**PRACTICAL # 12**

**OBJECT:**

Performing background operations

**THEORY:**

Android OS allows applications to perform time consuming tasks in background. This allows the main UI thread to respond to user even when a process is running. To perform a background operation on a thread in Android, use AsyncTask class. AsyncTask facilitates you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. AsyncTask

needs to be inherited to be implemented. The subclass will override the required methods to perform the background tasks.

AsyncTask's generic types

The three types used by an asynchronous task are the following:

Params: type of the parameters sent to the task upon execution.

Progress: type of the progress units published during the background computation.

Result: type of the result of the background computation.

Not all types are used by an asynchronous task. To mark a type as unused, simply use the type Void:

private class MyTask extends AsyncTask<Void, Void, Void> { ... }

AsyncTask performs execution in 4 steps:

**onPreExecute():** invoked on the UI thread before the task is executed. This is used to setup the task, like showing a progress bar.

**doInBackground(Params…):** invoked on the background thread immediately after onPreExecute() finishes executing. Here background computation is performed that can take a long time. The parameters of the asynchronous task are passed in this method. The result of the computation returned from this method are passed back to the last step. This step can also use publishProgress(Progress...) to publish one or more units of progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.

**onProgressUpdate(Progress…):** invoked on the UI thread after a call to publishProgress(Progress...). This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.

**onPostExecute(Result):** invoked on the UI thread after the background computation finishes. The result of the computation or task is passed to this step as a parameter.

AsyncTask class example:

*private class DownloadAsyncTask extends AsyncTask<String, String, Bitmap>{*

*private ProgressDialog progressDialog;*

*protected void onPreExecute() {*

*super.onPreExecute();*

*progressDialog = new ProgressDialog(MainActivity.this);*

*progressDialog.setMessage("Downloading...");*

*progressDialog.setCancelable(false);*

*progressDialog.show();*

*}*

*@Override*

*protected Bitmap doInBackground(String... strings) {*

*InputStream inputStream;*

*Bitmap bitmapImage = null;*

*try {*

*URL ImageUrl = new URL(strings[0]);*

*HttpURLConnection conn = (HttpURLConnection)*

*ImageUrl.openConnection();*

*conn.setDoInput(true);*

*conn.connect();*

*inputStream = conn.getInputStream();*

*BitmapFactory.Options options = new BitmapFactory.Options();*

*options.inPreferredConfig = Bitmap.Config.RGB\_565;*

*bitmapImage = BitmapFactory.decodeStream(inputStream, null, options);*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*return bitmapImage;*

*}*

*@Override*

*protected void onPostExecute(Bitmap bitmap) {*

*super.onPostExecute(bitmap);*

*progressDialog.hide();*

*mImageView.setImageBitmap(bitmap);*

*showNotification();*

*}*

*}*

After defining the class, use the class instance as:

*DownloadAsyncTask asyncTask=new DownloadAsyncTask();*

*asyncTask.execute("https://homepages.cae.wisc.edu/~ece533/images/fruits.png");*

Here the inherited AsyncTask class is instantiated. The execute method takes the URL of image file to download. It downloads the contents of the file in background and sets the imageView source after image is downloaded.

**ACTIVITIES**

**Activity 1**

Create a new project to implement the above application that downloads and shows image in main activity.

**REVIEW QUESTIONS**

1. What is the importance of background thread?
2. Which class is used to perform tasks in background?
3. What is the purpose of **onPostExecute** method?