Table of Contents

How to Perform CRUD Operations in Room Database in Android?

Last Updated : 28 Nov, 2021

Data from the app can be saved on users' devices in different ways. We can store data in the user's device in SQLite tables, shared preferences, and many more ways. In this article, we will take a look at **saving data, reading, updating, and deleting data in Room Database** on Android. We will perform CRUD operations using Room Database on Android. In this article, we will take a look at performing CRUD operations in Room Database in Android.

What we are going to build in this article?

We will be building a simple application in which we will be adding the different types of courses that are available on Geeks for Geeks. We will be storing all this data in Room Database and performing CRUD operations on these courses. A sample video is given below to get an idea about what we are going to do in this article. Note that we are going to implement this project using the **Java** language.

What is Room?

Room is a persistence library that provides an abstraction

Components of Room

The three main components of the room are **Entity, Database, and DAO**.

- **Entity**: Entity is a modal class that is annotated with @Entity. This class is having variables that will be our columns and the class is our table.
- **Database**: It is an abstract class where we will be storing all our database entries which we can call Entities.
- **DAO**: The full form of DAO is a **Database access object** which is an interface class with the help of it we can perform different operations in our database.

Now, let's move towards the implementation of Room Database in Android.

Step by Step Implementation

Step 1: Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select **Java** as the programming language.

Step 2: Adding dependency for using Room in build.gradle files

Navigate to the **app > Gradle Scripts > build.gradle** file and add the below dependencies in the dependencies section.

```
// add below dependency for using room.
implementation 'androidx.room:room-runtime:2.2.5'
annotationProcessor 'androidx.room:room-compiler:2.2.5'
// add below dependency for using lifecycle extensions for room.
implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
annotationProcessor 'androidx.lifecycle:lifecycle-compiler:2.2.0'
```

After adding the above dependencies section. Now sync your project and we will move towards our XML file.

Step 3: Working with the activity_main.xml file

Navigate to the **app > res > layout > activity_main.xml** and add the below code to that file. Below is the code for the **activity_main.xml** file.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/
    xmlns:app="http://schemas.android.com/apk/
    xmlns:tools="http://schemas.android.com/to
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!--recycler view to display our data-->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/idRVCourses"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!--fab to add new courses-->
    <com.google.android.material.floatingactio
        android:id="@+id/idFABAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true
        android:layout_marginStart="18dp"
        android:layout_marginTop="18dp"
        android:layout_marginEnd="18dp"
        android:layout_marginBottom="18dp"
        android:src="@android:drawable/ic_inpu
        app:tint="@color/white" />

</RelativeLayout>
```

Step 4: Creating a modal class for storing our data

Navigate to the **app > java > your apps package name > Right-click on it > New > Java class** and name the class as **CourseModal** and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import androidx.room.Entity;
import androidx.room.PrimaryKey;

// below line is for setting table name.
@Entity(tableName = "course_table")
public class CourseModal {

    // below line is to auto increment
    // id for each course.
    @PrimaryKey(autoGenerate = true)

    // variable for our id.
    private int id;

    // below line is a variable
```

```
    // for course name.
    private String courseName;

    // below line is use for
    // course description.
    private String courseDescription;

    // below line is use
    // for course duration.
    private String courseDuration;

    // below line we are creating constructor
    // inside constructor class we are not pas
    // our id because it is incrementing autom
    public CourseModal(String courseName, Stri
        this.courseName = courseName;
        this.courseDescription = courseDescrip
        this.courseDuration = courseDuration;
    }

    // on below line we are creating
    // getter and setter methods.
    public String getCourseName() {
        return courseName;
    }

    public void setCourseName(String courseNam
        this.courseName = courseName;
    }

    public String getCourseDescription() {
        return courseDescription;
    }

    public void setCourseDescription(String cc
        this.courseDescription = courseDescrip
    }

    public String getCourseDuration() {
        return courseDuration;
    }

    public void setCourseDuration(String cours
        this.courseDuration = courseDuration;
    }

    public int getId() {
        return id;
    }

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

Step 5: Creating a Dao interface for our database

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name as **Dao** and select Interface. After creating an interface class and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import androidx.lifecycle.LiveData;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Update;

import java.util.List;

// Adding annotation
// to our Dao class
@androidx.room.Dao
public interface Dao {

    // below method is use to
    // add data to database.
    @Insert
    void insert(CourseModal model);

    // below method is use to update
    // the data in our database.
    @Update
    void update(CourseModal model);

    // below line is use to delete a
    // specific course in our database.
    @Delete
    void delete(CourseModal model);

    // on below line we are making query to
    // delete all courses from our database.
    @Query("DELETE FROM course_table")
    void deleteAllCourses();

    // below line is to read all the courses f
    // in this we are ordering our courses in
    // with our course name.
    @Query("SELECT * FROM course_table ORDER B
    LiveData<List<CourseModal>> getAllCourses(
}
```

Step 6: Creating a database class

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name it as **CourseDatabase** and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import android.content.Context;
import android.os.AsyncTask;

import androidx.annotation.NonNull;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.sqlite.db.SupportSQLiteDatabas

// adding annotation for our database entities
@Database(entities = {CourseModal.class}, vers
public abstract class CourseDatabase extends R

    // below line is to create instance
```

```
    // for our database class.
    private static CourseDatabase instance;

    // below line is to create
    // abstract variable for dao.
    public abstract Dao Dao();

    // on below line we are getting instance f
    public static synchronized CourseDatabase
        // below line is to check if
        // the instance is null or not.
        if (instance == null) {
            // if the instance is null we
            // are creating a new instance
            instance =
                    // for creating a instance
                    // we are creating a datab
                    // our database class with
                    Room.databaseBuilder(conte
                            CourseDatabase.cla
                            // below line is u
                            // destructive mig
                            .fallbackToDestruc
                            // below line is t
                            // to our database
                            .addCallback(roomC
                            // below line is t
                            // build our datab
                            .build();
        }
        // after creating an instance
        // we are returning our instance
        return instance;
    }

    // below line is to create a callback for
    private static RoomDatabase.Callback roomC
        @Override
        public void onCreate(@NonNull SupportS
            super.onCreate(db);
            // this method is called when data
            // and below line is to populate c
            new PopulateDbAsyncTask(instance).
        }
    };

    // we are creating an async task class to
    private static class PopulateDbAsyncTask e
        PopulateDbAsyncTask(CourseDatabase ins
            Dao dao = instance.Dao();
        }
        @Override
        protected Void doInBackground(Void...
            return null;
        }
    }
}
```

Step 7: Create a new java class for our Repository

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name it as **CourseRepository** and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```java
import android.app.Application;
import android.os.AsyncTask;

import androidx.lifecycle.LiveData;

import java.util.List;

public class CourseRepository {

    // below line is the create a variable
    // for dao and list for all courses.
    private Dao dao;
    private LiveData<List<CourseModal>> allCou

    // creating a constructor for our variable
    // and passing the variables to it.
    public CourseRepository(Application applic
        CourseDatabase database = CourseDataba
        dao = database.Dao();
        allCourses = dao.getAllCourses();
    }

    // creating a method to insert the data to
    public void insert(CourseModal model) {
        new InsertCourseAsyncTask(dao).execute
    }

    // creating a method to update data in dat
    public void update(CourseModal model) {
        new UpdateCourseAsyncTask(dao).execute
    }

    // creating a method to delete the data in
    public void delete(CourseModal model) {
        new DeleteCourseAsyncTask(dao).execute
    }

    // below is the method to delete all the c
    public void deleteAllCourses() {
        new DeleteAllCoursesAsyncTask(dao).exe
    }

    // below method is to read all the courses
    public LiveData<List<CourseModal>> getAllC
        return allCourses;
    }

    // we are creating a async task method to
    private static class InsertCourseAsyncTask
        private Dao dao;

        private InsertCourseAsyncTask(Dao dao)
            this.dao = dao;
        }

        @Override
        protected Void doInBackground(CourseMo
            // below line is use to insert our
            dao.insert(model[0]);
            return null;
        }
    }

    // we are creating a async task method to
    private static class UpdateCourseAsyncTask
        private Dao dao;

        private UpdateCourseAsyncTask(Dao dao)
```

```
            this.dao = dao;
        }

        @Override
        protected Void doInBackground(CourseMo
            // below line is use to update
            // our modal in dao.
            dao.update(models[0]);
            return null;
        }
    }

    // we are creating a async task method to
    private static class DeleteCourseAsyncTask
        private Dao dao;

        private DeleteCourseAsyncTask(Dao dao)
            this.dao = dao;
        }

        @Override
        protected Void doInBackground(CourseMo
            // below line is use to delete
            // our course modal in dao.
            dao.delete(models[0]);
            return null;
        }
    }

    // we are creating a async task method to
    private static class DeleteAllCoursesAsync
        private Dao dao;
        private DeleteAllCoursesAsyncTask(Dao
            this.dao = dao;
        }
        @Override
        protected Void doInBackground(Void...
            // on below line calling method
            // to delete all courses.
            dao.deleteAllCourses();
            return null;
        }
    }
}
```

Step 8: Creating a class for our Repository

Navigate to the **app > java > your app's package name > Right-click on it > New > Java Class** and name the class as **ViewModal** and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import android.app.Application;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import java.util.List;

public class ViewModal extends AndroidViewMode

    // creating a new variable for course repo
    private CourseRepository repository;

    // below line is to create a variable for
    // data where all the courses are present.
    private LiveData<List<CourseModal>> allCou

    // constructor for our view modal.
    public ViewModal(@NonNull Application appl
        super(application);
        repository = new CourseRepository(appl
        allCourses = repository.getAllCourses(
    }

    // below method is use to insert the data
    public void insert(CourseModal model) {
        repository.insert(model);
    }

    // below line is to update data in our rep
    public void update(CourseModal model) {
        repository.update(model);
    }

    // below line is to delete the data in our
    public void delete(CourseModal model) {
        repository.delete(model);
    }

    // below method is to delete all the cours
    public void deleteAllCourses() {
        repository.deleteAllCourses();
    }

    // below method is to get all the courses
    public LiveData<List<CourseModal>> getAllC
        return allCourses;
    }
}
```

Step 9: Creating a layout file for each item of RecyclerView

Navigate to the **app > res > layout > Right-click on it > New > layout resource file** and name it as **course_rv_item** and add below code to it. Comments are added in the code to get to know in more detail.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/
    xmlns:app="http://schemas.android.com/apk/
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:elevation="8dp"
    app:cardCornerRadius="8dp">

    <LinearLayout
        android:id="@+id/idLLCourse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:orientation="vertical">

        <!--text view for our course name-->
        <TextView
            android:id="@+id/idTVCourseName"
            android:layout_width="match_parent
            android:layout_height="wrap_conten
            android:padding="8dp"
            android:text="Course Name"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <!--text view for our course duration-
        <TextView
            android:id="@+id/idTVCourseDuratic
            android:layout_width="match_parent
            android:layout_height="wrap_conten
            android:padding="8dp"
            android:text="Course Duration"
            android:textColor="@color/black"
            android:textSize="15sp" />

        <!--text view for our course descripti
        <TextView
            android:id="@+id/idTVCourseDescrip
            android:layout_width="match_parent
            android:layout_height="wrap_conten
            android:padding="8dp"
            android:text="Course Description"
            android:textColor="@color/black"
            android:textSize="15sp" />
    </LinearLayout>

</androidx.cardview.widget.CardView>
```

Step 10: Creating a RecyclerView Adapter class to set data for each item of RecyclerView

Navigate to the **app > java > your app's package name > Right-click on it > New > Java class** and name it as **CourseRVAdapter** and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.DiffUtil;
import androidx.recyclerview.widget.ListAdapte
import androidx.recyclerview.widget.RecyclerVi

public class CourseRVAdapter extends ListAdapt

    // creating a variable for on item click l
    private OnItemClickListener listener;

    // creating a constructor class for our ad
    CourseRVAdapter() {
        super(DIFF_CALLBACK);
    }

    // creating a call back for item of recycl
    private static final DiffUtil.ItemCallback
        @Override
        public boolean areItemsTheSame(CourseM
            return oldItem.getId() == newItem.
        }

        @Override
        public boolean areContentsTheSame(Cour
            // below line is to check the cour
            return oldItem.getCourseName().equ
                    oldItem.getCourseDescripti
                    oldItem.getCourseDuration(
        }
    };

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonN
        // below line is use to inflate our la
        // file for each item of our recycler
        View item = LayoutInflater.from(parent
                .inflate(R.layout.course_rv_it
        return new ViewHolder(item);
    }

    @Override
    public void onBindViewHolder(@NonNull View
        // below line of code is use to set da
        // each item of our recycler view.
        CourseModal model = getCourseAt(positi
        holder.courseNameTV.setText(model.getC
        holder.courseDescTV.setText(model.getC
        holder.courseDurationTV.setText(model.
    }

    // creating a method to get course modal f
    public CourseModal getCourseAt(int positic
        return getItem(position);
    }

    public class ViewHolder extends RecyclerVi
        // view holder class to create a varia
        TextView courseNameTV, courseDescTV, c

        ViewHolder(@NonNull View itemView) {
            super(itemView);
            // initializing each view of our r
            courseNameTV = itemView.findViewBy
            courseDescTV = itemView.findViewBy
            courseDurationTV = itemView.findVi

            // adding on click listener for ea
            itemView.setOnClickListener(new Vi
```

```
                @Override
                public void onClick(View v) {
                    // inside on click listene
                    // position to our item of
                    int position = getAdapterP
                    if (listener != null && pc
                        listener.onItemClick(g
                    }
                }
            });
        }
    }

    public interface OnItemClickListener {
        void onItemClick(CourseModal model);
    }
    public void setOnItemClickListener(OnItemC
        this.listener = listener;
    }
}
```

Step 11: Creating a new Activity for Adding and Updating our Course

Navigate to the **app > java > your app's package name > Right-click on it > New > Empty Activity** and name it as **NewCourseActivity** and go to XML part and add below code to it. Below is the code for the **activity_new_course.xml** file.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/
    xmlns:tools="http://schemas.android.com/to
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".NewCourseActivity">

    <!--edit text for our course name-->
    <EditText
        android:id="@+id/idEdtCourseName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Name" />

    <!--edit text for our course description--
    <EditText
        android:id="@+id/idEdtCourseDescriptic
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Description

    <!--edit text for course description-->
    <EditText
        android:id="@+id/idEdtCourseDuration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Course Duration" />

    <!--button for saving data to room databas
    <Button
        android:id="@+id/idBtnSaveCourse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:padding="5dp"
        android:text="Save your course"
        android:textAllCaps="false" />

</LinearLayout>
```

Step 12: Working with the NewCourseActivity.java file

Navigate to the **app > java > your app's package name > NewCourseActivity.java** file and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivit

public class NewCourseActivity extends AppComp
```

```
// creating a variables for our button and
private EditText courseNameEdt, courseDesc
private Button courseBtn;

// creating a constant string variable for
// course name, description and duration.
public static final String EXTRA_ID = "con
public static final String EXTRA_COURSE_NA
public static final String EXTRA_DESCRIPTI
public static final String EXTRA_DURATION

@Override
protected void onCreate(Bundle savedInstar
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_new_c

    // initializing our variables for each
    courseNameEdt = findViewById(R.id.idEd
    courseDescEdt = findViewById(R.id.idEd
    courseDurationEdt = findViewById(R.id.
    courseBtn = findViewById(R.id.idBtnSav

    // below line is to get intent as we
    // are getting data via an intent.
    Intent intent = getIntent();
    if (intent.hasExtra(EXTRA_ID)) {
        // if we get id for our data then
        // setting values to our edit text
        courseNameEdt.setText(intent.getSt
        courseDescEdt.setText(intent.getSt
        courseDurationEdt.setText(intent.g
    }
    // adding on click listener for our sa
    courseBtn.setOnClickListener(new View.
        @Override
        public void onClick(View v) {
            // getting text value from edi
            // the text fields are empty o
            String courseName = courseName
            String courseDesc = courseDesc
            String courseDuration = course
            if (courseName.isEmpty() || co
                Toast.makeText(NewCourseAc
                return;
            }
            // calling a method to save ou
            saveCourse(courseName, courseD
        }
    });
}

private void saveCourse(String courseName,
    // inside this method we are passing
    // all the data via an intent.
    Intent data = new Intent();

    // in below line we are passing all ou
    data.putExtra(EXTRA_COURSE_NAME, cours
    data.putExtra(EXTRA_DESCRIPTION, cours
    data.putExtra(EXTRA_DURATION, courseDu
    int id = getIntent().getIntExtra(EXTRA
    if (id != -1) {
        // in below line we are passing ou
        data.putExtra(EXTRA_ID, id);
    }

    // at last we are setting result as da
    setResult(RESULT_OK, data);

    // displaying a toast message after ad
    Toast.makeText(this, "Course has been
```

```
    }
}
```

Step 13: Working with the MainActivity.java file

Navigate to the **app > java > your app's package name > MainActivity.java** file and add the below code to it. Comments are added inside the code to understand the code in more detail.

Java

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivit
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.ItemTouchH
import androidx.recyclerview.widget.LinearLayc
import androidx.recyclerview.widget.RecyclerVi

import com.google.android.material.floatingact

import java.util.List;

public class MainActivity extends AppCompatAct

    // creating a variables for our recycler v
    private RecyclerView coursesRV;
    private static final int ADD_COURSE_REQUES
    private static final int EDIT_COURSE_REQUE
    private ViewModal viewmodal;

    @Override
    protected void onCreate(Bundle savedInstar
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main)

        // initializing our variable for our r
        coursesRV = findViewById(R.id.idRVCour
        FloatingActionButton fab = findViewByI

        // adding on click listener for floati
        fab.setOnClickListener(new View.OnClic
            @Override
            public void onClick(View v) {
                // starting a new activity for
                // and passing a constant valu
                Intent intent = new Intent(Mai
                startActivityForResult(intent,
            }
        });

        // setting layout manager to our adapt
        coursesRV.setLayoutManager(new LinearL
        coursesRV.setHasFixedSize(true);

        // initializing adapter for recycler v
        final CourseRVAdapter adapter = new Cc

        // setting adapter class for recycler
```

```
        coursesRV.setAdapter(adapter);

        // passing a data from view modal.
        viewmodal = ViewModelProviders.of(this

        // below line is use to get all the co
        viewmodal.getAllCourses().observe(this
            @Override
            public void onChanged(List<CourseM
                // when the data is changed in
                // adding that list to our ada
                adapter.submitList(models);
            }
        });
        // below method is use to add swipe to
        new ItemTouchHelper(new ItemTouchHelpe
            @Override
            public boolean onMove(@NonNull Rec
                return false;
            }

            @Override
            public void onSwiped(@NonNull Recy
                // on recycler view item swipe
                viewmodal.delete(adapter.getCo
                Toast.makeText(MainActivity.th
            }
        }).
                // below line is use to attach
                        attachToRecyclerView(c
        // below line is use to set item click
        adapter.setOnItemClickListener(new Cou
            @Override
            public void onItemClick(CourseModa
                // after clicking on item of r
                // we are opening a new activi
                // a data to our activity.
                Intent intent = new Intent(Mai
                intent.putExtra(NewCourseActiv
                intent.putExtra(NewCourseActiv
                intent.putExtra(NewCourseActiv
                intent.putExtra(NewCourseActiv

                // below line is to start a ne
                // adding a edit course consta
                startActivityForResult(intent,
            }
        });
    }

    @Override
    protected void onActivityResult(int reques
        super.onActivityResult(requestCode, re
        if (requestCode == ADD_COURSE_REQUEST
            String courseName = data.getString
            String courseDescription = data.ge
            String courseDuration = data.getSt
            CourseModal model = new CourseModa
            viewmodal.insert(model);
            Toast.makeText(this, "Course saved
        } else if (requestCode == EDIT_COURSE_
            int id = data.getIntExtra(NewCours
            if (id == -1) {
                Toast.makeText(this, "Course c
                return;
            }
            String courseName = data.getString
            String courseDesc = data.getString
            String courseDuration = data.getSt
            CourseModal model = new CourseModa
            model.setId(id);
```

```
                viewmodal.update(model);
                Toast.makeText(this, "Course updat
            } else {
                Toast.makeText(this, "Course not s
            }
        }
    }
```

Now run your app and see the output of the app.

Output:

Check out the project on the below link:
https://github.com/ChaitanyaMunje/GFG-Room-Database

Like 6

Next

How to Use Picasso Image Loader Library in Android?

RECOMMENDED ARTICLES Page : 1 2 3

Article Contributed By :

chaitanyamunje
@chaitanyamunje

Vote for difficulty

Easy | Normal | Medium | Hard | Expert

Improved By : kapoorsagar226

Article Tags : android, Technical Scripter 2020, Android, Java, Technical Scripter

Practice Tags : Java, Android

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org