PART-B ESSAY QUESTIONS WITH SOLUTIONS

4.1 USING APPLICATION SPECIFIC FOLDERS AND FILES

Q9. Define file. Discuss the usage of application specific folders and files.

Answer :

Model Paper-I, Q8

File

A file is a collection of records where each record provides an information to the user. The files are stored permanently on the secondary storage device and can be accessed through a set of library functions.

Most of the applications create or download the files that are specific to application. The application specific files can be stored either internally or externally. While referring to external storage, the shared/media storage which is accessible by all the applications and can be mounted to computer file system when device is connected through USB. Eventhough it is located on SD card certain devices will implement it as a separate part on internal storage.

Security is not imposed on the files which are stored on external storage. These files can be accessed, overwritten or deleted by any application. The files stored on external storage might not always be available. The application cannot read files on external storage when the SD card is ejected or device is mounted through computer.

Android provides two methods through application context. They are as follows,

1. **getDir( )**

   This method is used to return the file path that has path to the internal application file storage directory.

2. **getExternalFilesDir( )**

   This method is used to return the file path that has path to the external application file storage directory.

The getExternalFilesDir method was introduced in Android API. The Environment.getExternalStorageDirectory( ) must be called to return a path to root of external storage.

The methods getDir and getExternalFilesDir accept a string argument to specify subdirectory in which the files need to be stored. The files that are stored in application folders must be specific to parent application and are even not detected by media-scanner. Thus, they cannot be added to Media Library automatically. If files are created or application is downloaded that need to be added to Media Library or to be made available to other applications then move them to public external storage directory.

4.2 CREATING FILES, READING DATA FROM FILES, LISTING CONTENTS OF A DIRECTORY

Q10. Write short notes on the following,

(i) Creating files

(ii) Reading and writing data from and to files.

Answer :

Model Paper-III, Q8

(i) Creating Files

In android, a file can be created by using openFileOutput( ) method.

Syntax

```
String FILE_NAME = "file1.tmp";
FileOutputStream os = openFileOutput(FILE_NAME, context.MODE_PRIVATE);
FileInputStream is = openFileInput (FILE_NAME);
```

The methods used in above code support the files of present application folder. Specifying path separators makes exception to be thrown when specified file name does not exist while creating FileOutputStream then Android creates it. Overwriting the existing files is the default behaviour of them. In order to append data to the existing file, the mode must be specified as context. MODE_APPEND. The files that are created by using openFileOutput method are private by default to the calling application. The file can be shared in between application by using the content provider. The other method is to specify context.MODE_WORLD_READABLE or context.MODE_WORLD_WRITABLE while creating output file in order to make it available to other applications.

```
String outputfile = "public.txt";
FileOutputStream os = openFileOutput(OUTPUT_FILE context.MODE_WORLD_WRITABLE);
```

The location of files stored in sandbox can be known by calling getFilesDir. The method returns absolute path to files which are created by using openFileOutput.

Syntax

```
File f = getFilesDir( );
Log.d("OUTPUT_PATH_", file.getAbsolutesPath( ));
```

(ii) Reading and Writing Data from and to Files

In android, the data can be read and written by using openFileInput( ) and openFileOutput( ) respectively. The method openFileOutput( ) is used to create and save a file. After this data can be written to the file. Whereas, the method openFileInput( ) method is used to open a file for reading. It returns an instance of FileInputStream.

```
FileInputStream is = openFileInput(file);
```

Example

A simple example to read one character from a file is as given below,

```
int x;
String tmp = " ";
while((x = is.read( ))! = – 1)
{
    tmp = tmp + Character.toString ((char)x);
}
is.close( );    //where 'is' is the instance of FileInputStream.
```

Q11. How contents of a directory are listed?

Model Paper-II, Q8

Answer :

The files or contents of a directory can be displayed by using "ls" command. The ls command writes the contents of specified directory or name of specified file to the standard output. It lists the contents of present directory if a particular file or directory is not specified. It displays all the contents in alphabetical order of filename.

The ls Command with Options

The various options of 'ls' command are as follows,

Option	Description
-A	It displays all the contents excluding the hidden folders or files (i.e., dot(.) files and dot dot(..) files)
-a	It displays all the files along with the hidden files
-1	It displays one file per line
-r	It displays the files in the reverse order
-m	It displays the files as a comma-separated list
-d	It displays only directories
-t	It displays the files based on the time modified

The first character of every entry might be d(directory), b(block special file), c(character special file), l(symbolic link), p(first-in, first-out pipe special file), s(local socket) etc. The next nine characters are divided into three sets. The first set depicts the file or directory owner's permission, the second set depicts the permission of remaining users in a group. And, the last set depicts the permission of others who can access the file. The characters of each of these sets represent the read (r), write (w) and execute (x) permissions of file. The detailed information can be displayed by using the below command,

```
ls -l file1.profile
```

The detailed information of a directory can be displayed by using below command,

```
ls -d -l.manual manual/file1
```

4.3 SHARED PREFERENCES – CREATING SHARED PREFERENCES, SAVING AND RETRIEVING DATA USING SHARED PREFERENCE

Q12. What are shared preferences? How are they created?

(Model Paper-I, Q9 | Model Paper-III, Q9)

Answer :

Shared Preferences

Shared preferences are used to store the data of various primitive applications using their easy, light weight NVP (name/value pair) mechanism. They are required when the user wants to store UI state, user preferences or application settings. They allow the users to save sets of name/value pairs of primitive data as named preferences. The various primitive types supported by shared preferences include boolean, string, float, long and integer by making them ideal means to store current UI state, user preferences, default values and class instance variables.

Creating Shared Preferences

A shared preference can be created by calling getSharedPreferences( ) on current context, by passing name as an argument.

```
SharedPreferences sp1 = getSharedPreferences (PREF1, Activity.MODE_PRIVATE);
```

A shared preference can be modified by using SharedPreferences.Editor class. The editor object can be created by calling edit( ) on shared preferences object that is to be modified.

```
SharedPreferences.Editor ed1 = sp1.edit( );
```

The put<type> methods are used for inserting or updating the values accompanied with the given name,

Examples

```
editor.putBoolean("b1", true);
editor.putFloat("f", 1f);
editor.putInt("d", 2);
editor.putLong("l", 26);
editor.putString("textEntryval", "Not Empty");
```

Q13. Explain how shared preferences are used to save and retrieve the data.

Model Paper-II, Q9

Answer :

Saving Shared Preferences

The shared preferences are saved in sandbox of the application. Thus, they can be easily shared by components of the applications. Note that, the shared preferences of one application cannot be accessed by other applications.

The various modifications done on the editor object can be saved by using the apply or commit methods. Here, the apply( ) method is used to save the modifications asynchronously and commit( ) method is used to save the modifications synchronously.

Syntax

```
ed1.apply( );      //saving modifications asynchronously
ed1.commit( );     //saving modifications synchronously
```

The apply( ) method is most preferred method to save shared preferences since it is asynchronous. The commit( ) method can be called when confirmation to success is required. This method blocks the calling thread and returns true in case of successful write otherwise false is returned.

Retrieving Shared Preferences

The shared preferences can be retrieved or accessed by using getSharedPreferences( ) method. The saved values can be retrieved by using type-safe get <type> method. Every getter accepts a key and default value as arguments.

Syntax

```
SharedPreferences sp = getSharedPreferences("Settings", context.MODE_PRIVATE);
Now, this can be accessed by using the below statement,
String settings = SharedPreferences.getString("KeyName", "defaultValue");
boolean b1 = mySharedPreferences.getBoolean ("b1", false);
float f = mySharedPreferences.getFloat("f", 0f);
int d = mySharedPreferences.getInt("d", 1);
long l = mySharedPreferences.getLong("l", 0);
String sp = mySharedPreferences.getString("textEntryVal", " ");
```

A map of the available shared preferences key values can be obtained by calling getAll( ) method and the availability of a specific key can be determined by calling contains( ) method.

Syntax

```
Map<String, ?> allPreferences = mySharedPreferences.getAll( );
boolean clf = mySharedPreferences.contain("lf");
```

IMPORTANT QUESTIONS

SHORT QUESTIONS

Q1. What is persistent storage?

Ans: For answer refer Unit-IV, Q1. Important Question

Q2. Define shared preferences.

Ans: For answer refer Unit-IV, Q5. Important Question

Q3. Write about creating shared preferences.

Ans: For answer refer Unit-IV, Q6. Important Question

ESSAY QUESTIONS

Q4. Define file. Discuss the usage of application specific folders and files.

Ans: For answer refer Unit-IV, Q9. Important Question

Q5. Write short notes on the following,

(i) Creating files

(ii) Reading and writing data from and to files.

Ans: For answer refer Unit-IV, Q10. Important Question

Q6. How contents of a directory are listed?

Ans: For answer refer Unit-IV, Q11. Important Question

Q7. Explain how shared preferences are used to save and retrieve the data.

Ans: For answer refer Unit-IV, Q13. Important Question