mat);
Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGR2GRAY);
imgs = mat;
MatOfRect faces = new MatOfRect();
mJavaDetector.detectMultiScale(mat, faces, 1.1, 2, 2,
new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());
Rect[] facesArray = faces.toArray();
if(facesArray.length > 0) {
Rect roi = new Rect(facesArray[0].tl(), facesArray[0].br());
Mat cropped = new Mat(mat, roi);
namesIntList.add(personIndexList.indexOf(id));
sourceImages.add(cropped);
}
}
}while (result.moveToNext());
Line 256, Column 1
Tab Size: 4
Java

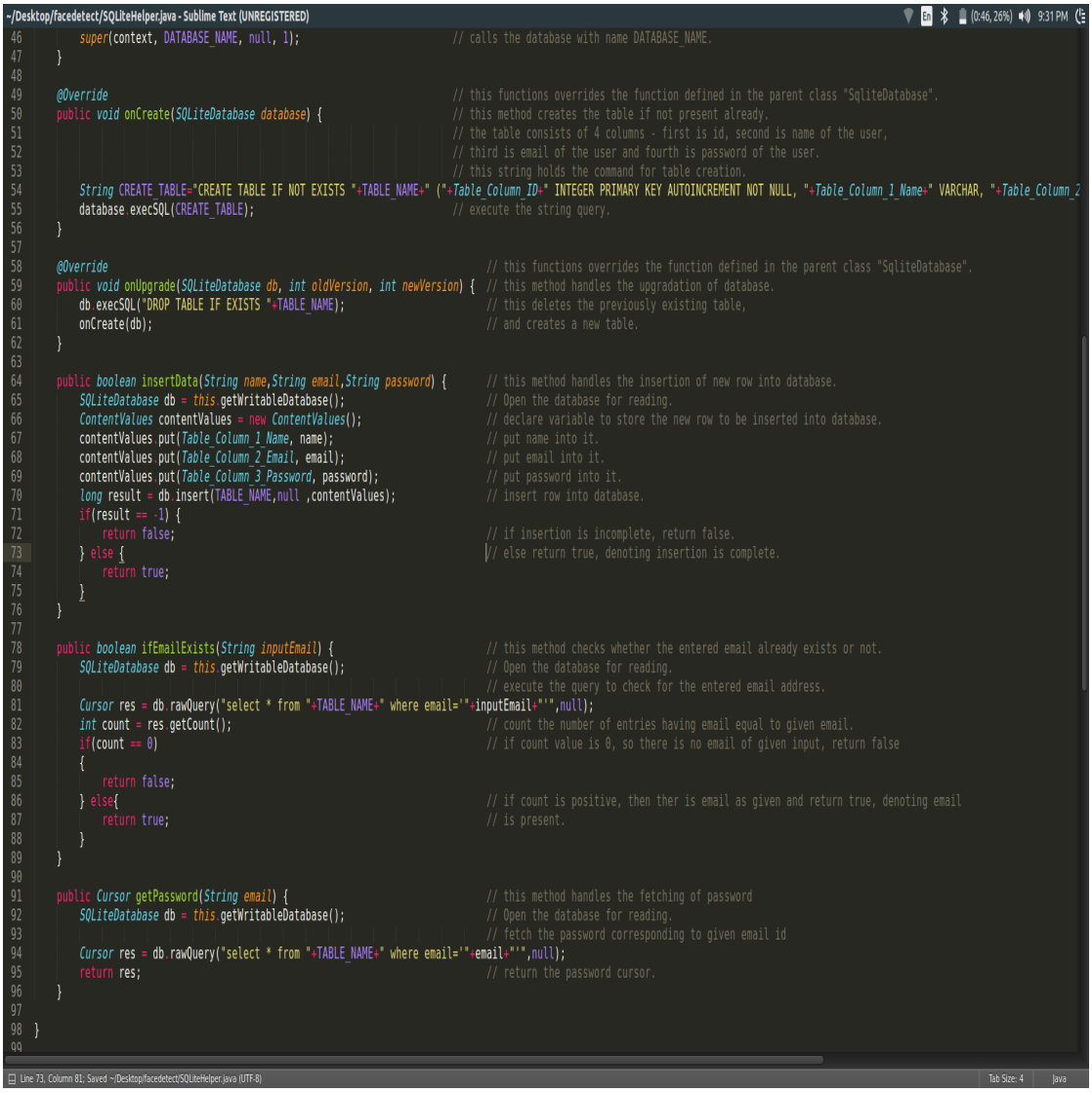

~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
(0:32, 66%) 11:33 PM
}
}
}while (result.moveToNext());
faceRecognizer = LBPHFaceRecognizer.create();
// instantiate the lbph recogniser
MatOfInt matOfInt = new MatOfInt();
// create empty mat of int
matOfInt.fromList(namesIntList);
// convert list to mat of int
faceRecognizer.train(sourceImages, matOfInt);
// train the face recogniser
isTrainComplete = true;
// make isTrainComplete to true
}
}
@Override
// this functions overrides the function defined in the parent class "AppCompatActivity".
public void onBackPressed() {
// this method handles the updation of database when back is pressed
for(Integer rollNumber : personIndexList) {// iterating over all persons
// update the modified time and box color
boolean isDone = myDb.updateModifiedTime(rollNumber.longValue(), boundaryBoxColor.get(rollNumber), lastModifiedTime.get(rollNumber));
if(!isDone) {
// if updation is not done
// show "not updated" onto screen
Toast.makeText(getApplicationContext(), "Not Updated", Toast.LENGTH_LONG).show();
}
}
finish();
// finish the activity
}
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
// this method handles the recognition part
mRgba = inputFrame.rgba();
// extract the rgba form of input frame
mGray = inputFrame.gray();
// extract the gray form of input frame
if(!isTrainComplete) {
// if the training is not complete, then train the dataset
training();
}
String globalTime = getTime();
// get the time of running this frame
if (mAbsoluteFaceSize == 0) {
// if absolute face size set to 0
int height = mGray.rows();
// get the height of frame
if (Math.round(height * mRelativeFaceSize) > 0) {
// if new face size is not 0, then make it new face size
mAbsoluteFaceSize = Math.round(height * mRelativeFaceSize);
}
mNativeDetector.setMinFaceSize(mAbsoluteFaceSize);
// set the face size for native detector
}
MatOfRect faces = new MatOfRect();
// creating empty mat of rect
if (mJavaDetector != null) {
// if there is java detector
mJavaDetector.detectMultiScale(mGray, faces, 1.1, 2, 2,
new Size(mAbsoluteFaceSize, mAbsoluteFaceSize), new Size());// set the detector with face size
} else {
// if not
finish();
// then finish the activity
}
Rect[] facesArray = faces.toArray();
// stores the corner of detected faces
for (int i = 0; i < facesArray.length; i++) {
// iterating over all faces detected
Rect roi = new Rect(facesArray[i].tl(), facesArray[i].br());
// create empty rectangle
Mat cropped = new Mat(mGray, roi);
// crop the face
int[] label = new int[1];
// used for storing label of identified person
double[] conf = new double[1];
// confidence value
Line 256, Column 1
Tab Size: 4
Java


```
            Mat cropped = new Mat(mGray, roi);                              // crop the face
            int[] label = new int[1];                                       // used for storing label of identified person
            double[] conf = new double[1];                                  // confidence value
            try {                                                           // predict the face
                faceRecognizer.predict(cropped, label, conf);
            } catch (Exception e) {
                Log.d(TAG, "Nothing");
            }
            Random randomVal = new Random();                                // used for generating random values
                                                                            // criteria for unknown face
            if(label[0] < 0 || conf[0] > 150.0) {                           // surround white rectangle with "unknown" lable
                Imgproc.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), new Scalar(255,255,255), 2);
                Imgproc.putText(mRgba,"Unknown "+conf[0],new Point(x,y),Core.FONT_HERSHEY_PLAIN, 1.0, new Scalar(255,255,255));
            } else {
                int roll = personIndexList.get(label[0]);                   // get the roll number of the student
                String playerTime = lastModifiedTime.get(roll);             // get time of checking
                int boxColor = boundaryBoxColor.get(roll);                  // get the boxcolor
                if(getTimeDiff(globalTime, playerTime) > 60000) {           // if the time differnce is greater than 1 minute, change the boxcolor and modification time
                    int newValue = randomVal.nextInt(9) + 1;
                    boxColor = newValue;
                    lastModifiedTime.put(roll, globalTime);
                    boundaryBoxColor.put(roll, newValue);
                }                                                           // put rectangle of corresponding color with label
                Imgproc.rectangle(mRgba, facesArray[i].tl(), facesArray[i].br(), getBoxColor(boxColor), 2);
                Imgproc.putText(mRgba,personName.get(roll)+" "+conf[0],new Point(x,y),Core.FONT_HERSHEY_PLAIN, 1.0, getBoxColor(boxColor));
            }
        }
        return mRgba;                                                       // return the final camera frame
    }

    @Override                                           // this functions overrides the function defined in the parent class "Menu".
    public boolean onCreateOptionsMenu(Menu menu) {     // this method handles the creation of menu for face size
        // Log.i(TAG, "called onCreateOptionsMenu");
        mItemFace50 = menu.add("Face size 50%");        // add the menu item with size 50%
        mItemFace40 = menu.add("Face size 40%");        // add the menu item with size 40%
        mItemFace30 = menu.add("Face size 30%");        // add the menu item with size 30%
        mItemFace20 = menu.add("Face size 20%");        // add the menu item with size 20%
        mItemType   = menu.add(mDetectorName[mDetectorType]);  // add the menu item for detector type
        return true;                                    // return true when complete
    }

    @Override                                           // this functions overrides the function defined in the parent class "Menu".
    public boolean onOptionsItemSelected(MenuItem item) {     // this method handles the work to be done when a particular option is selected

        if (item == mItemFace50) {                      // if face size 50% is seleted, set face size to 50%
            setMinFaceSize(0.5f);
        } else if (item == mItemFace40) {               // if face size 40% is seleted, set face size to 40%
            setMinFaceSize(0.4f);
        } else if (item == mItemFace30) {               // if face size 30% is seleted, set face size to 30%
            setMinFaceSize(0.3f);
        } else if (item == mItemFace20) {               // if face size 20% is seleted, set face size to 20%
            setMinFaceSize(0.2f);
        } else if (item == mItemType) {                 // if detection type is selected then set the detector as selected
            int tmpDetectorType = (mDetectorType + 1) % mDetectorName.length;
            item.setTitle(mDetectorName[tmpDetectorType]);      // set the new menu option title
```


~/Desktop/facedetect/FdActivity.java - Sublime Text (UNREGISTERED)
11:33 PM
mItemFace50 = menu.add("Face size 50%");
mItemFace40 = menu.add("Face size 40%");
mItemFace30 = menu.add("Face size 30%");
mItemFace20 = menu.add("Face size 20%");
mItemType = menu.add(mDetectorName[mDetectorType]);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
if (item == mItemFace50) {
setMinFaceSize(0.5f);
} else if (item == mItemFace40) {
setMinFaceSize(0.4f);
} else if (item == mItemFace30) {
setMinFaceSize(0.3f);
} else if (item == mItemFace20) {
setMinFaceSize(0.2f);
} else if (item == mItemType) {
int tmpDetectorType = (mDetectorType + 1) % mDetectorName.length;
item.setTitle(mDetectorName[tmpDetectorType]);
setDetectorType(tmpDetectorType);
}
return true;
}
private void setMinFaceSize(float faceSize) {
mRelativeFaceSize = faceSize;
mAbsoluteFaceSize = 0;
}
private void setDetectorType(int type) {
if (mDetectorType != type) {
mDetectorType = type;
if (type == NATIVE_DETECTOR) {
mNativeDetector.start();
} else {
mNativeDetector.stop();
}
}
}
}
Line 256, Column 1
Tab Size: 4
Java

10 User Database




~/Desktop/facedetect/SQLiteHelper.java - Sublime Text (UNREGISTERED)
9:31 PM
super(context, DATABASE_NAME, null, 1); // calls the database with name DATABASE_NAME.
}
@Override // this functions overrides the function defined in the parent class "SqliteDatabase".
public void onCreate(SQLiteDatabase database) { // this method creates the table if not present already.
// the table consists of 4 columns - first is id, second is name of the user,
// third is email of the user and fourth is password of the user.
// this string holds the command for table creation.
String CREATE_TABLE="CREATE TABLE IF NOT EXISTS "+TABLE_NAME+" ("+Table_Column_ID+" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+Table_Column_1_Name+" VARCHAR, "+Table_Column_2
database.execSQL(CREATE_TABLE); // execute the string query.
}
@Override // this functions overrides the function defined in the parent class "SqliteDatabase".
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { // this method handles the upgradation of database.
db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME); // this deletes the previously existing table,
onCreate(db); // and creates a new table.
}
public boolean insertData(String name,String email,String password) { // this method handles the insertion of new row into database.
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
ContentValues contentValues = new ContentValues(); // declare variable to store the new row to be inserted into database.
contentValues.put(Table_Column_1_Name, name); // put name into it.
contentValues.put(Table_Column_2_Email, email); // put email into it.
contentValues.put(Table_Column_3_Password, password); // put password into it.
long result = db.insert(TABLE_NAME,null ,contentValues); // insert row into database.
if(result == -1) {
return false; // if insertion is incomplete, return false.
} else { // else return true, denoting insertion is complete.
return true;
}
}
public boolean ifEmailExists(String inputEmail) { // this method checks whether the entered email already exists or not.
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
// execute the query to check for the entered email address.
Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where email='"+inputEmail+"'",null);
int count = res.getCount(); // count the number of entries having email equal to given email.
if(count == 0) // if count value is 0, so there is no email of given input, return false
{
return false;
} else{ // if count is positive, then ther is email as given and return true, denoting email
return true; // is present.
}
}
public Cursor getPassword(String email) { // this method handles the fetching of password
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
// fetch the password corresponding to given email id
Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where email='"+email+"'",null);
return res; // return the password cursor.
}
}
Line 73, Column 81; Saved ~/Desktop/facedetect/SQLiteHelper.java (UTF-8)
Tab Size: 4
Java

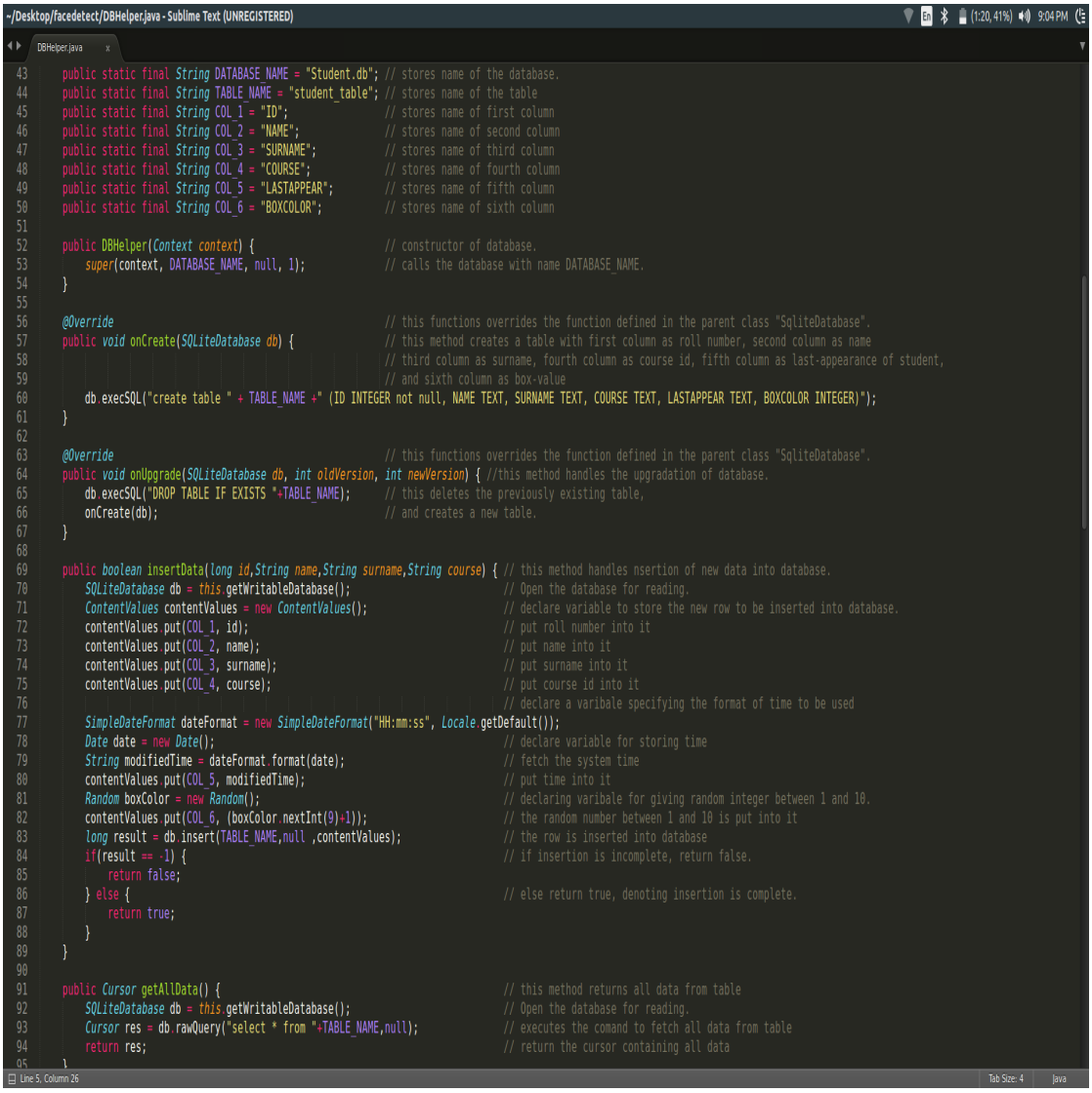
11 Student Database

~/Desktop/facedetect/DBHelper.java - Sublime Text (UNREGISTERED)

DBHelper.java

```
/*
<header>
Module Name         : DBHelper
Date of Creation    : 10-04-2018
Author              : Akul Agrawal

Modification History : 11-04-2018 :- getCourseData() added

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public DBHelper(Context)
                      public void onCreate(SQLiteDatabase)
                      public void onUpgrade(SQLiteDatabase, int, int)
                      public boolean insertData(long, String, String, String)
                      public Cursor getAllData()
                      public Cursor getCourseData(String)
                      public boolean updateData(long, String, String, String)
                      public boolean updateModifiedTime(long, int, String)
                      public Boolean ifStudentExists(long, String)
                      public Integer deleteData(long)

</header>
*/

package org.opencv.samples.facedetect;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.Random;


public class DBHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "Student.db"; // stores name of the database.
    public static final String TABLE_NAME = "student_table"; // stores name of the table
    public static final String COL_1 = "ID";                 // stores name of first column
    public static final String COL_2 = "NAME";               // stores name of second column
    public static final String COL_3 = "SURNAME";            // stores name of third column
    public static final String COL_4 = "COURSE";             // stores name of fourth column
    public static final String COL_5 = "LASTAPPEAR";         // stores name of fifth column
    public static final String COL_6 = "BOXCOLOR";           // stores name of sixth column

    public DBHelper(Context context) {                       // constructor of database.
        super(context, DATABASE_NAME, null, 1);              // calls the database with name DATABASE_NAME
```

Line 5, Column 26 Tab Size: 4 Java


~/Desktop/facedetect/DBHelper.java - Sublime Text (UNREGISTERED)
DBHelper.java
public static final String DATABASE_NAME = "Student.db"; // stores name of the database.
public static final String TABLE_NAME = "student_table"; // stores name of the table
public static final String COL_1 = "ID"; // stores name of first column
public static final String COL_2 = "NAME"; // stores name of second column
public static final String COL_3 = "SURNAME"; // stores name of third column
public static final String COL_4 = "COURSE"; // stores name of fourth column
public static final String COL_5 = "LASTAPPEAR"; // stores name of fifth column
public static final String COL_6 = "BOXCOLOR"; // stores name of sixth column
public DBHelper(Context context) { // constructor of database.
super(context, DATABASE_NAME, null, 1); // calls the database with name DATABASE_NAME.
}
@Override // this functions overrides the function defined in the parent class "SqliteDatabase".
public void onCreate(SQLiteDatabase db) { // this method creates a table with first column as roll number, second column as name
// third column as surname, fourth column as course id, fifth column as last-appearance of student,
// and sixth column as box-value
db.execSQL("create table " + TABLE_NAME +" (ID INTEGER not null, NAME TEXT, SURNAME TEXT, COURSE TEXT, LASTAPPEAR TEXT, BOXCOLOR INTEGER)");
}
@Override // this functions overrides the function defined in the parent class "SqliteDatabase".
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { //this method handles the upgradation of database.
db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME); // this deletes the previously existing table,
onCreate(db); // and creates a new table.
}
public boolean insertData(long id,String name,String surname,String course) { // this method handles nsertion of new data into database.
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
ContentValues contentValues = new ContentValues(); // declare variable to store the new row to be inserted into database.
contentValues.put(COL_1, id); // put roll number into it
contentValues.put(COL_2, name); // put name into it
contentValues.put(COL_3, surname); // put surname into it
contentValues.put(COL_4, course); // put course id into it
// declare a varibale specifying the format of time to be used
SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
Date date = new Date(); // declare variable for storing time
String modifiedTime = dateFormat.format(date); // fetch the system time
contentValues.put(COL_5, modifiedTime); // put time into it
Random boxColor = new Random(); // declaring varibale for giving random integer between 1 and 10.
contentValues.put(COL_6, (boxColor.nextInt(9)+1)); // the random number between 1 and 10 is put into it
long result = db.insert(TABLE_NAME,null ,contentValues); // the row is inserted into database
if(result == -1) { // if insertion is incomplete, return false.
return false;
} else { // else return true, denoting insertion is complete.
return true;
}
}
public Cursor getAllData() { // this method returns all data from table
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for reading.
Cursor res = db.rawQuery("select * from "+TABLE_NAME,null); // executes the comand to fetch all data from table
return res; // return the cursor containing all data
}
Line 5, Column 26
Tab Size: 4
Java

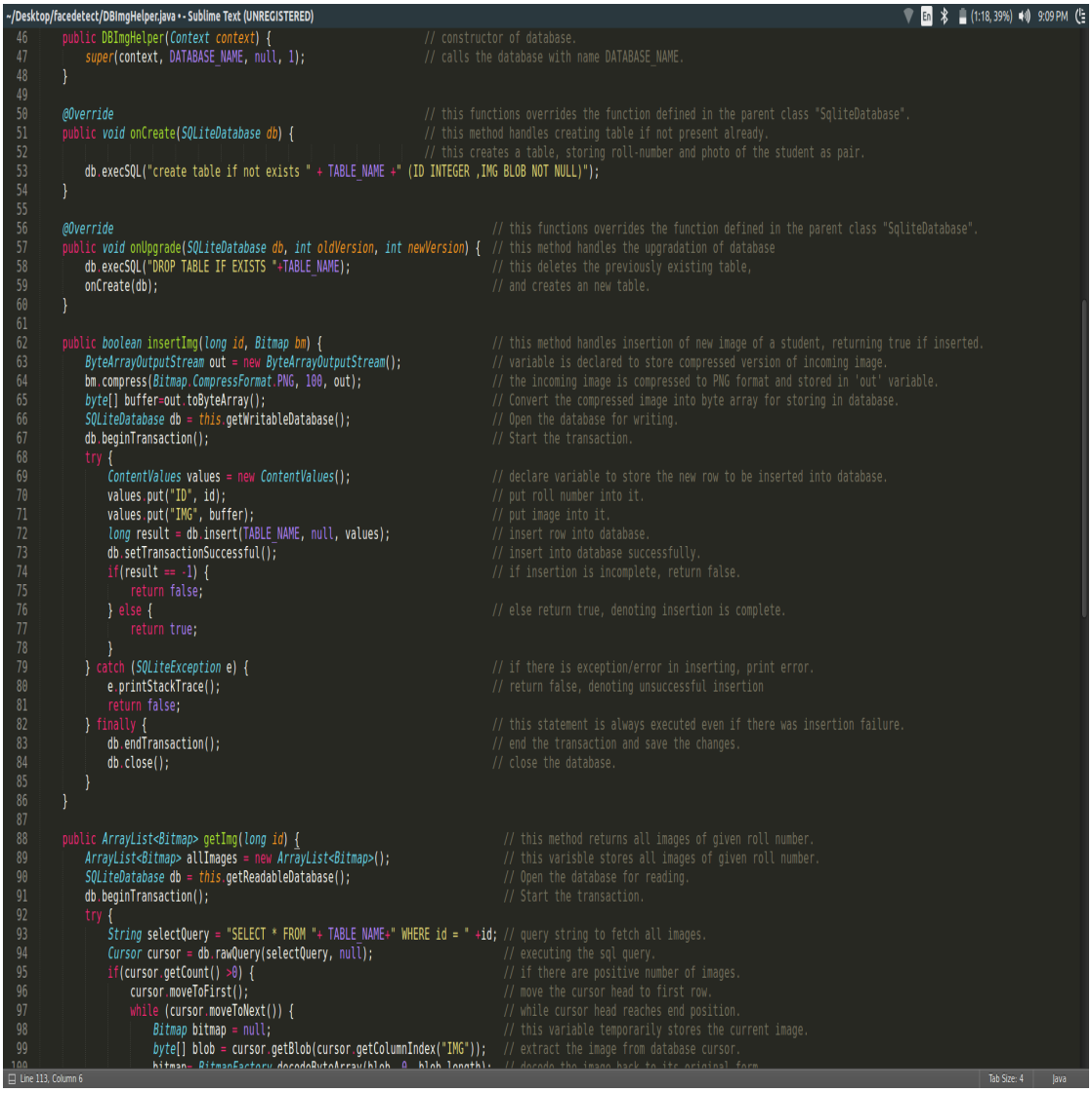
```
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help
    public Cursor getCourseData(String CourseID) {                    // this method returns all data where course id is same as input parameter
        SQLiteDatabase db = this.getWritableDatabase();               // Open the database for reading.
                                                                      // executes the comand to fetch all data from table where course id is same as input parameter
        Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where COURSE='"+CourseID+"'",null);
        return res;                                                   // return the cursor containing the required data
    }

    public Boolean ifStudentExists(long id, String courseID) {        // this method handles the presence of student in particular course
        SQLiteDatabase db = this.getWritableDatabase();               // Open the database for reading.
                                                                      // get list of all students in input course
        Cursor res = db.rawQuery("select * from "+TABLE_NAME+" where COURSE='"+courseID+"'",null);
        if(res.getCount()>0) {                                        // if the course exists in database
            res.moveToFirst();                                        // iterate over all students
            do {
                String rollID = res.getString(0);                     // extract the roll number of the student

                if (rollID.equals(String.valueOf(id))) {              // if they are equal, return true
                    return Boolean.TRUE;
                }
            } while (res.moveToNext());
        }
        return Boolean.FALSE;                                         // if no student roll number matches, return false.
    }

    public boolean updateData(long id,String name,String surname,String course) { // handles updating of data in table
        SQLiteDatabase db = this.getWritableDatabase();               // Open the database for reading.
        ContentValues contentValues = new ContentValues();            // declare variable to store the new row to be inserted into database.
        contentValues.put(COL_2,name);                                // put name into it
        contentValues.put(COL_3,surname);                             // put surname into it
        contentValues.put(COL_4,course);                              // put course id into it
                                                                      // declare a varibale specifying the format of time to be used
        SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss", Locale.getDefault());
        Date date = new Date();                                       // declare variable for storing time
        String modifiedTime = dateFormat.format(date);                // fetch the system time
        contentValues.put(COL_5, modifiedTime);                       // put last-appear time into it
        Random boxColor = new Random();                               // variable for generating random numbers
        contentValues.put(COL_6, (boxColor.nextInt(9)+1));            // generate randowm number between 1 and 10 and put boxcolor into it
        db.update(TABLE_NAME, contentValues, "ID = " +id,null );      // update the table for roll number as given with new values as stored
        return true;                                                  // return true denoting update completion
    }

    public boolean updateModifiedTime(long id, int boxColor, String modifiedTime){ // this method is used to modify the boxcolor and last-appear time for given roll id
        SQLiteDatabase db = this.getWritableDatabase();               // Open the database for reading.
        ContentValues contentValues = new ContentValues();            // declare variable to store the new row to be inserted into database.
        contentValues.put(COL_5, modifiedTime);                       // put last-appear time into it
        contentValues.put(COL_6, boxColor);                           // put boxcolor into it
        db.update(TABLE_NAME, contentValues, "ID = "+id,null);        // update the table for roll number as given with new values as stored
        return true;                                                  // return true denoting update completion
    }

    public Integer deleteData (long id) {                 // this method is used to delete data corresponding to given roll number
        SQLiteDatabase db = this.getWritableDatabase();   // Open the database for reading.
        return db.delete(TABLE_NAME, "ID = "+id,null);    // return the number of rows deleted, if roll number is not present, it returns 0, else returns positive value
    }
}
Line 5, Column 26                                                                                                    Tab Size: 4    Java
```

12 Student Image Database

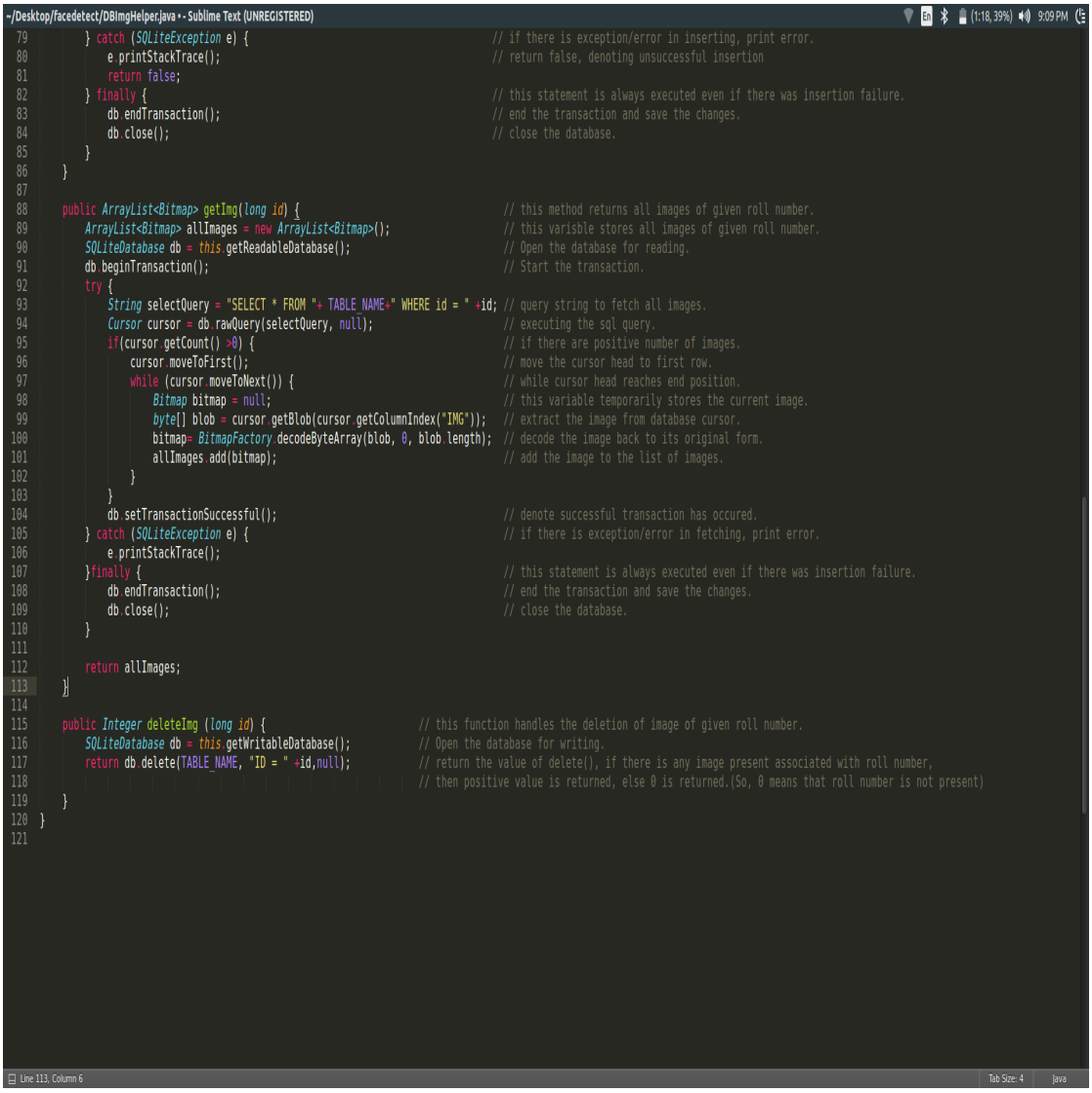
~/Desktop/facedetect/DBImgHelper.java • - Sublime Text (UNREGISTERED)

```
/*
<header>
Module Name         : AddStudentActivity
Date of Creation    : 10-04-2018
Author              : Sujoy Ghosh

Modification History : 11-04-2018 :- added error handling for roll number if it is not an integer.

Synopsis            : This module is responsible for adding new student records.

Global Variables    : None

Functions           : public DBImgHelper(Context)
                      public void onCreate(SQLiteDatabase)
                      public void onUpgrade(SQLiteDatabase, int, int)
                      public boolean insertImg(long, Bitmap)
                      public ArrayList<Bitmap> getImg(long)
                      public boolean updateImg(long, Bitmap)
                      public Integer deleteImg (long)

</header>
*/

package org.opencv.samples.facedetect;

//importing package for android sqlite database
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.util.Log;

import java.io.ByteArrayOutputStream;
import java.util.ArrayList;

public class DBImgHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "Img.db";           // stores name of the database.
    public static final String TABLE_NAME = "img_table";           // stores name of the table.
    public static final String COL_1 = "ID";                       // stores name of first column.
    public static final String COL_2 = "IMG";                      // stores name of second column.

    public DBImgHelper(Context context) {                          // constructor of database.
        super(context, DATABASE_NAME, null, 1);                    // calls the database with name DATABASE_NAME.
    }

    @Override                                                      // this functions overrides the function defined in the parent class "SqliteDatabase".
    public void onCreate(SQLiteDatabase db) {                      // this method handles creating table if not present already.
                                                                   // this creates a table, storing roll-number and photo of the student as pair.
        db.execSQL("create table if not exists " + TABLE_NAME +" (ID INTEGER ,IMG BLOB NOT NULL)");
    }
```

Line 113, Column 6 — Tab Size: 4 — Java




```
46      public DBImgHelper(Context context) {                       // constructor of database.
47          super(context, DATABASE_NAME, null, 1);                 // calls the database with name DATABASE_NAME.
48      }
49
50      @Override                                                   // this functions overrides the function defined in the parent class "SqliteDatabase".
51      public void onCreate(SQLiteDatabase db) {                   // this method handles creating table if not present already.
52                                                                  // this creates a table, storing roll-number and photo of the student as pair.
53          db.execSQL("create table if not exists " + TABLE_NAME +" (ID INTEGER ,IMG BLOB NOT NULL)");
54      }
55
56      @Override                                                   // this functions overrides the function defined in the parent class "SqliteDatabase".
57      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  // this method handles the upgradation of database
58          db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);         // this deletes the previously existing table,
59          onCreate(db);                                           // and creates an new table.
60      }
61
62      public boolean insertImg(long id, Bitmap bm) {              // this method handles insertion of new image of a student, returning true if inserted.
63          ByteArrayOutputStream out = new ByteArrayOutputStream();  // variable is declared to store compressed version of incoming image.
64          bm.compress(Bitmap.CompressFormat.PNG, 100, out);       // the incoming image is compressed to PNG format and stored in 'out' variable.
65          byte[] buffer=out.toByteArray();                        // Convert the compressed image into byte array for storing in database.
66          SQLiteDatabase db = this.getWritableDatabase();         // Open the database for writing.
67          db.beginTransaction();                                  // Start the transaction.
68          try {
69              ContentValues values = new ContentValues();         // declare variable to store the new row to be inserted into database.
70              values.put("ID", id);                               // put roll number into it.
71              values.put("IMG", buffer);                          // put image into it.
72              long result = db.insert(TABLE_NAME, null, values);  // insert row into database.
73              db.setTransactionSuccessful();                      // insert into database successfully.
74              if(result == -1) {                                  // if insertion is incomplete, return false.
75                  return false;
76              } else {                                            // else return true, denoting insertion is complete.
77                  return true;
78              }
79          } catch (SQLiteException e) {                           // if there is exception/error in inserting, print error.
80              e.printStackTrace();                                // return false, denoting unsuccessful insertion
81              return false;
82          } finally {                                             // this statement is always executed even if there was insertion failure.
83              db.endTransaction();                                // end the transaction and save the changes.
84              db.close();                                         // close the database.
85          }
86      }
87
88      public ArrayList<Bitmap> getImg(long id) {                    // this method returns all images of given roll number.
89          ArrayList<Bitmap> allImages = new ArrayList<Bitmap>();    // this varisble stores all images of given roll number.
90          SQLiteDatabase db = this.getReadableDatabase();           // Open the database for reading.
91          db.beginTransaction();                                    // Start the transaction.
92          try {
93              String selectQuery = "SELECT * FROM "+ TABLE_NAME+" WHERE id = " +id; // query string to fetch all images.
94              Cursor cursor = db.rawQuery(selectQuery, null);       // executing the sql query.
95              if(cursor.getCount() >0) {                            // if there are positive number of images.
96                  cursor.moveToFirst();                             // move the cursor head to first row.
97                  while (cursor.moveToNext()) {                     // while cursor head reaches end position.
98                      Bitmap bitmap = null;                         // this variable temporarily stores the current image.
99                      byte[] blob = cursor.getBlob(cursor.getColumnIndex("IMG"));  // extract the image from database cursor.
100                     bitmap= BitmapFactory.decodeByteArray(blob, 0, blob.length);  // decode the image back to its original form
```


~/Desktop/facedetect/DBImgHelper.java • - Sublime Text (UNREGISTERED)
(1:18, 39%)
9:09 PM
} catch (SQLiteException e) { // if there is exception/error in inserting, print error.
e.printStackTrace(); // return false, denoting unsuccessful insertion
return false;
} finally { // this statement is always executed even if there was insertion failure.
db.endTransaction(); // end the transaction and save the changes.
db.close(); // close the database.
}
}
public ArrayList<Bitmap> getImg(long id) { // this method returns all images of given roll number.
ArrayList<Bitmap> allImages = new ArrayList<Bitmap>(); // this varisble stores all images of given roll number.
SQLiteDatabase db = this.getReadableDatabase(); // Open the database for reading.
db.beginTransaction(); // Start the transaction.
try {
String selectQuery = "SELECT * FROM "+ TABLE_NAME+" WHERE id = " +id; // query string to fetch all images.
Cursor cursor = db.rawQuery(selectQuery, null); // executing the sql query.
if(cursor.getCount() >0) { // if there are positive number of images.
cursor.moveToFirst(); // move the cursor head to first row.
while (cursor.moveToNext()) { // while cursor head reaches end position.
Bitmap bitmap = null; // this variable temporarily stores the current image.
byte[] blob = cursor.getBlob(cursor.getColumnIndex("IMG")); // extract the image from database cursor.
bitmap= BitmapFactory.decodeByteArray(blob, 0, blob.length); // decode the image back to its original form.
allImages.add(bitmap); // add the image to the list of images.
}
}
db.setTransactionSuccessful(); // denote successful transaction has occured.
} catch (SQLiteException e) { // if there is exception/error in fetching, print error.
e.printStackTrace();
}finally { // this statement is always executed even if there was insertion failure.
db.endTransaction(); // end the transaction and save the changes.
db.close(); // close the database.
}
return allImages;
}
public Integer deleteImg (long id) { // this function handles the deletion of image of given roll number.
SQLiteDatabase db = this.getWritableDatabase(); // Open the database for writing.
return db.delete(TABLE_NAME, "ID = " +id,null); // return the value of delete(), if there is any image present associated with roll number,
// then positive value is returned, else 0 is returned.(So, 0 means that roll number is not present)
}
}
Line 113, Column 6
Tab Size: 4
Java