

Selected files

3 printable files

CS5800_Q5\src\com\cpp\cs5800q5\CS5800_Q5.java

CS5800_Q5\src\com\cpp\cs5800q5\File.java

CS5800_Q5\src\com\cpp\cs5800q5\Folder.java

CS5800_Q5\src\com\cpp\cs5800q5\CS5800_Q5.java

```
1 package com.cpp.cs5800q5;
2
3 public class CS5800_Q5 {
4     public static void main(String[] args) {
5         // root folder
6         Folder phpDemo1 = new Folder("php_demo1");
7
8         Folder sourceFiles = new Folder("Source Files");
9         Folder includePath = new Folder("Include Path");
10        Folder remoteFiles = new Folder("Remote Files");
11        phpDemo1.addFolder(sourceFiles);
12        phpDemo1.addFolder(includePath);
13        phpDemo1.addFolder(remoteFiles);
14
15        // inside source files folder
16        Folder phalcon = new Folder(".phalcon");
17        Folder app = new Folder("app");
18        Folder cache = new Folder("cache");
19        Folder publicFolder = new Folder("public");
20        sourceFiles.addFolder(phalcon);
21        sourceFiles.addFolder(app);
22        sourceFiles.addFolder(cache);
23        sourceFiles.addFolder(publicFolder);
24        sourceFiles.addFile(new File(".htaccess"));
25        sourceFiles.addFile(new File("htrouter.php"));
26        sourceFiles.addFile(new File("index.html"));
27
28        // inside app folder
```

```
29     Folder config = new Folder("config");
30     Folder controllers = new Folder("controllers");
31     Folder library = new Folder("library");
32     Folder migrations = new Folder("migrations");
33     Folder models = new Folder("models");
34     Folder views = new Folder("views");
35     app.addFolder(config);
36     app.addFolder(controllers);
37     app.addFolder(library);
38     app.addFolder(migrations);
39     app.addFolder(models);
40     app.addFolder(views);
41
42     System.out.println("Folder structure: ");
43     phpDemo1.print("");
44     System.out.println();
45
46     // Delete the "app" folder
47     System.out.println("Removal of app folder: ");
48     sourceFiles.removeFolder("app");
49     phpDemo1.print("");
50     System.out.println();
51
52     // Delete the "public" folder
53     System.out.println("Removal of public folder: ");
54     sourceFiles.removeFolder("public");
55     phpDemo1.print("");
56     System.out.println();
57
58 }
59
60 }
61
62
```

CS5800_Q5\src\com\cpp\cs5800q5\File.java

```
1 package com.cpp.cs5800q5;
2
```

```
3 public class File {
4     private String name;
5
6     public File(String name) {
7         this.name = name;
8     }
9
10    public String getName() {
11        return name;
12    }
13
14    public void setName(String name) {
15        this.name = name;
16    }
17
18    public void print(String currentFileDirectory) {
19        System.out.println(currentFileDirectory + this.name);
20    }
21
22 }
23
24
```

CS5800_Q5\src\com\cpp\cs5800q5\Folder.java

```
1 package com.cpp.cs5800q5;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Folder {
7
8     private String name;
9     private List<Folder> subFolders;
10    private List<File> files;
11
12    public Folder(String name) {
13        this.name = name;
14        this.subFolders = new ArrayList<>();
15    }
16
17    public void print() {
18        System.out.println("Folder: " + name);
19        for (Folder subFolder : subFolders) {
20            subFolder.print();
21        }
22        for (File file : files) {
23            file.print();
24        }
25    }
26
27    public void addSubFolder(Folder subFolder) {
28        subFolders.add(subFolder);
29    }
30
31    public void addFile(File file) {
32        files.add(file);
33    }
34
35    public void removeSubFolder(Folder subFolder) {
36        subFolders.remove(subFolder);
37    }
38
39    public void removeFile(File file) {
40        files.remove(file);
41    }
42
43    public String getName() {
44        return name;
45    }
46
47    public void setName(String name) {
48        this.name = name;
49    }
50
51    public List<Folder> getSubFolders() {
52        return subFolders;
53    }
54
55    public void setSubFolders(List<Folder> subFolders) {
56        this.subFolders = subFolders;
57    }
58
59    public List<File> getFiles() {
60        return files;
61    }
62
63    public void setFiles(List<File> files) {
64        this.files = files;
65    }
66
67    public boolean isEmpty() {
68        return subFolders.isEmpty() && files.isEmpty();
69    }
70
71    public boolean isNotEmpty() {
72        return !isEmpty();
73    }
74
75    public boolean containsSubFolder(Folder subFolder) {
76        return subFolders.contains(subFolder);
77    }
78
79    public boolean containsFile(File file) {
80        return files.contains(file);
81    }
82
83    public boolean equals(Folder folder) {
84        return name.equals(folder.name) && subFolders.equals(folder.subFolders) && files.equals(folder.files);
85    }
86
87    public boolean hashCode() {
88        return name.hashCode() + subFolders.hashCode() + files.hashCode();
89    }
90
91    public String toString() {
92        return "Folder{" + name + ", subFolders=" + subFolders + ", files=" + files + "}";
93    }
94
95    public static void main(String[] args) {
96        Folder folder = new Folder("Folder");
97        folder.addSubFolder(new Folder("SubFolder"));
98        folder.addFile(new File("File"));
99        folder.print();
100    }
101
102 }
```

```
15     this.files = new ArrayList<>();
16 }
17
18 public String getName() {
19     return name;
20 }
21
22 public void setName(String name) {
23     this.name = name;
24 }
25
26 public List<Folder> getSubFolders() {
27     return subFolders;
28 }
29
30 public void setSubFolders(List<Folder> subFolders) {
31     this.subFolders = subFolders;
32 }
33
34 public List<File> getFiles() {
35     return files;
36 }
37
38 public void setFiles(List<File> files) {
39     this.files = files;
40 }
41
42 public void addFolder(Folder folder) {
43     this.subFolders.add(folder);
44 }
45
46 public void addFile(File file) {
47     this.files.add(file);
48 }
49
50 public void removeFolder(String folderName) {
51     subFolders.removeIf(folder -> folder.getName().equals(folderName));
52 }
```

```
53  
54  
55     public void print(String currentFolderDirectory) {  
56         System.out.println(currentFolderDirectory + " " + name);  
57  
58         for(Folder folder : subFolders) {  
59             folder.print(currentFolderDirectory + "   |");  
60         }  
61  
62         for(File file : files) {  
63             file.print(currentFolderDirectory + "   |" + "");  
64         }  
65     }  
66 }
```