OrFileFilter.java - AndFileFilter.java

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.commons.io.filefilter;

import java.io.File;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

/**
 * A {@link java.io.FileFilter} providing conditional OR logic across a list of
 * file filters. This filter returns <code>true</code> if any filters in the
 * list return <code>true</code>. Otherwise, it returns <code>false</code>.
 * Checking of the file filter list stops when the first filter returns
 * <code>true</code>.
 *
 * @since Commons IO 1.0
 * @version $Revision: 490425 $ $Date: 2006-12-26 17:25:43 -0800 (Tue, 26 Dec
 2006) $
 *
 * @author Steven Caswell
 */
public class OrFileFilter
        extends AbstractFileFilter
        implements ConditionalFileFilter {

    /** The list of file filters. */
    private List fileFilters;

    /**
     * Constructs a new instance of <code>OrFileFilter</code>.
     *
     * @since Commons IO 1.1
     */
    public OrFileFilter() {
        this.fileFilters = new ArrayList();
    }

    /**
     * Constructs a new instance of <code>OrFileFilter</code>
     * with the specified filters.
```

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.apache.commons.io.filefilter;

import java.io.File;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

/**
 * A {@link java.io.FileFilter} providing conditional AND logic across a list of
 * file filters. This filter returns <code>true</code> if all filters in the
 * list return <code>true</code>. Otherwise, it returns <code>false</code>.
 * Checking of the file filter list stops when the first filter returns
 * <code>false</code>.
 *
 * @since Commons IO 1.0
 * @version $Revision: 490425 $ $Date: 2006-12-26 17:25:43 -0800 (Tue, 26 Dec
 2006) $
 *
 * @author Steven Caswell
 */
public class AndFileFilter
        extends AbstractFileFilter
        implements ConditionalFileFilter {

    /** The list of file filters. */
    private List fileFilters;

    /**
     * Constructs a new instance of <code>AndFileFilter</code>.
     *
     * @since Commons IO 1.1
     */
    public AndFileFilter() {
        this.fileFilters = new ArrayList();
    }

    /**
     * Constructs a new instance of <code>AndFileFilter</code>
     * with the specified list of filters.
```

OrFileFilter.java - AndFileFilter.java

```
56      *
57      * @param fileFilters  the file filters for this filter, copied, null ignored
58      * @since Commons IO 1.1
59      */
60     public OrFileFilter(final List fileFilters) {
61         if (fileFilters == null) {
62             this.fileFilters = new ArrayList();
63         } else {
64             this.fileFilters = new ArrayList(fileFilters);
65         }
66     }
67
68     /**
69      * Constructs a new file filter that ORs the result of two other filters.
70      *
71      * @param filter1  the first filter, must not be null
72      * @param filter2  the second filter, must not be null
73      * @throws IllegalArgumentException if either filter is null
74      */
75     public OrFileFilter(IOFileFilter filter1, IOFileFilter filter2) {
76         if (filter1 == null || filter2 == null) {
77             throw new IllegalArgumentException("The filters must not be null");
78         }
79         this.fileFilters = new ArrayList();
80         addFileFilter(filter1);
81         addFileFilter(filter2);
82     }
83
84     /**
85      * {@inheritDoc}
86      */
87     public void addFileFilter(final IOFileFilter ioFileFilter) {
88         this.fileFilters.add(ioFileFilter);
89     }
90
91     /**
92      * {@inheritDoc}
93      */
94     public List getFileFilters() {
95         return Collections.unmodifiableList(this.fileFilters);
96     }
97
98     /**
99      * {@inheritDoc}
100     */
101    public boolean removeFileFilter(IOFileFilter ioFileFilter) {
102        return this.fileFilters.remove(ioFileFilter);
103    }
104
105    /**
106     * {@inheritDoc}
107     */
108    public void setFileFilters(final List fileFilters) {
109        this.fileFilters = fileFilters;
110    }
111
```

```
56      *
57      * @param fileFilters  a List of IOFileFilter instances, copied, null ignored
58      * @since Commons IO 1.1
59      */
60     public AndFileFilter(final List fileFilters) {
61         if (fileFilters == null) {
62             this.fileFilters = new ArrayList();
63         } else {
64             this.fileFilters = new ArrayList(fileFilters);
65         }
66     }
67
68     /**
69      * Constructs a new file filter that ANDs the result of two other filters.
70      *
71      * @param filter1  the first filter, must not be null
72      * @param filter2  the second filter, must not be null
73      * @throws IllegalArgumentException if either filter is null
74      */
75     public AndFileFilter(IOFileFilter filter1, IOFileFilter filter2) {
76         if (filter1 == null || filter2 == null) {
77             throw new IllegalArgumentException("The filters must not be null");
78         }
79         this.fileFilters = new ArrayList();
80         addFileFilter(filter1);
81         addFileFilter(filter2);
82     }
83
84     /**
85      * {@inheritDoc}
86      */
87     public void addFileFilter(final IOFileFilter ioFileFilter) {
88         this.fileFilters.add(ioFileFilter);
89     }
90
91     /**
92      * {@inheritDoc}
93      */
94     public List getFileFilters() {
95         return Collections.unmodifiableList(this.fileFilters);
96     }
97
98     /**
99      * {@inheritDoc}
100     */
101    public boolean removeFileFilter(final IOFileFilter ioFileFilter) {
102        return this.fileFilters.remove(ioFileFilter);
103    }
104
105    /**
106     * {@inheritDoc}
107     */
108    public void setFileFilters(final List fileFilters) {
109        this.fileFilters = new ArrayList(fileFilters);
110    }
111
```

OrFileFilter.java - AndFileFilter.java

```
112     /**
113      * {@inheritDoc}
114      */
115     public boolean accept(final File file) {
116         for (Iterator iter = this.fileFilters.iterator(); iter.hasNext();) {
117             IOFileFilter fileFilter = (IOFileFilter) iter.next();
118             if (fileFilter.accept(file)) {
119                 return true;
120             }
121         }
122         return false;
123     }
124
125     /**
126      * {@inheritDoc}
127      */
128     public boolean accept(final File file, final String name) {
129         for (Iterator iter = this.fileFilters.iterator(); iter.hasNext();) {
130             IOFileFilter fileFilter = (IOFileFilter) iter.next();
131             if (fileFilter.accept(file, name)) {
132                 return true;
133             }
134         }
135         return false;
136     }
137
138}
139
```

```
112     /**
113      * {@inheritDoc}
114      */
115     public boolean accept(final File file) {
116         if (this.fileFilters.size() == 0) {
117             return false;
118         }
119         for (Iterator iter = this.fileFilters.iterator(); iter.hasNext();) {
120             IOFileFilter fileFilter = (IOFileFilter) iter.next();
121             if (!fileFilter.accept(file)) {
122                 return false;
123             }
124         }
125         return true;
126     }
127
128     /**
129      * {@inheritDoc}
130      */
131     public boolean accept(final File file, final String name) {
132         if (this.fileFilters.size() == 0) {
133             return false;
134         }
135         for (Iterator iter = this.fileFilters.iterator(); iter.hasNext();) {
136             IOFileFilter fileFilter = (IOFileFilter) iter.next();
137             if (!fileFilter.accept(file, name)) {
138                 return false;
139             }
140         }
141         return true;
142     }
143
144}
145
```