PrefixFileFilter.java - SuffixFileFilter.java

```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.io.Serializable;
import java.util.List;

import org.apache.commons.io.IOCase;

/**
 * Filters filenames for a certain prefix.
 *
 * <p>
 * For example, to print all files and directories in the
 * current directory whose name starts with <code>Test</code>:
 *
 * <pre>
 * File dir = new File(".");
 * String[] files = dir.list( new PrefixFileFilter("Test") );
 * for ( int i = 0; i &lt; files.length; i++ ) {
 *     System.out.println(files[i]);
 * }
 * </pre>
 *
 * @since Commons IO 1.0
 * @version $Revision: 606381 $ $Date: 2007-12-22 02:03:16 +0000 (Sat, 22 Dec 2007) $
 *
 * @author Stephen Colebourne
 * @author Federico Barbieri
 * @author Serge Knystautas
 * @author Peter Donald
 */
public class PrefixFileFilter extends AbstractFileFilter implements Serializable {

    /** The filename prefixes to search for */
    private final String[] prefixes;

    /** Whether the comparison is case sensitive. */
    private final IOCase caseSensitivity;
```

```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.io.Serializable;
import java.util.List;

import org.apache.commons.io.IOCase;

/**
 * Filters files based on the suffix (what the filename ends with).
 * This is used in retrieving all the files of a particular type.
 * <p>
 * For example, to retrieve and print all <code>*.java</code> files
 * in the current directory:
 *
 * <pre>
 * File dir = new File(".");
 * String[] files = dir.list( new SuffixFileFilter(".java") );
 * for (int i = 0; i &lt; files.length; i++) {
 *     System.out.println(files[i]);
 * }
 * </pre>
 *
 * @since Commons IO 1.0
 * @version $Revision: 606381 $ $Date: 2007-12-22 02:03:16 +0000 (Sat, 22 Dec 2007) $
 *
 * @author Stephen Colebourne
 * @author Federico Barbieri
 * @author Serge Knystautas
 * @author Peter Donald
 */
public class SuffixFileFilter extends AbstractFileFilter implements Serializable {

    /** The filename suffixes to search for */
    private final String[] suffixes;

    /** Whether the comparison is case sensitive. */
    private final IOCase caseSensitivity;
```

PrefixFileFilter.java - SuffixFileFilter.java

```java
    /**
     * Constructs a new Prefix file filter for a single prefix.
     *
     * @param prefix  the prefix to allow, must not be null
     * @throws IllegalArgumentException if the prefix is null
     */
    public PrefixFileFilter(String prefix) {
        this(prefix, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Prefix file filter for a single prefix
     * specifying case-sensitivity.
     *
     * @param prefix  the prefix to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the prefix is null
     * @since Commons IO 1.4
     */
    public PrefixFileFilter(String prefix, IOCase caseSensitivity) {
        if (prefix == null) {
            throw new IllegalArgumentException("The prefix must not be null");
        }
        this.prefixes = new String[] {prefix};
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Constructs a new Prefix file filter for any of an array of prefixes.
     * <p>
     * The array is not cloned, so could be changed after constructing the
     * instance. This would be inadvisable however.
     *
     * @param prefixes  the prefixes to allow, must not be null
     * @throws IllegalArgumentException if the prefix array is null
     */
    public PrefixFileFilter(String[] prefixes) {
        this(prefixes, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Prefix file filter for any of an array of prefixes
     * specifying case-sensitivity.
     * <p>
     * The array is not cloned, so could be changed after constructing the
     * instance. This would be inadvisable however.
     *
     * @param prefixes  the prefixes to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the prefix is null
     * @since Commons IO 1.4
     */
    public PrefixFileFilter(String[] prefixes, IOCase caseSensitivity) {
```

```java
    /**
     * Constructs a new Suffix file filter for a single extension.
     *
     * @param suffix  the suffix to allow, must not be null
     * @throws IllegalArgumentException if the suffix is null
     */
    public SuffixFileFilter(String suffix) {
        this(suffix, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Suffix file filter for a single extension
     * specifying case-sensitivity.
     *
     * @param suffix  the suffix to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the suffix is null
     * @since Commons IO 1.4
     */
    public SuffixFileFilter(String suffix, IOCase caseSensitivity) {
        if (suffix == null) {
            throw new IllegalArgumentException("The suffix must not be null");
        }
        this.suffixes = new String[] {suffix};
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Constructs a new Suffix file filter for an array of suffixs.
     * <p>
     * The array is not cloned, so could be changed after constructing the
     * instance. This would be inadvisable however.
     *
     * @param suffixes  the suffixes to allow, must not be null
     * @throws IllegalArgumentException if the suffix array is null
     */
    public SuffixFileFilter(String[] suffixes) {
        this(suffixes, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Suffix file filter for an array of suffixs
     * specifying case-sensitivity.
     * <p>
     * The array is not cloned, so could be changed after constructing the
     * instance. This would be inadvisable however.
     *
     * @param suffixes  the suffixes to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the suffix array is null
     * @since Commons IO 1.4
     */
    public SuffixFileFilter(String[] suffixes, IOCase caseSensitivity) {
```

PrefixFileFilter.java - SuffixFileFilter.java

```java
        if (prefixes == null) {
            throw new IllegalArgumentException("The array of prefixes must not be null");
        }
        this.prefixes = prefixes;
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Constructs a new Prefix file filter for a list of prefixes.
     *
     * @param prefixes  the prefixes to allow, must not be null
     * @throws IllegalArgumentException if the prefix list is null
     * @throws ClassCastException if the list does not contain Strings
     */
    public PrefixFileFilter(List prefixes) {
        this(prefixes, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Prefix file filter for a list of prefixes
     * specifying case-sensitivity.
     *
     * @param prefixes  the prefixes to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the prefix list is null
     * @throws ClassCastException if the list does not contain Strings
     * @since Commons IO 1.4
     */
    public PrefixFileFilter(List prefixes, IOCase caseSensitivity) {
        if (prefixes == null) {
            throw new IllegalArgumentException("The list of prefixes must not be null");
        }
        this.prefixes = (String[]) prefixes.toArray(new String[prefixes.size()]);
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Checks to see if the filename starts with the prefix.
     *
     * @param file  the File to check
     * @return true if the filename starts with one of our prefixes
     */
    public boolean accept(File file) {
        String name = file.getName();
        for (int i = 0; i < this.prefixes.length; i++) {
            if (caseSensitivity.checkStartsWith(name, prefixes[i])) {
                return true;
            }
        }
        return false;
    }
```

```java
        if (suffixes == null) {
            throw new IllegalArgumentException("The array of suffixes must not be null");
        }
        this.suffixes = suffixes;
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Constructs a new Suffix file filter for a list of suffixes.
     *
     * @param suffixes  the suffixes to allow, must not be null
     * @throws IllegalArgumentException if the suffix list is null
     * @throws ClassCastException if the list does not contain Strings
     */
    public SuffixFileFilter(List suffixes) {
        this(suffixes, IOCase.SENSITIVE);
    }

    /**
     * Constructs a new Suffix file filter for a list of suffixes
     * specifying case-sensitivity.
     *
     * @param suffixes  the suffixes to allow, must not be null
     * @param caseSensitivity  how to handle case sensitivity, null means case-sensitive
     * @throws IllegalArgumentException if the suffix list is null
     * @throws ClassCastException if the list does not contain Strings
     * @since Commons IO 1.4
     */
    public SuffixFileFilter(List suffixes, IOCase caseSensitivity) {
        if (suffixes == null) {
            throw new IllegalArgumentException("The list of suffixes must not be null");
        }
        this.suffixes = (String[]) suffixes.toArray(new String[suffixes.size()]);
        this.caseSensitivity = (caseSensitivity == null ? IOCase.SENSITIVE : caseSensitivity);
    }

    /**
     * Checks to see if the filename ends with the suffix.
     *
     * @param file  the File to check
     * @return true if the filename ends with one of our suffixes
     */
    public boolean accept(File file) {
        String name = file.getName();
        for (int i = 0; i < this.suffixes.length; i++) {
            if (caseSensitivity.checkEndsWith(name, suffixes[i])) {
                return true;
            }
        }
        return false;
    }
```

PrefixFileFilter.java - SuffixFileFilter.java

```java
159
160     /**
161      * Checks to see if the filename starts with the prefix.
162      *
163      * @param file  the File directory
164      * @param name  the filename
165      * @return true if the filename starts with one of our prefixes
166      */
167     public boolean accept(File file, String name) {
168         for (int i = 0; i < prefixes.length; i++) {
169             if (caseSensitivity.checkStartsWith(name, prefixes[i])) {
170                 return true;
171             }
172         }
173         return false;
174     }
175
176     /**
177      * Provide a String representaion of this file filter.
178      *
179      * @return a String representaion
180      */
181     public String toString() {
182         StringBuffer buffer = new StringBuffer();
183         buffer.append(super.toString());
184         buffer.append("(");
185         if (prefixes != null) {
186             for (int i = 0; i < prefixes.length; i++) {
187                 if (i > 0) {
188                     buffer.append(",");
189                 }
190                 buffer.append(prefixes[i]);
191             }
192         }
193         buffer.append(")");
194         return buffer.toString();
195     }
196
197}
198
```

```java
199
200     /**
201      * Checks to see if the filename ends with the suffix.
202      *
203      * @param file  the File directory
204      * @param name  the filename
205      * @return true if the filename ends with one of our suffixes
206      */
207     public boolean accept(File file, String name) {
208         for (int i = 0; i < this.suffixes.length; i++) {
209             if (caseSensitivity.checkEndsWith(name, suffixes[i])) {
210                 return true;
211             }
212         }
213         return false;
214     }
215
216     /**
217      * Provide a String representaion of this file filter.
218      *
219      * @return a String representaion
220      */
221     public String toString() {
222         StringBuffer buffer = new StringBuffer();
223         buffer.append(super.toString());
224         buffer.append("(");
225         if (suffixes != null) {
226             for (int i = 0; i < suffixes.length; i++) {
227                 if (i > 0) {
228                     buffer.append(",");
229                 }
230                 buffer.append(suffixes[i]);
231             }
232         }
233         buffer.append(")");
234         return buffer.toString();
235     }
236
237}
238
```