**Список файлов в папке**

в манифест надо добавить разрешение

|  | |
| --- | --- |
| |  | | --- | | **<uses-permission** android:name="android.permission.READ\_EXTERNAL\_STORAGE" **/>** | |  |

и код примерно такой

|  | |
| --- | --- |
| |  | | --- | | **File** rootFolder = **Environment**.getExternalStorageDirectory();  **File**[] filesArray = rootFolder.listFiles();  **System**.out.println("файлов: " + filesArray.length);  **for** (**File** f: filesArray) {  **if** (f.isDirectory()) **System**.out.println("Folder: " + f);  **else** **if** (f.isFile()) **System**.out.println("File: " + f);          } | |  |

Вариант 2

public ArrayList<File> listFilesWithSubFolders(File dir) {

ArrayList<File> files = new ArrayList<File>();

for (File file : dir.listFiles()) {

if (file.isDirectory())

files.addAll(listFilesWithSubFolders(file));

else

files.add(file);

}

return files;

}

Вариант 3

Класс Lister выводит отсортированный список содержимого пути, переданного ему в качестве аргумента. Если аргумента нет, выводится сождержимое текущей директории.

1. import java.io.File;
2. import java.io.IOException;
3. import java.util.Arrays;
4. import java.util.Comparator;
6. class Lister {
8. private File path;
9. private File[] files;
11. public Lister(String pathname) throws IOException {
12. path = new File(pathname);
13. if (!path.exists()) {
14. throw new IOException("Cannot access " + pathname + ": No such file or directory");
15. }
16. if (path.isFile()) {
17. files = new File[]{path};
18. } else {
19. files = path.listFiles();
20. Arrays.sort(files, new FilesComparator());
21. }
22. }
24. public void print() {
25. for (File f: files) {
26. System.out.println(
27. f.getName()
28. + ((f.isDirectory()) ? File.separator : "")
29. );
30. }
31. }
33. public static void main(String[] args) {
34. String path = (args.length == 0) ? "." : args[0];
35. try {
36. new Lister(path).print();
37. } catch (Exception e) {
38. e.printStackTrace();
39. }
40. }
42. private class FilesComparator implements Comparator<File> {
43. public int compare(File f1, File f2) {
44. if (f1.isDirectory() && f2.isFile()) {
45. return -1;
46. }
47. if (f1.isFile() && f2.isDirectory()) {
48. return 1;
49. }
50. return f1.compareTo(f2);
51. }
52. }
53. }
54. public class Dir {
55. public static void main(String[] args)
56. {
57. try (DirectoryStream<Path> stream = Files.newDirectoryStream(Paths.get(args[0]))) {
58. for (Path file: stream) {
59. if(!file.toFile().isDirectory() ) {
60. System.out.println(file.getFileName());
61. }
62. }
63. } catch (IOException | DirectoryIteratorException x) {
64. System.err.println(x);
65. }
66. }
67. }