Java Callbacks and AsyncTask

CS 371M Mobile Computing
Thanks to Android Documentation
Define an interface, implements

```java
public interface FetchCallback {
    void fetchStart();
    void fetchComplete(FetchRecord fetchRecord);
    void fetchCancel(FetchRecord fetchRecord);
}
```
Receiver implements, caller passed this

```java
public class GetSearch extends AppCompatActivity implements FetchCallback {

    public void fetchStart() {
    }
    public void fetchComplete(FetchRecord fetchRecord){
    }
    public void fetchCancel(FetchRecord fetchRecord) {
    }

    public class RedditFetch {
        protected FetchCallback fetchCallback = null;
        public RedditFetch(FetchCallback fetchCallback) {
            this.fetchCallback = fetchCallback;
        }
        void doingSomething() {
            fetchCallback.fetchStart();
        }
    }
}
```
AsyncTask

• `onPreExecute()`, invoked on the UI thread before the task is executed.
  • E.g., show progress bar.

• `doInBackground(Params...)`, invoked on the background thread immediately after `onPreExecute()` finishes executing.
  • The parameters of the asynchronous task are passed to this step.
  • The result is returned and passed back to the last step.
  • This step can also use `publishProgress(Progress...)`.
  • `publishProgress` values are published on the UI thread, in `onProgressUpdate(Progress...)` step.

• `onProgressUpdate(Progress...)`, invoked on UI thread after `publishProgress(Progress...)`.
  • The timing of the execution is undefined.
  • Can be used to animate a progress bar or show logs in a text field.

• `onPostExecute(Result)`, invoked on UI thread after background computation finishes.
  • The result of the background computation is passed to this step as a parameter.
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }
    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }
    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}